PROGRAM SYSTEMS: THEORY AND APPLICATIONS ISSN 2079-3316
HARDWARE AND SOFTWARE FOR DISTRIBUTED AND SUPERCOMPUTER SYSTEMS

UDC 004.272+004.382.2+004.8+004.43
10.25209/2079-3316-2024-15-2-139-473

# New generation of GPGPU and related hardware: computing systems microarchitecture and performance from servers to supercomputers

Mikhail Borisovich **Kuzminsky**✉
Zelinsky Institute of Organic Chemistry of RAS, Moscow, Russia
✉*kus@free.net*

Abstract. An overview of the current state of GPGPUs is given, with orientation towards their using to traditional HPC tasks (and less to AI). The basic GPGPUs in the review include Nvidia V100 and A100. Nvidia H100, AMD MI100 and MI200, Intel Ponte Vecchio (Data Center GPU Max), as well as BR100 from Biren Technology are considered as new generation GPGPUs. The important for HPC and AI tasks microarchitecture and hardware features of these GPGPUs, as well as the most important additional hardware for building computer systems with GPGPUs, that are CPUs specialized (albeit only possible for the initial period of their use) for working with the new generation of GPGPUs and interconnects — are analyzed and compared. Brief information is given about the servers (including multi-GPUs) using them, and new supercomputers (using these GPGPUs), where data on the achieved performance when working with GPGPUs was obtained.

The SDK of GPGPU manufacturers and software (including mathematical libraries) from other firms are briefly reviewed. Examples are given that demonstrate the tools of widely used programming models that are important for achieving maximum performance, while contributing to the non-portability of program codes to other GPGPU models.

Particular attention is paid to the possibilities of using tensor cores and their analogues in modern GPGPUs from other companies, including the possibility of using calculations with reduced (relative to the standard for HPC FP64 format) and mixed precision, which are relevant due to the sharp increase of the achieved performance when using them in GPGPU tensor cores. Data is analyzed on their "real-world" performance in benchmarks and applications for HPC and AI. The use of modern batch linear algebra libraries in GPGPU, including for HPC applications, is also briefly discussed.

Key words and phrases: GPGPU, V100, A100, H100, Grace, GH200 Grace Hopper, MI100, MI200, Ponte Vecchio, Data Center GPU Max BR100, CUDA, HIP, DPC++, Fortran, performance, HPC, AI, deep learning

2020 *Mathematics Subject Classification*: 65Y05; 68M20

Acknowledgments: The author thanks Alexnder Malyavko (NSTU) for assistance in preparing the text of the article

For citation: Mikhail B. Kuzminsky. *New generation of GPGPU and related hardware: computing systems microarchitecture and performance from servers to supercomputers*. Program Systems: Theory and Applications, 2024, **15**:2(61), pp. 139–473. `https://psta.psiras.ru/read/psta2024_2_139-473.pdf`

## Introduction

A widespread feature of modern computing systems, from servers to supercomputers, is the use of a heterogeneous construction of servers and cluster nodes, very often containing not only processors (CPUs), but also accelerators, primarily GPUs, areas of use of which are rapidly expanding. Here we mean GPGPU, but in what follows the abbreviation GPU will be used.

**Relevance of GPUs and areas of their using.** Currently GPUs are actively used for a wide range of very important tasks, including high-performance computing (HPC) and AI (AI tasks in this review also include all tasks of any type of machine learning, but the use of GPUs is especially relevant for deep learning). In addition, the use of multiple GPUs in one server (multi-GPU) is expanding. All this is connected with the main direction of growth in the performance of computing systems, mainly due to the strong increase in the number of cores and, accordingly, parallelization on them. The increasing importance of power efficiency over time also contributes to the focus on the use of GPUs, which contain many more functionally simpler cores than in the CPU, but with a reduced frequency. Thus, of the 50 leading supercomputers on the Green500 list for June 2023, only four did not use a GPU. Moreover, two of them — the Japanese supercomputers NA-J2 and MN-3 — used specialized rarely used multi-core processors (coprocessors), and the other two were based on multi-core ARM processors Fujitsu A64FX [1]. Another important advantage of GPUs is the achievement of high-density packaging of large computing resources, which is especially evident in servers containing several GPUs (multi-GPU).

As for the often cited use of GPUs in data centers, the term data center has now practically replaced the previously used term computer center, which can also be considered [2] as one of the parts of the data center. In reality, data centers often assume the use of GPUs for the above HPC and AI tasks, and the term data center itself is focused primarily on the use of cloud technology. Further in the text, references to data centers refer specifically to cloud technology, the tasks of which are not discussed in the review — specific benchmarks and applications for HPC and AI are considered here. Although GPUs began to be noted as the main accelerator in other areas, for example, sorting when working with databases [3].

To illustrate the widespread use of modern GPUs, we can use the Top500 supercomputer list, as it provides interesting statistics (see, for example, [4]). In the Top500, the world's performance leaders have been using GPUs for a long time. A notable exception to this rule in recent years was the Japanese supercomputer Fugaku, whose nodes were homogeneous and contained only the CPUs — the A64FX. He topped this list for more than two years. Probably, such a success of Fugaku was facilitated by the rather large number of cores (48 computing) in the A64FX [5]. However, in 2022, a new Chinese supercomputer Sunway appeared (the successor to the Sunway TaihuLight, which ranks 7th in the Top500, and in [6] classified as "pre-exascale"), containing heterogeneous 260-core SW26010 processors in its nodes — without a GPU. As another not very widely used alternative GPU option, we can mention the huge, specialized for AI tasks, Cerebras WSE-2 processors, containing 850 thousand cores [7]. But WSE-2 does not support greater precision than FP32, and the Andromeda supercomputer based on them [8] is accordingly absent from the Top500 list.

Statistics from the June 2020 Top500 list [9] indicated the use of accelerators in 26.6% of Top500 supercomputers (20.2% of supercomputers used Nvidia V100). In the June 2023 list, accelerators were used in 32.4% of all supercomputers (13% used Nvidia V100, 15.6% used Nvidia A100, 2.2% used AMD MI250X and MI210, 2% used Nvidia H100) [4]. The unambiguous modern leadership in the Top500 GPUs from Nvidia is obvious today and predictable for the near future. All data presented later in this review refers to the June 2023 Top500 list, and by default the Top500 list below refers to this June list.

**Features of the next expected GPUs.** Integration of CPU and GPU in one die is now becoming possible. For personal computers with conventional graphics processors, similar integration with the CPU has long been known, for example, in the form of the AMD APU (Accelerated Processing Unit), but here we mean the integration of the server CPU with the GPU. AMD expects such integration into the APU in the MI300 [10]. Intel talked about its plan to combine x86 chiplets together with GPU chiplets called Falcon Shores back in 2022 [11, 12], but its implementation will take more than one year. In a certain sense, a similar development from Nvidia, Grace Hopper [13–15], appears on the market earlier. But this whole direction is a possible way to intensify the use of the GPU itself.

**Limitations and difficulties of using GPU.** It must be kept in mind that GPUs are installed in only 32.4% of all supercomputers from the Top500 [4] (in the June 2022 list it was 30.2%). Performing calculations exclusively on GPUs is associated with the use of high-speed memory, but having a fixed and not very large capacity compared to the possible memory size of the servers or certain input data of applications (research objects), this may cause inefficiency on the GPU altogether. Therefore, for example, in the manual for specialized parallelization tools on Nvidia GPUs, CUDA [16], there is a section dedicated to exceeding the required memory capacity of the memory available on the GPU. In modern versions of GPU CUDA also provides the ability to work with virtual memory [16]. However, it is clear that actually working with virtual memory can lead to severe performance losses.

GPU computing involves the use of applications that are highly parallelized across a large number of cores. A classic example of this is molecular dynamics tasks and, especially, AI area. But this may not hold true for certain applications or even HPC areas. Therefore, in the tuning guide for applications using CUDA (for the Nvidia Ampere architecture used in the A100) [17], the first point of recommendations is to find a way to parallelize sequential code, which may mean the need to create new, improved algorithms that allow parallelization where in the "natural" algorithm it might be missing. It may also not meet the expectations of specialists in the relevant HPC fields, who may often be programming applications for this field themselves.

Since effective use of GPUs requires a very high level of parallelization scalability, this also requires the use of SDKs specialized for GPUs, and possibly an increase in the size of source code. Many HPC applications did not natively respond to this level of parallelism. Additionally, optimizing to the high expected level of GPU performance often requires many manual work, which is especially difficult when porting code from one type of GPU to another. All this complicates the work of programmers and can cause them some rejection.

The situation is simplified to a certain extent in cases where there are small parts of the program that limit performance (in the GPU world they become program kernels), which are often typical mathematical problems. And over time, more and more HPC applications are becoming capable of running on GPUs. For example, quantum chemical software systems that have been running on supercomputers for a long time are also moving in this direction. But for the most widespreads of the modern methods used there (without explicit non-empirical computations of electronic correlation), too long execution times can be associated with several mathematically different types of calculations (and not always related to those that are widespread in mathematical area; This is especially true for the use of gaussian basis functions), which requires a correspondingly much larger programming. For calculation by the widely used quantum chemical DFT method in a plane wave basis, a demonstration of the possible execution times of various parts of the program is given, for example, in [18].

Other important characteristics of using GPUs are cost indicators. If the goal is not to achieve an acceptable execution time at any cost (which is perhaps achievable only with the use of a GPU), then the relevant question becomes how much the cost of a computer increases when adding a GPU to it, and how much the application performance increases.

As an illustration, we indicate the acceleration data when calculating with the well-known Quantum Espresso software package, which is focused on calculations in the basis of plane waves using the quantum chemical DFT method. An illustration using the application of quantum chemistry, rather than the popular molecular dynamics on GPUs, was chosen here specifically — problems of quantum chemistry have long been performed on supercomputers, but the possibilities of quantum chemical calculations on GPUs began to appear later than in classical molecular dynamics — it is more difficult to implement, and the speedups achieved often smaller. Calculations by Quantum Espresso 6.5 were performed on a server with an 18-core Intel Xeon E5-2697 v4 (2.3 GHz), and adding V100 gave speedup in the range of 1.4-3.7 times [19]. The achieved acceleration naturally depends on the object being calculated. But we must keep in mind that this Xeon model began to be produced by Intel back in early 2016, and the calculation time was compared using only one processor.

As for the prices for new generation GPUs — this naturally applies to boards with a GPU (for example, an OAM module), they are not discussed in the review, since the corresponding "official" (for example, recommended by the manufacturer) prices for such new equipment are usually not available. But keep in mind that GPUs often provide greater power efficiency while requiring a relatively small square, so it's best to use total cost of ownership (TCO) rather than just price when evaluating GPUs.

**Modern realities of growing GPU use.** All of the above possible difficulties are gradually being resolved by creating new calculation methods and algorithms, new programming models for the SDK, as well as by improving GPU hardware. Naturally, this is reflected in the growing number of applications running on GPUs. The GPU application area is constantly growing, which, naturally, is most clearly demonstrate with Nvidia GPUs and was convincingly demonstrated at the latest GTC 34 (2022) and 35 (2023) conferences.

As time goes on, the number of supercomputers with GPUs included in the Top500 increases — with a GPU it is easier to obtain high performance in the HPL benchmark. The most powerful (as measured with the HPL) supercomputers in the world typically use GPUs. In the first twenty leaders of the Top500, only three supercomputers do not use GPUs (although the Chinese Tianhe-2A, which closes the top ten, also uses the Matrix-2000 accelerator, but this is not a GPU); The percentage of GPU utilization decreases further when considering a larger number of supercomputers.

But all this becomes weakly significant compared to the growing use of AI, which is covering more and more new areas of using — this primarily determines the requirements for GPUs (the HPC market is negligibly small compared to AI). Modern supercomputers included in the Top500 are also becoming AI-focused.

A review of the current global GPU market by renowned Chinese electronics industry analyst Chen Lizhong also suggests continued growth in the industry [20].

**Relevance of the review, selection of GPUs under consideration and areas of their analysis.** As general modern overview of different types of accelerators, including GPUs, can be considered [21] from the famous European BPG (Best Practice Guide) series. But today GPUs are characterized by ultra-fast development, and we can already talk about the emergence of a new generation of GPUs. In this review, GPU performance analysis focuses primarily on HPC tasks. There are publications that implement the fusion of traditional HPC fields with AI, for example, quantum molecular dynamics (QMD) and AI [22], or computational fluid dynamics and AI [23]. But currently combining traditional HPC tasks with AI methods may not be necessary (for example, for QMD — see [24]).

However, currently there is also an integration of HPC, AI tasks and processing of large volumes of data (as an example of work in the last 2 years, we can cite [25–30]), and benchmarks for this area have already appeared [31]. As an illustration of the active progress of work in this direction, we can note the consolidation of the well-known HPC developers of mvapich2 parallelization tools into new teams at Ohio University (US), where software tools running on top of mvapich2 are now being created— High-Performance Deep Learning (HiDL) [32] and High-Performance Big Data (HiBD) [33].

Given the potentially widespread use of AI in the commercial area, and the corresponding increased AI focus of modern GPUs [34], this review considers AI tasks for performance evaluations, although the paper is aimed primarily at traditional HPCs. The relevance of the analysis of modern GPUs is even increasing due to the emergence of the latest GPUs with higher performance (with their use, EFLOPS-level supercomputers are being created and are expected to be created) and the need for their optimal selection for the acquisition and use of appropriate hardware and software. To date, GPUs have come a very long way in the development of their architectures and performance indicators. Conventionally, the Nvidia V100 and A100 are classified as the modern "basic" generation in this review. This was chosen both because the V100 was the first to use tensor cores, and because of the breadth of use of these GPUs on modern supercomputers and servers. This GPUs review data are based on the V100 and A100 [21].

With the V100 and A100, this review will compare new generation GPUs— AMD Instinct (Radeon Instinct) MI100 and the MI200 family, Nvidia Hopper (H100), Intel Ponte Vecchio (Intel now produces a whole series of GPUs, Data Center GPU Max for which this is a codename), and partly the latest Chinese BR100 from Biren Technology. These GPUs are conventionally classified as a new generation, including because it was with their use that the exascale barrier was first overcome or it is planned to be further overcome (this applies to GPUs from AMD, Nvidia and Intel). The BR100 is included here due to its significantly higher specified performance indicators compared to the A100 [35]: at the processor level, Chinese developers have not previously outperformed processors from the US and Japan (for example, the ARM Kunpeng 920 [5] or the Zhaoxin x86 processor [36]), but the appearance in 2022 of the SW26010pro processors containing 390 cores with a total peak performance of more than 14 TFLOPS (by default in the text of this review, double precision, FP64 is assumed) [37] and the BR100 GPU gave such a high performance achievement that, one might say, for the first time allowed the Chinese industry to surpass the performance of some similar US products.

The relevance of a comparative analysis of AMD MI100 and MI200 with the above GPUs from Nvidia, Intel and Biren Technology seems obvious. The AMD MI250X began to be produced primarily for Frontier, which became the world's first exascale supercomputer [38–40], but is already used in about ten different supercomputers from the Top500, including the third-ranking supercomputer LUMI [41, 42], and GPUs actually determine the maximum performance, achieved there in the Top500. The well-known supercomputers Summit and Sierra with V100 nodes have been in the top ten Top500 for a number of years. Intel $X^e$-HPC Ponte Vecchio GPUs will be used in the US Argonne National Laboratory's Aurora supercomputer, where peak double precision performance is expected to exceed 2 EFLOPS [43], and in the SuperMUC-NG supercomputer upgrade at the Leibniz Supercomputing Center in Germany [44].

In addition, the most energy-efficient supercomputers are built on new-generation GPUs — for example, Henri with H100 heads the Green500, and supercomputers with MI250X occupy all the places there from 2 to 7 positions.

The relevance of comparing MI100 with Nvidia GPUs was recently noted in [45],and now GPU comparison has become even more important due to the emergence of new higher performance and more energy efficient GPUs. MI100 and MI200 are already actively used in HPC and AI. Much attention is paid to the performance data of the MI250X and A100 GPUs, obtained using the latest HPE/Cray EX supercomputer systems containing them [46].

The new generation of GPUs is distinguished not only by the construction of exascale supercomputers on them (Frontier — on MI250X [39], Aurora — on Ponte Vecchio [47], and the Selene supercomputer, which ranks 9th in the Top500, was supposed to be replaced with H100 [48]), but also by using other specialized hardware with them. First of all, these are interconnects (for example, Nvidia NVLinik [13, 49, 50], AMD Infinity Fabric, also characterized by regular improvements of versions [10], or the

CXL standard in BR100 [51]). These hardware closely related to GPU are discussed in this review. Some server processors — for example, ARM — Grace processors from Nvidia for work with GPU Hopper [13–15, 52] or AMD EPYC Zen 3 with (alleged) support of Infinity Fabric 3.0 in the I/O die [53]) were originally intended to work in conjunction with new GPUs, and are also discussed in the review.

But these CPUs could also be targeted at HPC or AI applications without using of GPUs. The Intel Xeon Max series [54] (codenamed Sapphire Rapids), which was originally intended to be used in the GPU-containing nodes of the Aurora supercomputer, can be used independently of the GPU.

For future EFLOPS-level supercomputers, the EPAC accelerators being developed within the European Processor Initiative (EPI) may be of interest, which are based on RISC-V with the ability to work with vectors of length 256 numbers in the FP64 format [55]– but these are accelerators that are not related to the GPUs, and EPAC is still at the development stage (only its test version 1.0 is available [56]), and a chiplet is being constructed from a number of tiles of various types, where EPAC is only one of them [57]. Accordingly, EPAC is not within the scope of this review.

**The review consists of sections with subsections.** Section 1 discusses the general hardware and software features of GPUs from different manufacturers. Section 2 analyzes the new Chinese GPU, Birentech BR100. Section 3 analyzes Intel Data Center GPU Max (Ponte Vecchio). Section 4 analyzes Nvidia GPUs: in Section 4.1 — A100, and in Section 4.2 — H100. Section 5 analyzes the AMD MI200 GPUs. In conclusion, general conclusions are drawn.

All sections review the hardware and software (SDK) for the respective GPUs and provide an overview of available performance data. In Section 3 and Section 5, and in Section 4.1 and Section 4.2, this is implemented as separate lower-level subsections. When comparing data, primarily on performance, a comparison was also used with Nvidia V100 performance data, and in Section 5 there is a separate Section 5.3.1 with AMD MI100 performance data.

The review necessarily uses a very large number of abbreviations. The author often provides explanations of well-known abbreviations, keeping in mind the possible reading of the text by specialists from different fields. A list of abbreviations used in several different sections of the review (in sections about different GPUs) is given in the appendix.

## 1. Common features for GPUs from different manufacturers

Before considering specific GPUs from different manufacturers, it is necessary to at least list the main software development tools (programming models) used on modern GPUs with a focus on HPC and AI tasks. Maximum performance is usually achieved using, of course, SDKs that are clearly focused on hardware manufacturers: for Nvidia — CUDA (Compute Unified Device Architecture) [16], for AMD — HIP (Heterogeneous Computing Interface for Portability) [58], part of the overall ROCm software stack (lower-level software tools are not discussed in this section of the review). The noticeable appearance of alternative GPU manufacturers to Nvidia on the market has increased interest in SDK components that work with various types of accelerators. HIP already has the ability to work with Nvidia GPU [58].

Among the programming tools that are not oriented towards working with the GPU of a certain manufacturer, we first note OpenACC and modern versions of OpenMP (support for working with accelerators appeared in OpenMP version 4.0, and since 2021 there is already a 5.2 specification [59]). Later, OpenCL (Open Computing Language) [60], and then SYCL [61] — an open standard for heterogeneous programming, became more widely used as tools for developing programs for GPUs and PGA accelerators. SYCL is developed by the Khronos Group, and (beginning from SYCL 2020) is based on C++17.

Data Parallel C++, developed by Intel (DPC++) [62] is also an open cross-architecture language built on C++ and SYCL — may become widespread. DPC++ uses SYCL with extensions that are expected to be included in future versions of the SYCL standard. OpenCL, SYCL and DPC++ can also be used for CPUs. Of these, DPC++ now appears to be the most advanced; A benchmarks already appeared on its basis [63].

Finally, GPU software mentioned here also includes Kokkos [64, 65]. Kokkos (supported in a US Department of Energy project) targets exascale supercomputers, uses C++, and aims to be «hardware-neutral». It can use, in particular, CUDA, HIP, SYCL and OpenMP as a back-end. A famous example of an application using Kokkos is the LAMMPS package of programs for molecular dynamics [66].

The listed software tools reflect the growing use of C/C++ in the areas of HPC and AI. But the question of the achieved performance compared, for example, with CUDA programs on Nvidia GPUs requires further study.

The functional simplicity of GPU cores makes it possible to quickly switch thread context from active to passive and back, which is not true for the CPU.

Nvidia's long-term dominance of the GPU market has led to the widespread use of terms for GPUs proposed by Nvidia. But the emergence of a new generation of GPUs, including from other companies, was characterized by their use of other terms for the same things (most of them are SIMT terms for APIs — CUDA, HIP, OpenCL and others). Accordingly, there are many publications and conference reports that provide correspondences between terms from different manufacturers, including in tabular form (see, for example, [67, 68]). Below in Table 1 such a comparison is made for the purposes of this review.

All rows of the table, except the last two, are API terms. The last two lines contain terms for similar important hardware components of GPUs from different manufacturers. This table does not include the terminology used for the BR100.

The table in the right column shows in bold the terms that will be used later in this review as common for GPUs from different manufacturers (although in sections about a specific manufacturer its terminology is also used).

The used by GPU manufacturers terms may vary depending on their using for hardware or software, and may change as new models become available. Thus, AMD uses the term wavefront in the architecture and ISA manuals discussed in the review of this company's GPUs — but in the modern HIP manual only warp is used [58]. And the emergence of a new generation of Nvidia GPUs caused the emergence of a new term for them — a cluster of thread blocks for the H100 GPU [16] in the hierarchy of various levels of thread groups.

Table 1. A comparison of terms used by various GPU manufacturers, including their programming models

| Nvidia (CUDA) | AMD (HIP) | Intel (oneAPI/ SYCL) | Desrription; **the general term used in the review** (if used as a general term) |
|---|---|---|---|
| Thread | Work item; Thread | Work-item | Individual thread (they work together in a group of threads — in a warp or in a sub-group); **thread** |
| Warp | Wavefront; (sometimes Warp) | Sub-group | A set of operations (threads) that execute synchronously, execute the same instructions, and follow the same control flow path: a group of parallel threads executed by a hardware unit (Nvidia SM has 32 of them) is the smallest computing unit, a block of threads per they are divided; **warp** |
| Thread block | Workgroup | Work-group | A group of warps/sub-groups running concurrently on a GPU (running on a single SM on an Nvidia GPU). Can synchronize together and communicate using shared memory; **thread block** |
| Grid | Grid | ND-range | A grid of blocks of threads, the top level of the hierarchy of the thread system of the entire GPU; **thread grid** |
| Streaming Multiprocessor (SM) | Compute Unit (CU) | $X^e$ core | An analogue of a functionally simplified CPU core (contain parallel ALUs). For example, in Intel Data Center GPU Max $X^e$ core contains several SIMD-type ALUs. |
| Tensor core | Matrix core unit | Matrix Engine (XMX) | Computational unit for multiplying small matrices (similar to GEMM, with mixed precision); **tensor core** |
| Shared memory | Shared memory | Local memory[1] | High-speed (cache-like) low-capacity memory shared by all threads in a thread block/work-group; **shared memory** |
| Global Memory[2] | | | DRAM memory available in the GPU; its data passes through several levels of cache memory |
| Device[2] | | | GPU (with the memory); **device** |
| Host[2] | | | Processors and memory (more generally — the entire part of the computer without the GPU); **host** |
| Kernel[2] | | | Part of a program executed on the GPU (function in C, subroutine in Fortran). Kernel can run in parallel with the CPU; **kernel** |

[1] Incomplete compliance;
[2] a common term for all GPU developers.

Nvidia terms are taken from [16]; AMD — from [58]; Intel — from [69].

Different levels of thread groups allow you to effectively organize highly scalable SIMT parallelization. Since a situation may arise where a warp is waiting for data from memory, the active calculation then switches to another warp. In classic Nvidia GPUs, for this purpose, SM contains a warp scheduler (it forms a warp group of threads) and a dispatch unit for activating warp execution. Similar hardware units exist in GPUs from other manufacturers.

Another very important common feature of modern GPUs is the ability to work with data of varying precision, including mixed precision operations. This, when using reduced precision and, accordingly, the number of bits to represent a number, makes it possible to achieve several times higher peak performance, reduce the requirements for GPU memory size and its bandwidth, which very often limits performance. When reducing the required memory capacity, the reduced amount of GPU communication with the CPU can also improve performance. This doesn't make sense when working with traditional CPUs, and all floating point calculations in HPC are done traditionally with FP64. However, for very actively developing AI areas, working with neural networks uses matrix multiplication, and it has been found possible to work with lower precision and with mixed precision.

The typical data format for use in deep learning is single precision, FP32, but many works have shown that lower precision, such as FP16, is sufficient [70]. The calculation time for deep learning is limited usually by matrix multiplications, which is what tensor cores in Nvidia GPUs or their analogues in other new generation GPUs are focused on. The tensor core in a GPU first appeared in the V100, and from the very beginning it was considered as an application specific integrated circuit (ASIC) integrated into the GPU — see, for example, [71]).

Formula (1) reflects the BLAS function **GEMM** (here **A**, **B**, **C** are two-dimensional matrices, the dimension of **A** is $M \times K$, the dimension of **B** is $K \times N$, the dimension of matrix **C** is $M \times N$).

$$C = \alpha A \times B + \beta C \tag{1}$$

Already in the first tensor cores (in V100), the FP32 format was used for **C**, and FP16 — for **A** and **B** [72]. In the A100 you can use BF16 for **A** and **B**, and TF32 for **C** (although in the A100 it is now possible to work with the FP64 format in tensor cores) [73].

Table 2. Reduced precision floating point formats on GPUs

| Number format | Number of bits | | | |
|---|---|---|---|---|
| | Number's sign | Exponent | Mantissa | In register[1] |
| FP32 | 1 | 8 | 23 | 32 |
| TF32 | 1 | 8 | 10 | 32 |
| TF32+ | 1 | 8 | 15 | 32 |
| FP16 | 1 | 5 | 10 | 16 |
| BF16 | 1 | 8 | 7 | 16 |
| FP8-E4M3 | 1 | 4 | 3 | 8 |
| FP8-E5M2 | 1 | 5 | 2 | 8 |

[1] TF32+ format is only supported by BR100 [35], and FP8 formats are supported by H100 [78].

This table uses data from Table 11 in [79] with the addition of a TF32+ format line for BR100 [35].

Similar mixed-precision matrix operations are performed in modern GPUs on special matrix blocks (see terminologies from different manufacturers in Table 1) and are carried out for matrices of very small sizes from a fixed set. For example, in tensor cores A100 for all matrices from formula (1) with FP64 format $M \times N \times K = 8 \times 4 \times 8$ [17]. Many reduced-precision floating-point number formats have begun to be used on GPUs (primarily for AI tasks); The basic parameters of formats with reduced (relative to FP64) precision are shown in Table 2 (this table shows only formats for floating point numbers — but in AI it is also possible to work with integers reduced to 8 bits in length, INT8).

Some of these reduced precision formats are not supported by the IEEE-754 standard [74], but are supported by specific GPU manufacturer models (TF32, TF32+, BF16, and FP8 formats). It should be noted here that the TF32 and BF16 formats are considered effective for deep learning (see, for example, [17, 75]). TF32 uses the same 10 bits for the mantissa as FP16, but due to the longer exponent, the range of numbers represented is larger, which is important for AI tasks [76]. And in [77] the possibility of using FP8 formats for deep learning is considered.

Since using reduced-precision formats on the GPU can lead to very important performance increase, little by little the ability to work with reduced (relative to FP64) precision has begun not only to be used in AI, but to be studied in other well-known HPC areas, including: FP32 in CFD (there were also attempts to work with FP16) [80], FP32 in classical molecular dynamics (in [81] the performance increase on FP32 was measured not on the GPU), FP32 in quantum molecular dynamics (there were also attempts to work with FP16) [82, 83], in quantum chemistry [84, 85]. The corresponding increase in performance may be due not only to a direct increase in the actual performance of the GPU cores due to a decrease in precision, but also to a possible dramatic reduction in the requirements for GPU memory capacity.

Naturally, studies began to appear on achieving acceptable precision of results when working with reduced precision in mathematical methods, for example, when solving the Poisson equation [86]. Methods for correcting possible errors relative to FP32 during calculations with FP16 and TF32 (when working on A100 tensor cores) are proposed in in [87]. It is clear that when working with reduced precision, fairly detailed systematic studies are required, which may not have time to be carried out due to the ultra-fast development of modern GPUs and the creation of new data formats in them.

Even in AI, the use of, for example, TF32 with a mantissa reduced relative to FP32 makes detailed studies relevant due to possible problems with the convergence of deep learning. Therefore, the TF32+ format available for BR100, which has a larger number of bits for the mantissa than in TF32, may be interesting. And, for example, in molecular dynamics, modern software packages running on the GPU often have options that allow calculations with reduced precision (for example, for two-point atom-atom interactions) and in a large number of cases give acceptable results — however, sometimes this leads to error. The author is not aware of publications that formulate in which cases such errors occur. For quantum chemistry the situation may be more complicated, for example, in iterations with self-consistent total energy. The author is currently generally wary of HPC calculations with reduced precision.

But until now, tensor cores are considered as ASICs focused on machine learning tasks (see, for example, [88]), although recently there have been works aimed at expanding the application of matrix multiplication with tensor cores to HPC (see, for example, [89]). But in general, for HPC it's necessary to be based on the performance achieved by specific applications. And in [90] it was found that of the 77 known HPC benchmarks selected there, only 10 used GEMM. And from the point of view of power efficiency, the use of emulation of FP64 and FP32 formats with reduced precision on the V100 tensor cores is significantly worse than the use of vector cores there. Therefore, for wider use of tensor cores on HPC, from the author's point of view, they need hardware support for FP64, which Nvidia began with the A100.

In general, issues of working with numbers of different precision are of more general importance, not only for GPUs. For example, in [91] proposed a new combined multiply-and-add unit targeting HPC and AI tasks to handle formats having different precision. Some information about the achievable precision when running on tensor cores for matrix multiplication using mixed-precision operations is given very briefly later in Section 4.1.4, which discusses the achievable performance on the A100.

## 2. New Chinese GPU BR100

This review starts with the Biron Technology BR100 for a number of reasons. This accelerator differs quite significantly in design from more traditional Nvidia and AMD GPUs, even in the form of terminology used. There are almost no publications assessing the performance of the BR100 in benchmarks and applications, and the prospects for their further production have become doubtful due to US sanctions. Therefore, the terminology used for BR100 was not shown in Table 1. Nevertheless, the analysis of BR100 seems interesting, including as a possible effective alternative to modern Nvidia GPUs.

The appearance of these GPUs was clearly evident for two reasons — the high speed of development (they were made "almost from scratch" in just 3 years) and the declared superiority in performance of this Chinese GPU over the Nvidia A100 (the H100 simply did not exist then).

It should immediately be noted that the BR100 is very clearly focused on working in the field of AI, which allowed the developers to make a clear gradation of importance when designing the microarchitecture. The main available source of information on the BR100 (also used in this review) is a report at the 2022 Hot Chips 34 conference [35], and additional information is available on the developer's website [51]. There were also minor clarifications in the interview with Biren Technology director Zhang Wen [92]. Certain comparisons of the characteristics of the BR100 with other GPUs are then carried out only in relation to the A100, since this model is classified as a base model in the review.

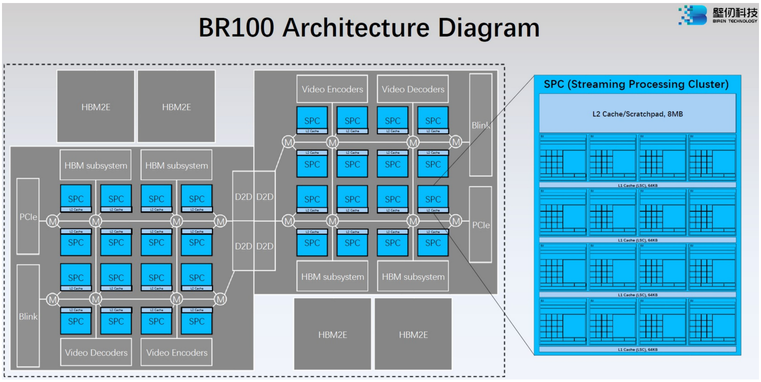


Figure 1. BR100 microarchitecture (figure from [35])

The overall microarchitecture of the BR100 is shown in Figure 1 [35].

The BR100 family contains two different models — the BR100 and the simpler, cheaper BR104, so BirenTech now also uses the BR10X name [51]. General characteristics demonstrating the success of the BR100 family are usually given for the BR100 model (see also Table 3). It should be noted here that the BR100 uses chiplets, is based on the use of two tiles (see Figure 1), and is manufactured using TSMC 7 nm technology (CoWoS 2.5D [35]). BR100 contains 77 billion transistors with a total area of 1074 mm$^2$. The BR104 has one tile, and many indicators are also half that of the BR100 (see Table 3).

The main component shown in Figure 1 that determines the achieved performance of the BR100 is the SPC (Streaming Processing Cluster), which can be partly considered a kind of analogue of the Nvidia GPC (Graphics Processing Cluster) in the A100. Each of the two BR100 tiles has 16 SPCs.

Each SPC contains 16 EU (Execution Unit) blocks, which contain the actual computing components of the GPU — 16 vector cores (V-cores), and one tensor core TDA (Tensor Data Accelerator) [35]. With a target clock frequency of 1 GHz (it is noticeably lower than the accelerated core frequency in the A100, see Table 13 below) and knowing the number of FP32 results achieved per clock cycle in the V-core (FP64 is not supported in the BR100, which is due to the focus on AI), this makes it possible to calculate peak performance with FP32. Table 3 provides data on the peak performance achieved when working with TDA (since they are especially relevant for AI tasks, which the BR100 is primarily focused on) [51]. The peak performance values achieved (for AI-relevant data

Table 3. Comparison of BR100 and A100 specifications

| GPUs | BR100[1] (Walli 100P) | BR104[2] (Walli 104P) | A100-PCIe[5] | A100-SXM4[5] |
|---|---|---|---|---|
| Technology, nm | 7 (TSMC) | | | |
| Form factor | OAM | Full-length two slot PCIe board | PCIe | SXM |
| Performance (peak):[4] | | | | |
| FP32 (TFLOPS) | 240 | 112 | | |
| TF32+ (TFLOPS) | 480 | 224 | 156/312[3,6] | 156/312[3,6] |
| BF16 (TFLOPS) | 960 | 448 | 312/624[3] | 312/624[3] |
| INT8 (TOPS) | 1920 | 896 | 624/1248[3] | 624/1248[3] |
| Memory type and capacity | HBM2E 64 GB | HBM2E 32 GB | HBM2E 40 GB | HBM2E 80 GB |
| Memory bus width (bits) | 4096 | 2048 | 5120 | 5120 |
| Peak Bandwidth (TB/s) | 1.64 | 0.819 | 1.9 | 2.0 |
| Interconnect to GPU, Peak Bandwidth (GB/s) | BLink (8 ports ×8), 448 | BLink (3 ports ×8), 192 | NVLink3, 600 | NVLink3, 600 |
| Interconnect to CPU | PCIe-5.0, ×16 with CXL support | PCIe-5.0, ×16 with CXL support | PCIe-v4 ×16 | NVLink3 |
| TDP, W | 550 | 300 | 250 | 400 |

[1] see [99];
[2] see [100];
[3] after the slash data is given when using sparsity;
[4] data using tensor cores are presented;
[5] data from [93, 94];
[6] for A100 data is given for TF32.

formats) are only slightly below initial expectations [35] and 1.5–2 times higher than in A100.

In fact, in the hierarchy from the SPC to EU level in BR100 there is an intermediate level — CU (Compute Unit) blocks, each of which can contain 4, 8 or 16 EU blocks (see the right side of Figure 1) [35]. CU can be considered an analogue of SM (Streaming Multiprocessor) in A100.

A CU with four EUs has a 64 KB L1 cache (LSC). Next in the memory hierarchy is the L2 cache with a capacity of 8 MB per SPC, which gives 256 MB for the entire BR100. HBM2E memory with a capacity of 64 GB (in modern A100 models the capacity is increased to 80 GB — see, for example, [93]) has an interface width of 4096 bits with a bandwidth that is also lower than that of the A100 with 80 GB [93] (see Table 3).

To achieve high GPU performance, memory bandwidth is important, and some lag here between the BR100 and the A100 is compensated by the huge capacity of the L2 cache (the A100 has a much smaller L2 cache capacity — 40 MB [94, 95]). To maintain more efficient operation of the L2 cache in the BR100, use Near Memory Computing [92]. Obviously, this is a certain analogue of the Near Memory Processing paradigm, close to the PIM, processing-in-memory paradigm, the goal of which is to spatially combine computing units with memory and greatly reduce data transfers between them [96], applicable for AI tasks [97, 98].

Turning to the bandwidth, it is equally important for communication between two BR100 tiles, and is 896 GB/s [35], which allows the BR100 to be treated as one common GPU.

For communication with the CPU, PCIe-5.0 ($\times16$) is used, with support for CXL (Compute Express Link) — an interconnect with support for cache coherence [101, 102], which has a clear tendency towards standardization.

Up to 8 BR100 GPUs can be installed in one server, and for communication between such GPUs, point-to-point communication channels (i.e., between each pair of GPUs) BLink are used, which uses SerDes (serializer/deserializer) [92]. One BLink has a bidirectional bandwidth of 64 GB/s [35], respectively, for 7 BLink channels connecting one GPU to all others, the total bandwidth is 448 GB/s. Choosing to fully support all point-to-point connections gives the BR100 an advantage due to the absence of potential contention when multiple GPUs share interconnect bandwidth, while GPU-to-GPU communication via the CPU has its downsides in bandwdith and latency [3]. Therefore, as noted in [3], the topology of such interconnect is very important.

For communication between the A100 GPU with the SXM4 form factor [103] the Nvidia NVLink3 interconnect [94] is used, where 12 channels with a bandwidth of 50 GB/s are used to connect the GPU-GPU — accordingly, a bidirectional bandwidth of 600 GB/s is obtained [94], which is much more than for BR100. The BR100's communication with the CPU has significantly higher bandwidth (896 GB/s) than the NVLink3's 600 GB/s. And in the A100 with PCIe-4.0 model, the communication bandwidth between GPUs is the same as that of the BR100, 64 GB/s [93].

It is clear that the effectiveness of a GPU interconnect can only be assessed by the measured performance of benchmarks or applications, which is practically non-existent for the BR100 at the moment. The observed orientation of applications to minimize all communications between the CPU and GPU (this is one of the basic rules of optimization in CUDA on the A100 [104]) suggests that scaling performance with increasing number of GPUs in a server with A100/SXM4 (in the case of rather large requirements for such communications) will be higher than in a server with BR100.

The advantage of BR100 is the use of OAM Spec v1.1 [92] — a rapidly spreading and actually claiming to standardize the OAM (OCP Accelerator Module) form factor [105] (the corresponding module with BR100 is called Walli100 [92]); And the BR100 is located on the UBB board [92], which can also become the standard of the future [105]. For comparison, the A100 with a high-speed NVLink3 GPU interconnect uses Nvidia's own SXM form factor, while the maximum number of GPUs in a server (for example, in the famous Nvidia DGX for AI) is also 8 [106].

Of course, BR100 provides a number of other features not discussed here — including 4 blocks of special functions (Special Function Units — SFU, analogues previously known in Nvidia GPUs, used, among other things, to calculate elementary functions); ability to work with NUMA and UMA; support for up to 8 virtual GPUs (Secure Virtual Instance, SVI — analogous to Nvidia MIG, Multi-Instance GPU), which allows several applications (that do not support good scaling to a full GPU) to effectively work with the one GPU simultaneously [35, 99]. Classic shader blocks for image processing are completely absent in the BR100, which is due to the unique orientation of this GPU towards AI (the video codec is supported [99], but this is not discussed in the review).

Another important modern GPU parameter is TDP — according to Table 3, BR100 needs more power consumption than A100. If we calculate power efficiency (performance per watt), then, for example, for the BF16 format relevant for AI in the BR100 it is higher than that of the A100 with SXM4 or PCIe (without using sparsity). The created Haixuan OAM server [92, 99] contains 8 BR100s and has a peak performance (for BF16) of about 8 PFLOPS with a maximum TDP of 7 kW [92] (see Figure 2) [35]. Together with Inspur, a well-known manufacturer of multi-GPU servers for AI, cluster solutions are also planned [92].

The BR104 not only has half the performance and memory capacity (see Table 3), but also the number of Blink ports is 3 — accordingly, fewer GPUs can be installed in one server. The announcement of the Wallen Technology Wallace 104 server with BR104 was reported in a number of media. The form factor is also important here — the BR104 uses a full-size two-slot board [100]. This can be compared with the A100-PCIe — based on them, Nvidia produces, for example, HGX modules (with a PCIe form factor) for servers, including two connected via the NVLink Bridge GPUs [93].

Regarding the BR100 (Bi Liren) architecture in a general sense, and not about the microarchitecture, it should be noted that there is a large set of supported data formats (Table 3 shows the performance for only some

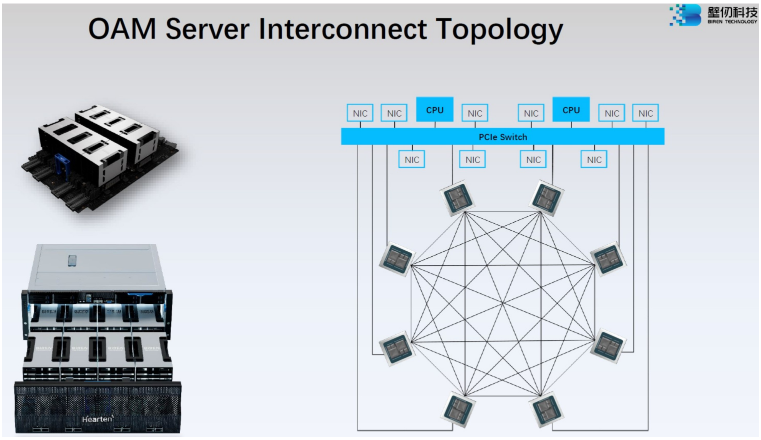


Figure 2. Interconnect topology in a server with BR100 (Figure from [35])

of them): INT8, INT16, INT32, FP16, BF16, FP32, TF32+. Information about them, primarily about the original development of Biren Technology TF32+, is available on a number of sites (see, for example, [107]). TF32+ looks like an attempt to improve the well-known TF32 format for tensor cores from Nvidia [94] (see Table 3), with the help of which the BR100 developers wanted to increase precision and performance [51]).

To work with BR100, a set of software tools BIRENSUPA (BIREN Scalable Unified Parallel Architecture) [92] — was developed — drivers, the BRCC compiler with support for extended C++, program libraries and other tools focused primarily on deep learning [35, 108]. BIRENSUPA has a programming paradigm and language style similar to NVidia CUDA, and also uses hardware capabilities unique to the BR100 [92]. But there were no publications demonstrating the use of these software tools and the actual performance achieved at the time of writing the review.

As for performance for AI, for BR104 there is data for two tests from the well-known set of benchmarks for the machine learning inference stage, *MLPerf inference datacenter* version 2.1 [109, 110] for data centers. The MLPerf inference datacenter benchmarks results show how quickly a trained neural network can perform inference tasks on new input data.

The first benchmark for image classification from the MLPerf Inference datacenter is based on ResNet (residual neural network) using the famous artificial neural network technology; ResNet has been expanded and modernized many times, and has been used, for example, for image processing for the diagnosis of COVID-19 (see, for example, [111]). Another

Table 4. MLperf 2.1 inference datacenter benchmarks data (December 2022) on servers with GPUs

| GPU models | Number of GPUs in the server | Server model and manufacturer | Image classification (A=99%) |  | Natural Language Processing (A=99.9%) |  |
|---|---|---|---|---|---|---|
|  |  |  | server (queries/s) | offline (samples/s) | server (queries/s) | offline (samples/s) |
| BR104, PCIe | 4 | Inspur NF5468M66[1] | 150027 | 212391 | 8993 | 11106 |
|  | 8 | Inspur NF5468M66-P[2] | 200052 | 424660 | 13952 | 22134 |
| A100, SXM, 80GB | 4 | Lenovo SR670v2[3] | 150027 | 174180 | No data |  |
|  | 4 | Dell PowerEdge XE8545[4] | 128029 | 131364 | 5297 | 5476 |
| A100, PCIe, 80GB | 8 | ASUS ESC8000A-E11[5] | 270066 | 283838 | 11496 | 13129 |
| A100, SXM, 80GB | 8 | Inspur N5688M6[6] | 313069 | 347202 | No data |  |
|  | 8 | Inspur N5488A5[7] | 290066 | 346954 | 13594 | 14977 |
| H100, SXM, 80GB | 1 | Nvidia Preview[8] | 58995 | 81292 | 6195 | 7921 |

[1] with Intel Xeon Gold 6354 and suInfer;
[2] with Intel Ice Lake-SP 8368 and suInfer;
[3] with Xeon Platinum 8360Y 2.40 GHz and with CUDA 11.6;
[4] with EPYC 7763 and with MaxQ, TensorRT 8.4.2 and CUDA 11.6;
[5] with 64-kernel EPYC 7763, TensorRT 8.4.0 and CUDA 11.6;
[6] with Xeon Platinum 8358, TensorRT 8.4.2 and CUDA 11.7;
[7] with EPYC 7713, TensorRT 8.4.2 and CUDA 11.7;
[8] with 8-kernel EPYC 7252, with TensorRT 8.5.0 and CUDA 11.8.

benchmark for work with natural language, BERT (Bidirectional Encoder Representations from Transformers), uses a transformer-based machine learning model to pre-train natural language processing using a bidirectional encoder [112]. This method is extremely widely used and is probably most famous for its use by Google LLC.

The results of these famous tests, presented in [109] for servers with 4 or 8 BR104 GPUs, as well as for servers with 4 or 8 A100 GPUs, are shown in Table 4, which selects the highest performance results achieved. The specified data for GPU BR104 and A100 refers to the class of available (that is, the corresponding servers can be purchased).

In these tests, the required accuracy of inference (A) is 99% or 99.9% relative to FP32. But we must keep in mind that the achieved performance may significantly depend on the software development systems used. For the A100, in addition to basic CUDA tools, to achieve high performance, To achieve high performance, special Nvidia TensorRT software [113] was used; For BR104, suInfer tools were used.

Performance data from MLperf inference datacenter 2.1 using 4 and 8 BR104 GPUs in Inspur servers showed the performance advantage of BR104 in the natural language processing (NLP) test with the BERT model by one and a half to two times when using the same number of BR104 or A100 in the server. In the "offline" scenario of the image classification test with the ResNet model on servers with 4 and 8 GPUs, servers with BR104 are also faster than servers with A100 (see Table 4), and in the "server" scenario of this test on servers with 4 GPUs, A100 is not ahead of servers with BR104, but significantly faster than them when using 8 GPUs. This may also be due to the advantage of NVLink3 over BLink with such a number of GPUs.

These tests may require less computational resources than MLPerf Training, often run on the A100 GPU: MLPerf Training measures the time required to train machine learning models to a target level of accuracy, where the main tasks are the actual training.

The above data about the BR100 clearly indicates that it is designed to compete with Nvidia GPUs, which should be supported not only by the higher peak performance of the BR100 and the high performance achieved in AI tests. A clear focus on extremely rapidly developing and commercially relevant AI tasks (which made it possible to target the BR100 hardware more narrowly and economically), and the use of hardware that claims to be standardized should help reduce the cost of the BR100, which is combined with the initially rather typical for Chinese manufacturers lower cost relative to products Western countries.

However, the situation with the BR100 changed dramatically due to US sanctions imposed in 2022, as a result of which TSMC stopped manufacturing and supplying BR100 chips. This has been widely discussed in various media, but here it should only be noted that this ban is aimed against possible competition with Nvidia and does not contribute to the acceleration of the development of the global GPU market.

## 3. Intel Data Center GPU Max (Ponte Vecchio)

The choice of Intel Ponte Vecchio (hereinafter abbreviated PVC) as a new generation of GPUs as the next object of analysis is due to the fact that at the time of writing the review they had just appeared on the market in the form of several different Data Center GPU Max models, and scientific publications about their performance are almost are missing.

Intel's information about available PVC models was changing at the time of writing this review. And the PVC architecture (and the software SDKs used) are quite different from the more "traditional" GPU architectures from Nvidia and AMD.

The appearance of PVC has been expected for several years; in September 2022, Intel announced the start of supply of PVC to the Aurora supercomputer at the Argonne National Laboratory in the US. But this supercomputer is missing from the June 2023 Top500 list.

## 3.1. PVC hardware

Intel developed the $X^e$ architecture ("eXascale for everyone") for use in a wide variety of classes of GPUs (not just GPGPU), to work with both PCs and servers. $X^e$ has a common instruction set architecture (ISA), and Intel uses 4 different microarchitectures — each class of GPU has its own microarchitecture [69]; PVC uses the $X^e$ HPC GPUs [114, 115]. For data centers, Intel offers Data Center GPU products, including two series — Data Center GPU Max [116], — this is the official name that replaces the use of the Ponte Vechio code word (abbreviation PVC in this review), and the Data Center GPU Flex series [117] with $X^e$-HPG microarchitecture.

The review considers only the family of HPC-oriented GPUs with the $X^e$ HPC-GPU microarchitecture — Data Center GPU Max (PVC is designed to work in exascale supercomputers; PVC further means the senior model of this series, Max 1550 — for the formation of the Aurora supercomputer, Intel supplied this GPUs, and data about it have been presented in a number of publications cited here in the review). Intel points to 3 different models in this family, the recommended price data for which was expectedly not provided on the website ark.intel.com at the time of writing the review — in accordance with the corresponding lack of similar price recommendations from other manufacturers of new generation GPUs. Table 5 shows the basic specifications of various PVC models. This table shows only GPU specifications that are primarily relevant for classic HPC tasks. And, for example, the number of blocks in PVC for processing ray tracing, which can also be used for AI tasks (see, for example, [118]), is not given here (there are as many of them in PVC as $X^e$ cores — 128) — because hardware capabilities of PVC for ray tracing are also not discussed in the review.

For the production of PVC, it was planned from the very beginning to use three-dimensional laying based on tiles [114, 120]. Intel could have been pushed to this point by a certain lag in its own semiconductor technology (sometimes formulated as rumors [121]). According to a report from the famous exascale computing project ECP [67], PVC was planned for delivery in 2021.

TABLE 5. Basic characteristics of Data center GPU Max series models (PVC)

| GPU model | Number of | | | Frequency (GHz) | | Memory | | TDP, Watt |
|---|---|---|---|---|---|---|---|---|
| | $X^e$ cores | XMX | XVE | Base | Max. | capacity, GB | bandwidth, GB/s | |
| 1100 | 56 | 448 | 448 | 1.0 | 1.55 | 48 | 1228.8 | 300 |
| 1350 | 112 | 896 | 896 | 0.75 | 1.55 | 96 | 2457.6 | 450 |
| 1450 | 128 | 1024 | 1024 | no data | | 128 | no data | 600 |
| 1550 | 128 | 1024 | 1024 | 0.9 | 1.6 | 128 | 3276.8 | 600 |

XMX($X^e$ Matrix eXtensions) — matrix units, XVE ($X^e$ Vector Engines)— vector units. The data in the table is taken from different (in time) versions [116] and [119]. Models 1350 and 1450 are marked in red because they are not listed on ark.intel.com at the time of review writing.

PVC uses three-dimensional Co-EMIB (Co-Embedded Multi-Die Interconnect Bridge) technology using chiplets, which promotes high performance, and with PVC we can talk about working with PIM architecture, aimed at solving the problem of exchanging large amounts of data with memory [122]. The construction of three-dimensional processors from several crystals (dies) in [123, 124] is indicated as a progressive way to ensure the continuation of Moore's law, and three-dimensional memory began to be made quite a long time ago, which was the case not only for HBM (see, for example, [125]).

Here it is necessary to point out an alternative option for integrating different dies into a whole (Intel's tiles in PVC) using chiplets, used by AMD, including when building the MI200 GPU. For a modern overview of chiplets, see [126], and modern AMD GPUs are discussed below. It should also be noted that the standard chiplets interconnect, which is being developed by a consortium of a number of companies, including Intel, AMD, ARM, Google and TSMC, includes not only the physical layer — there is now a UCIe 1.0 specification [127].

Model PVC contains 100 billion transistors arranged in 47 functional tiles (thermal tiles are not counted here), combined into 5 nodes [116, 128–130]. Of this large number of tiles, two tiles are basic, 16 are compute tiles (see Figure 3 [131]), 8 are HBM2E memory tiles. The PVC is structured as 2 hardware stacks [129] — each of the two base tiles contains 8 compute tiles and 4 memory tiles (the logical relationships of these PVC components are presented in Figure 3). Basic tiles also contain PCIe interfaces and channels to HBM2E.

These tiles, as well as the interconnect tiles between GPU PVCs, $X^e$ Link [129, 130] will be briefly discussed below. Rambo (Random Access Memory, Bandwidth Optimized) cache tiles were also planned for PVC [129], but for the $X^e$ architecture in [114] they are listed as optional, and they are not included in the microarchitecture datasheet [130].

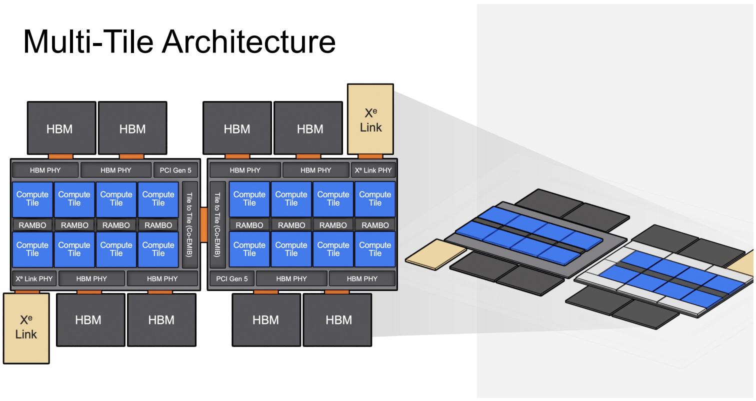


Figure 3. Tile-based PVC architecture (drawing from [129])

The implementation of advanced multi-chip 3D technology is discussed in more detail in [128]. Consideration of technologies is not within the scope of this review; It should be noted only the use in PVC of a 24-layer substrate, which ensures operation with overos three-dimensional stacking technology [132] and with 2.5D EMIB (Embedded Multi-die Interconnect Bridge) technology [133], which when used together they are called Co-EMIB [122]. EMIB is used to support local internal interconnects; Foveros with Intel 7 technology (formerly Enhanced 10nm SuperFin) is used in base tiles. Memory tiles are made using Intel 7 technology, Compute tiles are made using TSMC N5 technology, and HBM2E memory tiles are made using TSMC N7 technology. PVC is manufactured for the OAM 1.1 form factor [134]), which provides high equipment packaging density and, as noted above in Section 2, is intended to be used as a standard. Tiles are further considered not in technological terms, but simply as certain blocks of microarchitecture — accordingly, some types of tiles are not mentioned here at all.

It should be noted here that Intel also uses other terms (not just tiles) to refer to $X^e$ class of microarchitectures hierarchies [69, 130]. Subslices are not used for PVC, they are an analogue of $X^e$ cores in PVC. In PVC (Data Center GPU Max, below the level of the entire GPU, there are 2 levels of hierarchy — a slice ($X^e$ HPC Slice) and a stack ($X^e$ HPC stack). Slice has 16 $X^e$ cores and 16 ray tracers. The stack contains 4 slices (respectively with 64 $X^e$ cores and 64 ray tracing acceleration devices) [69, 115]. In addition, the stack has an L2 cache, a memory controller, a PCIe-v5 interconnect and 8 $X^e$ Link channels (see below). PVC scales up to two stacks — up to 128 $X^e$ cores [116, 130]. $X^e$ cores are analogous to SM in Nvidia GPUs, and comparing their number is often used when comparing different GPUs (as well as the number of cores in the CPU).

TABLE 6. The number of operations per clock cycle performed in one $X^e$ core and in the hierarchy of units performing calculations for different data formats [69]

| Execution units | Data format | Number of operations per cycle |
|---|---|---|
| $8 \times$ XVE | FP64<br>FP32<br>FP16 | 256<br>256<br>512 |
| $8 \times$ XMX | TF32<br>FP16<br>BF16<br>INT8 | 2048<br>4096<br>4096<br>8192 |
| Slice — 16 times more execution units and operations per clock cycle | | |
| Stack — executing blocks and operations per clock cycle are 4 times more | | |
| Double-stack PVC (model 1550) — still 2 times more execution units and operations per clock cycle | | |

XMX does not support the use of the FP64 format

A general description of the PVC microarchitecture is available in [130]. Each computing tile contains 8 $X^e$ cores, of which there are 128 pieces for the entire PVC. Each $X^e$ core has 8 XVE vector engines with 512-bit vector lengths and 8 XMX matrix engines working with 4096-bit operands [69, 129, 130], and has a $X^e$ Link, interface based on a coherent interconnect CXL [135] (also previously developed by Intel).

Accordingly, the PVC has a total of 1024 XVE vector engines and 1024 XMX matrix engines. XMX are an analogue of Nvidia tensor cores (this was indicated in Table 1 in the introduction), which are found not only in the V100 and A100 GPUs, but also in other Nvidia graphical processors [136].

XVE units operate on 512-bit operands and use multiply-and-add operations. This gives the $X^e$ core 256 FLOPS per clock for FP64 (and for FP32 too) [129].

Table 6 shows the number of operations performed per clock cycle with the different data formats available for XVE and XMX [129, 130]. This allows peak performance $P$ to be calculated using clock frequency $\nu$. So, for FP64 or FP32 formats when working with XVE $P = 128 \times 256 \times \nu$, that provides 52.4 TFLOPS (Table 7 shows numbers rounded to the nearest integer, taken from the Intel overview [130]). Considering the possible values of $\nu$, given in Table 5, it becomes clear that this numbers are calculated assuming operation at the maximum, and not at the base frequency.

From the data in Table 7, we can conclude that when using vector (without using matrix blocks) operations, peak performance for FP64, traditional for HPC, grows monotonically with the start date of new GPU models.

Table 7. Comparison of vector and matrix (XMX) peak performance of Intel PVC and GPUs of other companies for different data formats

| Performance for different formats | PVC | A100 | H100-SXM | H100-PCIe | MI250X |
|---|---|---|---|---|---|
| FP64 (TFLOPS)[1] | 52 | 9.7 | 33.5 | 25.6 | 47.9 |
| FP32 (TFLOPS)[1] | 52 | 19.5 | 66.9 | 51.2 | 47.9 |
| XMX TF32 (TFLOPS) | 419 | 156 | 494.7 | 378 | No |
| XMX BF16 (TFLOPS) | 832 | 312 | 989.4 | 756 | 383 |
| XMX FP16 (TFLOPS) | 832 | 312 | 989.4 | 756 | 383 |
| XMX INT8 (TOPS) | 1664 | 624 | 1978.9 | 1513 | 383 |

[1] Data without matrix operations (XMX in PVC does not support FP64). With the use of tensor cores, the peak performance of the H100, A100 and MI250X is twice as high. Data for Nvidia H100 are taken from [78], for A100 — from [73]; data om PVC — from [130, 138].

In the field of AI the FP32 format is normal, and it's possible use FP16/BF16, including on XMX (see the text above in the discussion of formula (1)). The square root operation in FP32 format, according to [69], gives four results per clock cycle in XVE.

For Intel graphical processors, not very low frequencies are observed; For example, Arc Alchemist can reach more than 2 GHz [137]. Previously, in [129] it was indicated that the FP32 peak performance was expected to be at least 45 TFLOPS — this most likely means that Intel managed to increase the clock frequency compared to last year's prototype in PVC.

Nevertheless we must keep in mind that the power consumed in this case increases in proportion to the frequency, and increasing the frequency to maintain reliable operation of the chip often also requires an increase in voltage, the square of which is proportional to this power. The TDP of the PVC model under consideration is 600 W, and assumes liquid cooling [129] (see also comparisons of Data Center GPU Max models in Table 5). In [129] another PVC model with possible air cooling and a TDP of 450 W is indicated — this corresponds to the Data Center GPU Max 1350 model.

However, in mid-2023, Intel announced the discontinuation of the Max 1350 model due to the creation of a modified version of the Max 1550 with air cooling, and plans to produce a new Max 1450 model [139].

Table 8 provides a comparison of PVC (by default, this refers to the only model 1550 available at the time of writing the review) with other GPUs considered in the review in terms of other very important for performance indicators.

TABLE 8. Comparison of PVC hardware specifications with other modern GPUs

| GPUs | PVC | H100 | A100-SXM4-40GB | MI250X |
|---|---|---|---|---|
| Technology | Intel 7 (7 nm) & TSMC N5 & TSMC N7 | TSMC 4N (5 nm) | TSMC N7 (7 nm) | TSMC (6 nm) |
| Memory capacity, GB | 128 | 80 | 40 | 128 |
| Memory type | HBM2E | HBM3 | HBM2 | HBM2E |
| Memory bus width, bits | 8192 | 5120 | 5120 | 8192 |
| Bandwidth, TB/s | 3.3 | 3.0 | 1.6 | 3.2 |
| Form factor | OAM | SXM | SXM | OAM |
| TDP, W | 600 | 700 | 400 | 560 |

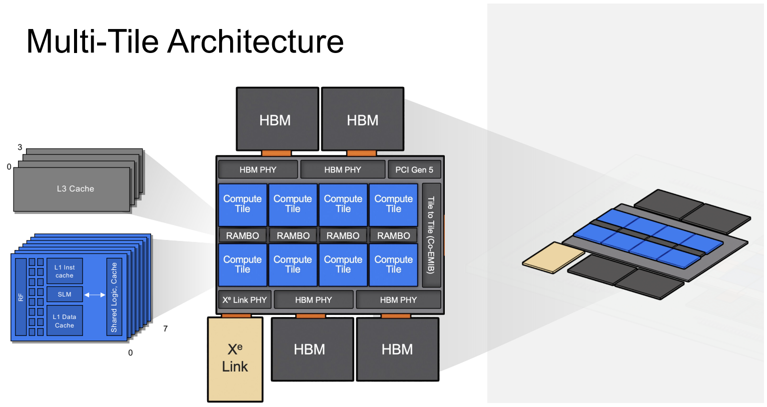


FIGURE 4. Memory hierarchy in PVC (Figure from [129])

Due to the high computing performance of modern GPUs, actual performance achieved when working with them is often limited by memory bandwidth, which happened before (see, for example, [140]), and now often happens (see, for example, [141, 142]).

A discussion of the PVC memory hierarchy should begin with the register file, which has a capacity of 64 MB (512 KB per $X^e$ core) and provides a bandwidth of 419 TB/s [120]. The L1 cache is traditionally divided into an instruction cache and a data cache, which has a capacity of 512 KB per core [129, 143]. The PVC also has SLM (Shared Local Memory) with a total capacity of 8 MB for the entire PVC [69] — see Figure 4.

Each compute tile has 4 MB L1 [129], a total of 64 MB per PVC, with a bandwidth of 105 TB/ [120]. The 408 MB L2 cache (204 MB per stack)[144] on the base tile [129] has a bandwidth of 13 TB/s [120]. Data for other Data Center GPU Max models is shown in Table 9.

Table 9. Memory hierarchy and interconnects of data center GPU Max series models

| GPU model | L1 cache, MB — per GPU/per one $X^e$ core | L2 cache, MB –per GPU/per one $X^e$ HPC stack | HBM2E capacity per one $X^e$ HPC stack, GB | HBM2E bandwidth, TB/s | Number of $X^e$ Link ports |
|---|---|---|---|---|---|
| 1110 | 28/0.5 | 108 | 48[1] | 1.2 | 6 |
| 1350 | 48/0.5 | 216/108 | 48 | 2.5 | 16 |
| 1450 | 64/0.5 | 408/204 | 64 | no data | |
| 1550 | 64/0.5 | 408/204 | 64 | 3.3 | 16 |

[1] Model 1110 has only 1 stack. The table data is taken from [69, 116] and [119]. Models 1350 and 1450 are marked in red because they are not listed on ark.intel.com at the time of writing.

In addition, in [129] it is indicated that the base tile contains a TLB buffer, traditional for conventional CPUs (which can also be considered some kind of cache), and also a RAMBO SRAM cache — see Figure 4. This cache consists of 4 banks with a capacity of 3.75 MB on a base tile that also contains a cache switch [129]. But the technical review of Intel Data Center GPU Max[130] does not mention the presence of a RAMBO cache.

The HBM2E memory itself in PVC combines up to 128 GB (8 tiles) and operates over a channel with a width of 8192 bits [120]. As stated in [69], the GTI (Graphics Technology Interface that connects the GPU to the rest of the computing system) bandwidth of a single PVC stack is 1024 bytes/cycle for reading or writing, which for a dual-stack PVC with a maximum frequency of 1.6 GHz (for 1550 model) provides bandwidth of approximately 3.3 TB/s (see Table 5). The bandwidth of this memory for other Data Center GPU Max models is also shown in this table.

Although each of the two EMIB-linked (up to 230 GB/s in both directions) stacks has its "own" L2 cache and 64 GB HBM2E memory [130],

Before considering scaling computing resources using the $X^e$ Link hardware available in Data Center GPU Max which provides coherent interconnection between GPUs in the server [130], it is necessary to point out the differences between different GPU models of this series (see Tables 5, 9). In addition to the older Max 1550 model, there is also data on the Max 1450, 1350 and 1100 models (as noted above, Intel no longer plans to produce the 1350 model).

Of particular interest is the future Max 1450 model, which has the same number of $X^e$ cores as the Max 1550. And Max 1110 consists not of two, but of one stack. The number of $X^e$ cores in different models is proportional to the total capacity of HBM2E and L1 and L2 caches:

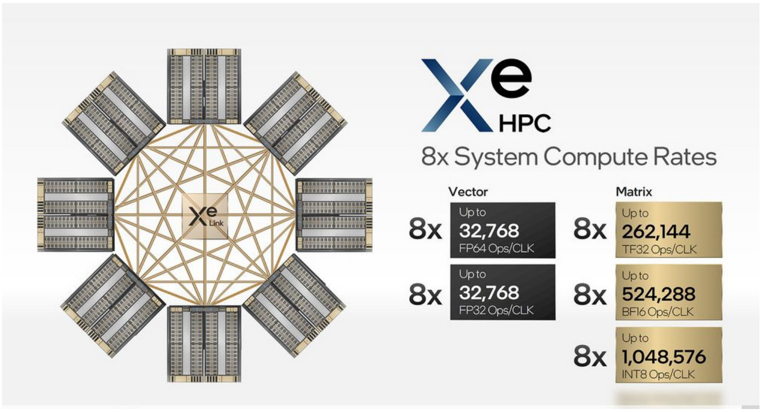

Xe
Link
Xe HPC
8x System Compute Rates
Vector
Matrix
8x Up to 32,768 FP64 Ops/CLK
8x Up to 262,144 TF32 Ops/CLK
8x Up to 32,768 FP32 Ops/CLK
8x Up to 524,288 BF16 Ops/CLK
8x Up to 1,048,576 INT8 Ops/CLK

FIGURE 5. Interconnect between PVCs (Figure from [146])

in the Max 1100 model the corresponding capacities are 28 and 108 MB. Naturally, as the number of $X^e$ cores in the models decreases, the TDP also decreases (see data in Table 5).

Here we can draw an analogy with three different models of GPU MI200 from AMD: the top model MI250X has two GCD (Graphics Compute Die), like the MI250 (which simply has fewer cores), and the bottom model MI210 has only one GCD (see below in the section about GPUs from AMD).

The bottom Max 1100 model is also distinguished by a reduced number of physical $X^e$ Link ports (see Table 9), but it also uses a form factor different from OAM—PCIe AIC (add-in card) [116] (add-in card). PVC, which can be classified as a system on a chip (SoC) [114] has a PCIe 5.0 $\times 16$ interface [130] with a bandwidth of about 63 GB/s [69]; In all Data Center models, GPU Max PCIe 5.0 is used to communicate between the GPU and the host [116].

The $X^e$ Link tile supports embedded switch and 8 $X^e$ Link channels, with each channel communicating directly to each (see Figure 5) [143]. And in two stacks there are respectively 16 $X^e$ Link channels [69]. In [130] reported a per-$X^e$ Link bandwidth of 26.5 GB/s in each direction. $X^e$ Link is designed to provide coherent communication between $X^e$ HPC stacks. This accordingly includes both internal communication in PVC (OAM) and between multiple PVCs (OAM) [130]. This interconnect, used for communication between multiple GPUs in a server, allows for efficient operation in SYCL with USM (Unified Shared Memory) mode [69].

What's important is how the Max 1550 and Max 1450 logically look to the user. Given the $X^e$ Link bandwidth mentioned above, the communication bandwidth between the two stacks is much lower than the memory bandwidth on a single stack. In the AMD MI250X/MI250, which is similar in this regard, therefore each GCD there logically looks like one GPU. In [144] states that users treat two PVC tiles as two processors. However, in Intel's oneAPI GPU Optimization Guide [145] states that moving from one stack to two requires simply changing an environment variable, meaning the Max 1550 gives you the choice of running it as two different GPUs or as one big GPU.

PVC officially began to be supplied by Intel already at the time of writing the review, and many details at the hardware level were not yet available, for example, latencies values, which is also important for communications via $X^e$ Link, including between PVC stacks.

In [130], two types of solutions supported by Intel for scaling computing systems due to an increase in the number of $X^e$ HPC stacks are indicated: scaling inside (scale-up) — inside multi-GPU servers, and scaling outside (scale-out), which refers to the cluster, containing GPU nodes. Above we actually talked about scale-up scaling.

The scale-out solution can scale to a cluster containing a maximum of 64 OAMs interconnected via $X^e$ Link Glueless. Further scaling along with Infiniband allows to build a system with up to 512 interconnected OAMs [130]. "Glueless" interconnects ((where are no intermediate chips) have been known for a very long time; They provide very low latencies [147].

And first it is planned to use the Tuscany configuration, in which a common board with four OAMs can be connected to the server via PCIe 5.0. Options are available with Data Center GPU Max with TDP 450 or 600 W, with air or liquid cooling [130]. Taking into account the data in Table 5, they probably meant the GPU Max 1350 and 1550 models, respectively.

PVC was originally intended to be used in two-socket servers with the new Xeon Sapphire Rapids CPUs (now called Xeon Max they were used in Intel's server performance data [116]). Intel's currently offered Xeon Max supports DDR5 and HBM2E (see, for example, [148]). Using HBM in these CPUs can greatly improve performance, as shown in [149].

In the Aurora supercomputer, the computing nodes are dual-processor servers (with a Xeon Max 9480 CPUs) containing another 6 PVCs [47]. In addition to the Aurora supercomputer being created, PVC is planned to be used on the European exascale supercomputer with SiPearl Rhea processors [150]. Of course, this strengthens PVC's potential position in the market, but it will be more important to consider the realities of performance achieved in applications, transfer of software to PVC and cost indicators. Data Center GPU Max of course, is intended to be used not only on supercomputers, but also at the level of small clusters and separate servers with GPUs — such servers have already been announced by famous manufacturers, including Dell, Lenovo and Supermicro.

## 3.2. Software tools and first initial data on PVC performance

As for software, the existing publication [120], taking into account the work for the Aurora supercomputer being created at the Argonne National Laboratory in the US [67, 151, 152] indicates the accelerated movement of Intel towards the SYCL standard line from the Khronos Group [153], extended by Intel in the form of DPC++ (Data Parallel C++) [154]. The relevance of this direction is clearly, since DPC++ is focused on working with different accelerators, which provides portability, including between different GPUs. Intel's DPC++ compiler is part of the overall Intel oneAPI software [155], which can also run on Intel processors and PGA accelerators, giving developers the ability to create a degree of common program code. Intel is actively developing new versions of oneAPI — for example, oneAPI 2023 is now available [155].

The oneAPI software includes a wide range of tools — compilers, debuggers, profilers, libraries and specialized tools for working with AI. The oneAPI DPC++/C++ compiler makes it possible to work with GPUs from Nvidia and AMD [156]. For optimization tasks when working with oneAPI, Intel has prepared a special guide [69]. A big advantage of DPC++ itself is the presence of an open source compiler [157].

In [158], for V100 on the STREAM (triad) benchmark, the bandwidth achieved with DPC++ was close to that obtained using OpenCL and CUDA, and the use of SGEMM from the oneMKL library gave 66% of the peak performance. Obviously, these results can be improved when working with new versions of oneAPI.

Since oneAPI is an open, standards-based unified programming model [159], aimed at working with both processors and accelerators of different types and from different companies [156], taking into account the end of the GPU monopoly from Nvidia and the transition to the use of GPUs from different companies, this is definitely promising. But it is clear that proposals from different companies may evolve in the direction of unification (for example, AMD's HIP software is already focused not only on AMD GPUs), and in this actively developing area it is now difficult to predict what will become the most used.

Intel, of course, has also taken care of the pressing issue of possibly porting code from Nvidia GPUs to PVC, and in oneAPI it also provides the DPC++ Compatibility Tool (often used abbreviation — DPCT), which automatically translates CUDA code to DPC++. According to [160], such automatic porting typically does this for 90–95% of the CUDA code, and full completing the porting of all code requires manual work. And how effectively all this will work on the GPU in terms of performance, of course, there is practically no data yet. In [154] does some preliminary research, but it does not apply to the GPUs covered in this review, and it is unclear how relevant it is to HPC workloads. In [161] based on the experience of porting SGEMM from the MAGMA library, it was concluded that DPCT can be successfully used for the initial port of CUDA code to DPC++, but the results used here are for GPUs that are not included in the review.

In general, as examples of work on porting GPU-using applications to DPC++ should indicate the porting of several well-known molecular dynamics applications to DPC++ — NAMD [162], GROMACS (ported to SYCL, using the DPC++ compiler [163, 164]); LAMMPS can also work on PVC — but through the use of Kokkos. Data about similar porting of other areas applications are available in [154]. In [165] discusses the use of SYCL on Nvidia and AMD hardware by using hipSYCL with different backends (now hipSYCL is renamed to *AdaptiveCpp*URL).

The tasks of porting program codes from CUDA to oneAPI have already been considered in a number of articles, since oneAPI tools are applicable to various graphical processors that appeared before PVC. Thus, in [166], for numerical integration problems, porting from CUDA (with V100) to oneAPI required special manual optimization before performance with oneAPI became slightly lower than with CUDA. In [167], when porting unstructured grid CFD code to SYCL for the A100 and V100, hipSYCL was used, but concluded that CUDA was still better for achieving maximum optimization. For the same CFD area, a similar result was obtained in [168]. These works noted that better results for DPC++/SYCL can be obtained in the future as the quality of optimization by compilers increases. The report at the supercomputer conference at NASA on the transfer of the FUN3D software package from CUDA to oneAPI (for CFD, with the solution of the Navier-Stokes equations) [169] essentially also indicated the advisability of manually optimizing the code for a specific architecture.

It is also worth pointing out that oneAPI has a certain disadvantage associated with being based solely on C++, since this does not help codes that natively use modern Fortran. For CUDA there is CUDA Fortran [170]. At the same time, the modern Fortran language when working on the

GPU began to be considered versions of the standard with support for object-oriented tools [171]. And modern versions of OpenMP, also implemented in Fortran compilers [172], have suitable tools for working with GPUs. But the above disadvantage is associated with delays in the development of possible new extensions of modern Fortran standards with tools for efficient work with GPUs.

A more detailed analysis of Intel's oneAPI is not practical here due to the very small number of scientific publications demonstrating the performance of PVC compared to other modern GPUs, including comparisons with work with other software such as CUDA. Intel has achieved a lot primarily with oneAPI, as these tools, including DPC++, have begun to be used on other Intel GPUs — for example, working with the well known Ginkgo library [173]. In addition, DPC++ learning for students studying programming of heterogeneous computing systems is already being implemented [174].

Among the isolated publications with data about the performance of Data Center GPU Max we point out two works in which the performance of the Max 1100 model was compared with Nvidia and AMD GPUs. Article [175] is focused on supporting modern versions of OpenMP, which allow to work on GPUs. Here, using Max 1100 (with Intel oneAPI 2023) and A100-40GB (with Nvidia NVHPC 23.3), calculations were performed on the CFD-related mini-application LULESH using OpenMP parallelization, and it was claimed that the A100 was 34% better than Max 1100 (although the performance gain for the A100 depended on the size of the problem used in the calculation and varied from 15 to 42%). However, the large memory size of the Max 1100 made it possible to perform LULESH calculations for higher problem size, where the memory on the A100 was not enough.

In [176], the portability of SYCL to different hardware platforms was studied and performance data was obtained on a number of different applications using parallelization with SYCL (mainly in the area of CFD), executed on Max 1100 (with oneAPI 2023.1), A100-40GB (with CUDA 11.6) and MI250X (with ROCm 5.4.2; Only one of the two GCDs of the full GPU was used here, see Section 5.1.1 below for more information).

All selected applications are memory-bound (which limits their performance) and use structured and unstructured grid methods. On the BabelStream triad benchmark, the memory bandwidth obtained in [176] was 1310 GB/s for the A100, 1290 GB/s for the MI250X, and 803 GB/s for the Max 1100.

These applications used high-level, method-specific parallelization tools that generated various lower-level parallelization codes, including SYCL. In [176], different parallelization variants were used on each GPU for each application. In the optimal (for A100 performance) variants, this GPU was noticeably faster in all applications than the Max 1100 in any variants. And with a similar comparison of the performance of the Max 1100 with the MI250X, in some cases the Max 1100 was faster, in others the MI250X.

The information on PVC performance discussed below applies to the Max 1550 model. The report [120] provides data on acceleration in PVC (using SYCL tools in DPC++) relative to the A100 using CUDA (although data was also provided for the A100 with SYCL) — according to 8 benchmarks, of which the traditional HPC area includes SYCL HP Linpack (obviously, an HPL implementation using SYCL). Here an acceleration of 1.5 times was achieved. Among other benchmarks, the most famous is Google's hashing benchmark (hashtable), in which a maximum speedup of 2.5 times was achieved. In other benchmarks, the acceleration achieved was in the range of 1.4 to 1.8. It is clear that all these numbers require more detailed clarification. Interestingly, in some of these tests, using SYCL on the A100 gave about 10 percent better performance than using CUDA.

Other data on the performance of PVC are given in [116] — about a 12.8 times acceleration of calculations using the famous LAMMPS molecular dynamics software package on a dual-processor server with an Intel Xeon Max 9480 and six Data Center GPU Max — relative to a dual-processor server with a Xeon 8380. Considering, that 6 GPUs are used here, and the comparison is made with the Xeon 8380, which are inferior by the maximum achieved floating-point performance in the widespread SPEC CPU2017 benchmarks to the AMD EPYC 7763 processors in the speed and rates variants [177], it would be more interesting to compare performance with other modern GPUs.

An important publication that provides real data on the performance of the Data Center GPU Max 1550 compared to the A100-80GB is [178], which provides not only data on the achieved performance, but also on porting code from CUDA to SYCL. Here, using the FP32 format, molecular docking calculations (an extremely simplified specialized technique for calculating the orientation of closely located molecules, for example, ligands relative to a protein), which are often used for drug design, were performed.

For this purpose, the CUDA version of the well-known AutoDock-GPU program was transferred to SYCL using DPCT tools, followed by manual modification. When moving from calculations on a single Max 1550 stack to calculations on two stacks (simply changing the environment variable), the performance of various test calculations increased from 6% to 58%.

It is worth noting, by the way, the found ratio between the performance of calculations achieved on the A100 using CUDA and SYCL versions of AutoDock. Of the 10 calculations performed, the CUDA version was faster (from 1.24 to 3.64 times) in nine, and the SYCL version was faster in one.

Calculations using the SYCL version of AutoDock using both Max 1550 stacks were faster than on the A100 with the CUDA version in 7 out of 10 calculations (maximum 1.88 times faster); A100 was more successful in larger calculations (with more atoms or numbers of rotatable bonds). But all the achieved accelerations of the Max 1550 relative to the A100 were less than the ratios of the peak performance (for FP32) of these GPUs. As noted in [**178**], the calculations here are still preliminary, and improvements are expected in the AutoDock code.

In [**179**], data were obtained on the achieved performance of PVC using package linear algebra tools from the ginkgo library for an iterative solver of the Navier-Stokes equations (in the PeleLM lattice hydrodynamics application). When switching from a single Max 1550 stack to dual stacks, performance increases by 1.5 to 2 times, and the larger performance gains apply to larger task sizes.

In [**179**] also obtained similar performance data for A100-80GB and H100-PCIe. A comparison of these results shows that the Max 1550 using a single stack is on average 1.3 times faster than the H100, and 2.4 times faster when using two stacks.

It should be noted that this work is also largely preliminary. Thus, it lists the peak performance of the Max 1550 as 45.8 TFLOPS — less than the current "official" value of 52 TFLOPS (a lower clock speed was probably available). The calculations on the A100 and H100 used CUDA 11.8.0, while with the H100 typically used version 12 (for example, all Nvidia H100 MLPerf benchmarks used version 12.2, see Section 4.2.6).

Even these few publications show the high achievable performance of PVC. Peak performance data is insufficient to make good estimates of actual performance. The Max 1550's greater peak performance compared to that of the H100 does not necessarily translate into an advantage in actual application performance achieved. It should also be noted that the peak performance data given above in Table 7 do not include data on the peak performance of Nvidia and AMD GPUs for FP64 using tensor cores (with their use, the performance of top models of these GPUs is higher, see Table 20), since XMX in PVC this format is not support. It is clear that for PVC and oneAPI/DPC++ tools to be practically successful, there is still a lot of work to be done, which will require some time.

## 3.3. Summary for Intel PVC

Technologically complex PVC production may not produce a very high percentage of usable PVC chips (which could also result from a slowdown in the supply of PVC for Aurora, which is not on the June 2023 Top500 list) and contribute to an increase in cost. In fact, the same thing was stated earlier in [180].

Both the Intel Max 1550 and the AMD MI250X can be considered to consist of two logical GPUs, which raises the question of which comparisons should be made — for one logical GPU or for a full (physical) one, which is also associated with cost indicators that can vary greatly.

When compared with the whole AMD MI250X, the PVC is slightly ahead of the MI250X in terms of peak performance in conventional vector operations (see Table 7), but also has a slightly higher TDP. But the peak performance advantage of the new GPUs from Intel and AMD relative to the Nvidia H100 does not mean an advantage in real application performance.

A disadvantage of PVC for certain HPC applications can be considered the lack of support for the FP64 format in XMX. When using the Tensor Cores in the H100-SXM4 (or the equivalent in the MI250X), these GPUs significantly outperform PVC in terms of peak double precision performance (see also Table 20 below).

Taking into account the fact that Nvidia already offers the GH200 Grace Hopper superchip with H100 integration with ARM processors (see Section 4.2.3), and at the CES (Consumer Electronics Show) in early 2023, AMD announced the MI300 GPU with an integrated CPU (in the form 3D chiplet, assumption of readiness — for 2023), from the author's point of view, a significantly wider use of Intel GPUs in HPC may be realized somewhat later, after the implementation of the expected successful Intel technological processes (especially 18A), which are not expected in 2023. Intel recently abandoned the "second generation" PVC previously planned for 2023, codenamed Rialto Bridge [180], containing 160 $X^e$ cores, so we should rather expect the release of the Falcon Shores GPU or its subsequent XPU variant with integration of the GPU and x86 CPU [181] — success in technology probably determines everything.

Almost more important (compared to the competition between Intel and Nvidia for part of the GPU market — where AMD also participates) the author now seems to be the introduction of Intel-supported software for GPUs in the direction of SYCL — > DPC++ (and oneAPI), since this is not only allows us to move further away from a narrow focus on specific GPUs (Nvidia CUDA and partly AMD HIP), but it may also one day become effective for new multi-core server processors, the number of cores in which continues to grow. Perhaps the use of SYCL may immediately become the most attractive for modernizing C++ applications that have not yet been ported to the GPU.

However, one should not exaggerate the capabilities of the new SYCL standard. Thus, the appearance in CUDA tools for H100 of a new level in the hierarchy of the thread system — a cluster of thread blocks (see Section 4.2.5) — has no analogues in SYCL. It should also be kept in mind that SYCL does not solve all performance portability issues, and it may be necessary to use different algorithms for different hardware [176].

## 4. GPU from Nvidia: from A100 to H100

As noted above, the base GPUs here include the Nvidia V100 and A100, which are currently most widely used in the HPC and AI fields, including in supercomputers. Since the V100 became available back in 2017, this microarchitecture review only covers the A100; V100 specifications are shown in comparison tables only. In addition, the most important computational units and data types that first appeared in the A100 (which were not present in the V100) are mentioned. Detailed information about all the improvements to the various components of the A100 microarchitecture relative to the V100 is available in [73].

The most modern of the "basic" Nvidia GPUs, the A100, has been produced since 2020, and a detailed analysis of its microarchitecture is not carried out here, considering the relevant information is already quite well known. But in the microarchitecture of the H100, naturally, a lot coincides with the A100, and accordingly, after analyzing the A100, it will be convenient to talk mainly about improvements in the H100. Accordingly, only the figure of SM from the H100 is shown here (the A100 is only slightly different here — and it will simply be stated below what the differences are). The same applies to software — the review focuses in more detail on what is appropriate to use on a new generation GPUs.

Data on the achieved performance of the A100 in applications and benchmarks are also considered somewhat limited, since the main purpose of the review was the new generation GPUs, the performance of which is also considered with a comparison with respect to the V100 and A100.

## 4.1. GPU A100

### 4.1.1. *A100 microarchitecture*

In current CPUs generations, HPC performance gains come primarily from microarchitecture advancements rather than from ISA improvements. n current CPU generations, HPC performance gains come primarily from microarchitecture advancements rather than from ISA improvements. In modern Nvidia GPUs, microarchitecture data is also the most important, but to maximize performance, especially for AI tasks, new improvements/extensions to all levels of the CUDA API and low-level warp are especially important. CUDA tools can be classified as low-level because it is possible to work at a higher level — for example, use only already ready library functions, or work with directives. There is a lower level in relation to CUDA, where a virtual instruction system (ISA) for Nvidia GPUs is used — PTX (Parallel Thread eXecution) [182], and accordingly an assembler [183], but it is, naturally, used very rarely in programming. Strictly speaking, PTX is an intermediate level — it is then converted into binary code for a specific GPU model.

By improving the architecture, the review primarily refers to the microarchitecture. But there is a more general architecture — for different models of graphics processors, its microarchitectures retain some of its basic common features.

The A100 uses Ampere GA100 architecture. The A100 was the first GPU released with this architecture in 2020. Then the other, including less computationally powerful, GA100 models appeared. They use a smaller number of SMs and contain a smaller number of tensor cores, but more high-performance ones. But unlike the A100, their tensor cores do not support FP64 [184]. In addition, after the US imposed sanctions against China, Nvidia began to produce the A800 GPUs, which is not subject to these restrictions, with reduced computing capabilities compared to the A100. Information about these GPUs is available, for example, in the well-known Techpowerup database on GPUs produced in the world [185] (on the Nvidia website consumers were offered to use this database). The review below only considers the A100.

The most complete consideration of the A100 architecture is in [73], on which the subsequent consideration of its microarchitecture in this review is based. The consideration in [73] is partly integrated with SIMT-based parallelization (using CUDA) due to CUDA's deep coupling with Nvidia GPU hardware. This also applies to the hierarchy of different types of memory in GPU and CUDA (extremely important components for realizing high performance), and to the hierarchy of different levels of formation of large groups of parallel threads in CUDA. Table 1 lists all this briefly in terminological terms. However, not all memory types used in CUDA have

Table 10. Comparison of Nvidia V100, A100 SXM and H100 GPU specifications important for traditional HPC (data from [73, 78, 185])

| GPUs | V100 | A100 SXM | H100 PCIe | H100 SXM |
|---|---|---|---|---|
| Architecture | Volta | Ampere | Hopper | Hopper |
| TSMC Technology | 12 nm FFN | 7 nm N7 | 4N[1] | 4N[1] |
| Chip area, mm$^2$ | 815 | 826 | 814 | 814 |
| Number of transistors ($\times 10^9$) | 21.1 | 54.2 | 80 | 80 |
| TDP (W) | 300 | 400 | 350 | 700 |
| Form factor | SXM2 | SXM4 | PCIe-v5 | SXM5 |
| Number of SMs | 80 | 108 | 114 | 132 |
| Number of FP64 vector cores in SM | 32 | 32 | 64 | 64 |
| Number of FP32 vector cores in SM | 64 | 64 | 128 | 128 |
| Number of INT32 cores in SM | 64 | 64 | 64 | 64 |
| Number of tensor cores in SM | 8 | 4[2] | 4 | 4 |
| Boost Clock, GHz | 1.53 | 1.41 | 1.76[4] | 1.98[4] |
| Register file size in SM | 256 KB | 256 KB | 256 KB | 256 KB |
| Shared memory capacity in SM | Up to 96 KB[3] | Up to 164 KB[3] | Up to 228 KB[3] | Up to 228 KB[3] |
| L2 cache capacity | 6144 KB | 40960 KB | 50 MB | 50 MB |
| Memory type/capacity, GB | HBM2/16 or 32 | HBM2E/40 or 80 | HBM2E /80[5] | HBM3/80[5] |
| Memory bus width, bits | 4096 | 5120 | 5120 | 5120 |
| Memory Clock, MHz | 876 | 1215 or 1593 | 1593 | 1313 |
| Memory bandwidth, GB/s | 897 | 1555 or 2039 | 2039 | 1681 |

[1] modified for Nvidia;
[2] The number of matrix multiply-and-add operations per clock cycle in the A100 tensor cores is 4 times greater than that of the V100;
[3] Shared memory capacity in V100, A100 and H100 is configurable;
[4] for FP32 and FP64, for less precision the frequency is slightly lower;
[5] there is a model with HBM3/96 GB and a different GPU frequency.

a unique hardware equivalent. There is also the integration of various types of CUDA memory into one hardware type of memory A100. In addition, [73] sometimes refers to the V100 architecture discussed in the other Nvidia description.

The basic specifications of the A100 in comparison with the similar specifications of the V100 and H100 are shown in Table 10. In technological terms, due to the more modern 7nm TSMC technology in the A100, with almost the same chip area, the number of transistors has more than doubled compared to the V100, but the TDP has not increased as much strongly.

Table 10 shows data for two different A100 models with the SXM form factor, differing in memory capacity (40 or 80 GB). In addition, there are two other A100 models with PCIe interconnect, which can also have capacities of 40 or 80 GB. This somewhat limited representation of the A100 models in this table was chosen for two reasons. Firstly, for ease of reading and comparison with other GPU models from Nvidia in the format of this publication. And secondly, the peak performance discussed at this point in the review (see Table 12 below) was calculated for increased clock speeds, which are the same for all four different A100 models. In addition, further in Section 5, which describes the new generation GPUs from AMD (the comparison with which seems to be one of the most actual in this review), a more detailed comparison of all characteristics, including performance, with different A100 models is carried out (see Table 20 and Table 21).

For a starting understanding of the performance of the A100 (to begin with the peak) it is natural to base it on the SM execution units: the peak performance of the entire A100 is simply proportional to the number of SMs, which is indicated in Table 10. In the hierarchy from one SM to the full A100 there are 2 types of clusters: texture processing clusters (TPC, Texture Processing Clusters), containing by 2 SMs, and GPC clusters, containing by 8 TPCs. In the Ampere architecture (in GA100), it is permissible to have 8 GPCs and, accordingly, 128 SMs [**73**]. And the sense of GPC from the point of view of GPGPU (if we ignore textures): it is a group of SMs physically located close to each other, which gives locality of parallel processed data with access to them with higher bandwidth and lower latencies (so to speak, a kind of localised analogue of PIM).

The general construction of SMs, the many of which make up the computing power of the A100 (also both V100 and H100) can be seen in Figure 6. Although this is a figure of an SM from the H100 [**78**]), at the macro level of the figure, the differences in the A100 from the H100 are easily visible: the A100 does not have Tensor Memory Accelerator; L1 D-cache capacity is 192 KB versus 256 KB on H100; And in H100, accordingly, there is a new version of the tensor core. But the main thing is that in the H100 in each of the 4 SM sections (what is meant here by the SM section is clear from Figure 6) there are 2 times more vector FP64, FP32 and INT32 engines. In the A100, each section has 8 FP64 enginess, and 16 FP32 and INT32 engines.

Since SM runs warps consisting of 32 threads, each partition has a warp scheduler that produces 32 threads per clock and a dispatcher with the same "performance". This reveals the close intertwining of Nvidia GPU hardware and CUDA tools. A block of CUDA threads (containing warps) is assigned to one SM. And clarification of the operations of the warp scheduler is considered not in [**73**], but in the CUDA manual [**16**], where it is stated that SM statically distributes its warps between its schedulers. And then each scheduler issues one instruction for one of its assigned warps,

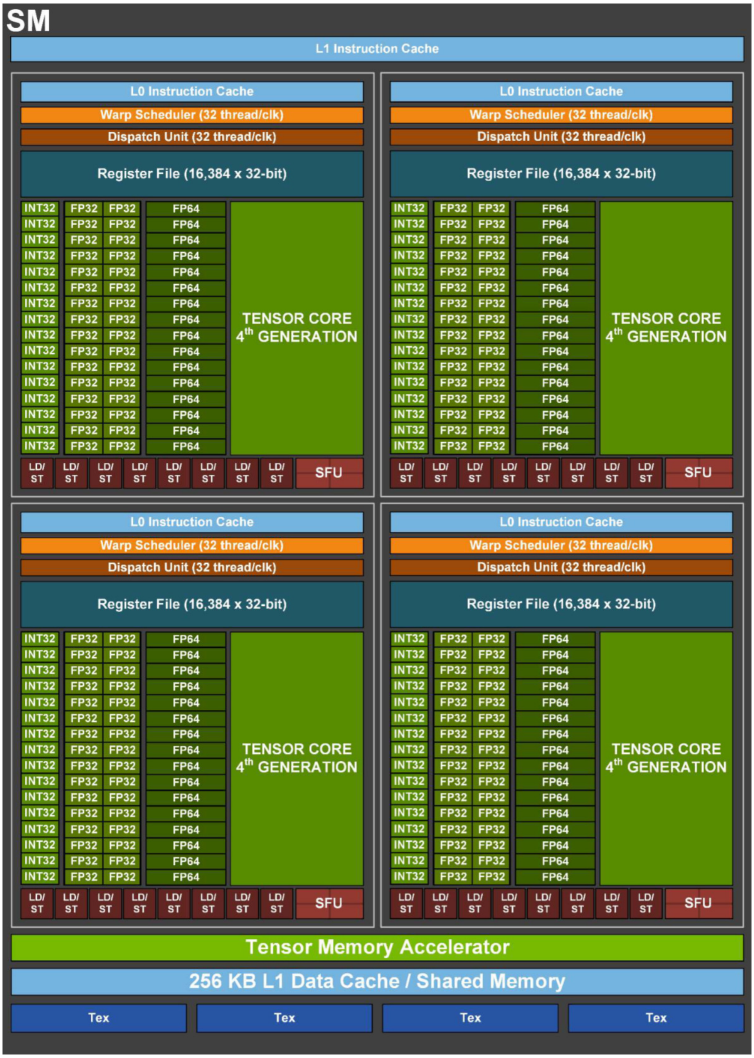

SM
L1 Instruction Cache
L0 Instruction Cache
Warp Scheduler (32 thread/clk)
Dispatch Unit (32 thread/clk)
Register File (16,384 x 32-bit)
INT32
FP32
FP64
TENSOR CORE 4th GENERATION
LD/ST
SFU
Tensor Memory Accelerator
256 KB L1 Data Cache / Shared Memory
Tex

FIGURE 6. General construction of SM in GH100 (Figure from [78])

which is ready to be executed, if there is one. The warp scheduler is used to ensure that the computing engines of the SM section are loaded: if a warp is waiting, for example, for memory operations to complete, then another warp can be executed. And the dispatcher redirects the selected commands to computing engines (they take data from registers).

Table 11. Technical characteristics of different versions of Compute Capability (CC) depending on Nvidia GPUs

| GPUs | V100 | A100 | H100 |
|---|---|---|---|
| Compute capability version | 7.0 | 8.0 | 9.0 |
| Threads per one warp | 32 | | |
| Maximum warps per one SM | 64 | | |
| Maximum thread blocks (CTA) per one SM | 32 | | |
| Maximum thread blocks/thread block clusters | No | | 16 |
| Maximum number of 32-bit registers per thread | 255 | | |
| Maximum number of registers per one SM | 65536 | | |
| Maximum number of registers per block | 65536 | | |
| Maximum number of threads per block | 1024 | | |
| Shared memory per one SM, KB, Up to | 96 | 164 | 228 |
| Number of FP32 cores per one SM | 64 | | 128 |
| Number of FP64 cores per one SM | 32 | | 64 |

The data in the table is taken from [78]

The actions of the warp scheduler may depend on the version of Compute Capability described in [16] for different GPU models; The above is true for V100, A100 and H100. These capabilities for these models are shown in Table 11.

Each of the 4 SM sections has a register file, and 8 load/store devices for memory access. Each SM section has its own L0 I-cache (the L1 I-cache is shared across the entire SM), but typically performance in HPC and AI does not require code to be specifically targeted to their use. When analyzing the performance of Nvidia GPUs, they often start from the SM level, since they also contain hardware components common to all 4 sections, including texture memory and a common L1 D-cache (SM can be considered some kind of analogue of the processor core in the CPU).

Each SM in the A100 and V100 has 32 vector CUDA cores for FP64, which allows the SM to perform 32 multiply-and-add operations per clock, resulting in 64 FLOPS per clock (for FP32 — everything is twice as much). Accordingly, the peak GPU performance when working with these vector cores is obtained by multiplying 64 FLOPS by the number of SMs (there are more of them in the A100 than in the V100) and the clock frequency, which gives higher performance for the A100 (see Table 10). But the main advances in performance are now focused primarily on AI and, accordingly, on the use of tensor cores, which can produce more results per clock cycle than conventional vector units — and correspondingly higher peak performance for various data formats.

TABLE 12. Peak performance of GPUs V100, A100, H100 (TFLOPS, for integer operations — TOPS)

| Data format | V100 | A100 | H100 PCIe | H100 SXM4 |
|---|---|---|---|---|
| FP16 | 31.4 | 78 | 204.9 | 267.6 |
| FP16[1] | 125 | 312/624 | 756/1513 | 989.4/1978.9 |
| BF16[1] | — | 312/624 | 756/1513 | 989.4/1978.9 |
| FP32 | 15.7 | 19.5 | 51.2 | 66.9 |
| TF32[1] | — | 156/312 | 378/756 | 494.7/989.4 |
| FP64 | 7.8 | 9.7 | 25.6 | 33.5 |
| FP64[1] | — | 19.5 | 51.2 | 66.9 |
| INT8[1] | — | 624/1248 | 1513/3026 | 1978.9/3957.8 |

[1] values when using tensor cores.
Following the slash numbers are the performances using sparsity when available.
Data from [73, 78, 185].

Tensor cores in A100 work with matrices of different possible sizes depending on the data formats used (a list of all possibilities is available in [16]). Traditional HPC requires working with FP64, which was first supported in tensor cores at the hardware level in the A100 — FP64 is what we will be talking about here. And data on peak performance for other data formats, primarily actual for AI, are shown in Table 12.

The A100 introduces a hardware implementation of the calculations for formula (1) for double precision, a new matrix instruction "multiply-and-add" (Double Precision MMA), which replaces the 8 similar DFMA instructions in the V100 and produces 128 FP64 results per clock — 2 times more than V100 [73]. For FP64 in A100, at the lowest warp level, the WMMA operation is applicable, with which you can work with matrices of dimensions $8 \times 4$ or $4 \times 8$ and accumulator matrices $8 \times 8$ (i.e. $M = N = 8$ and $K = 4$ for formula (1)) [16]. As noted in [89], WMMA operations can be performed on slightly larger matrices than supported by the tensor core hardware, and multiple tensor cores are used simultaneously when performing a WMMA operation.

The Nvidia shipped A100 has 108 SMs [73], so the peak performance achieved is $108 \times 128 \times \nu$, where $\nu$ is the clock speed. It follows that the peak performance information provided by Nvidia in [73] and used in Table 12 refers to the boost frequency. A100 users have the ability to control the clock frequency, which allows them to adjust the TDP and stabilize the measured performance — for example, in [186]) a frequency of 1005 MHz was used to study the performance achieved by GEMM.

Table 13. Base clock speeds of various A100 models and their memory

| Model GPU A100 | Base frequency | Memory frequency |
|---|---|---|
| A100-PCIe-40GB | 765 MHz | 1215 MHz |
| A100-SXM4-40GB | 1095 MHz | 1215 MHz |
| A100-PCIe-80GB | 1065 MHz | 1512 MHz |
| A100-SXM4-80GB | 1275 MHz | 1593 MHz |

The current Nsight Compute profiler [187] for the A100 defaults fix to a base clock speed, which gives repeatable results, but correspondingly lower peak performance than those listed in Table 12. Base frequencies depend on the specific A100 model used (these are the 4 most important for this review — they were usually used for calculations and benchmarks for HPC and AI), they differ in HBM2E size and form factor. Table 13 shows the values of their base frequencies from the database [185].

But in addition, the memory frequencies used and, accordingly, its bandwidth differ in different A100 models. Only two A100 models with 40 GB memory capacity have the same frequencies, and since the memory bus width (5120 bits) is the same for all A100 models, the theoretical bandwidth of the A100 models with 40 GB memory is the same (it can be calculated by multiplying the bus width by memory frequency). Table 13 also shows the memory frequencies of different A100 models (data from [185]).

But let's return to the tensor cores that determine the maximum peak performance of the A100. Even with such small hardware-supported matrices, they still have the ability to take into account fine-grained sparsity, which applies to formats with reduced precision relative to FP64 and gives there a two-fold increase in peak performance [94] (see Table 10). This sparsity was not supported in V100 and is mainly targeted at AI tasks (see [73] for details).

Since many HPC applications do not require matrix multiplication (the feasibility of supporting it in hardware is generally debated [90]), equally important to the double-precision performance of the A100 is the peak performance of the FP64 vector CUDA cores, which are not related to the tensor cores (see Figure 6).

One SM has 32 FP64 vector units (the corresponding FP32 vector units, of which there are twice as many in each SM, are traditionally called CUDA cores) [73]. For all 108 SMs, this gives a corresponding 6912 results per clock (thanks to the use of the multiply-and-add instruction), which when multiplied by the boosted frequency gives the A100 a peak performance of 9.7 TFLOPS for double precision — without the use of tensor cores (see Table 10).

In addition, each SM has 4 SUs, that speed up the calculations of transcendental functions. This can be practically important for a wide variety of applications, but SFUs have been available on Nvidia GPUs for a long time, even before the V100, but only for the FP32 format (see [188] for more details).

Since achievable GPU performance is so often dependent on memory bandwidth, we now need to look at the memory hierarchy in the A100. Because traditionally large numbers of threads running in parallel on a GPU require many registers, the A100, like older GPUs, has a 64 KB 32-bit register file in each of the 4 SM sections (see Figure 6). Each executing thread uses its own local registers from the corresponding file.

The next level in the memory hierarchy in the A100 is the L1 D-cache and shared memory with a total capacity of 192 KB. It provides high bandwidth and low latency; Real data are available for [189]. This memory is common to the entire SM (and accordingly to the block of threads in CUDA). In CUDA, it is possible to dynamically change the amount of shared memory.

The penultim level of the memory hierarchy on the path to the HBM2E global memory, the 40 MB L2 cache, is discussed in [73] in a common subsection with HBM2E. The spaces of these memory levels are common to all SMs and all applications running on the GPU. The L2 cache capacity has increased more than 6 times compared to the V100. This cache reads and writes to the HBM2E device's memory. Higher parallelization in the A100 required higher bandwidth per SM. To achieve this, the L2 cache split into partitions using a hierarchical crossbar structure, and each partition caches data nearer to the accessing SMs, reducing latency [94].

And the increased width of the HBM2 interface and memory frequency compared to the V100 gave an increase in bandwidth by approximately 1.7 times (see Table 10). To increase the effective DRAM bandwidth and L2 cache capacity, the A100 hardware features of sparse data compression provides up to 4× compression in DRAM, up to 2× compression in L2 cache for AI workloads [190].

CUDA uses its own types of memory, some of which actually reside together on the same type of hardware memory. Unlike shared memory, local memory is the private memory of each thread. In CUDA terminology, global and local memory areas (as opposed to global, the latter is cached [16]) are called device memory and are located in HBM2E. Constant memory (lives only while the application is running, this memory is read-only) is located in the device memory (constant memory is cached). As for the texture memory, it is located in each SM (see Figure 6) and cached in a special cache [73].

The A100 introduces MIG tools to address a possible data center problem of underutilization of A100 resources by certain applications where the use of A100 would become correspondingly ineffective. The A100 can function as up to 7 isolated GPU instances [73], reconfigurable on the fly to meet dynamically changing needs [94], i.e. the result is virtualized GPUs with smaller computing resources.

The new MIG feature provides improved client (including virtual machines and processes) isolation and QoS for multi-tenant and virtualized GPUs, which is especially useful for cloud service providers [73]. New fault handling technologies in the Nvidia Ampere architecture are very important to MIG to ensure proper isolation and security between clients using the same GPU.

As stated in [94], two types of MIG instances are supported. One type isolates computing resources but not the memory system, allowing the operating system to schedule processes with simplified administration. The other type further provides functional and performance isolation in the memory system. In this case, A100 assigns each MIG its own physical paths (including to the L2 cache and memory interface), providing additional security for cloud computing [94].

Using MIG allows you to expand the exeediency of using of the A100 to a wider area of work. For more information on setting up and using MIG, see [191].

Nvidia's A100-PCIe-80GB manual [192] describes software power management capabilities that allow to customize graphics card's power consumption or performance per watt. The A100 naturally has many more features that are important for achieving high performance, among which we should first of all mention support for asynchronous copying (at the ISA level) — it loads data directly from global memory into shared memory in SM, bypassing the register file, and can run in the background while the SM performs other calculations (this also reduces power consumption). For more information on this and other A100 features, see [73]; A brief discussion of asynchronous data transfers and computations is provided later in Section 4.2 about the H100, in which the corresponding capabilities have been significantly expanded.

A description of A100 interconnects using NVLink channels is provided below in Section 4.1.2, since it partly applies to hardware additional to the GPU — this also includes connections to the CPU via PCIe.

#### 4.1.2. *Interconnections for A100 and computing systems with A100: from servers to supercomputers*

NVLink interconnects for communication between multiple Nvidia GPUs in a server predate the V100. Initially, there was a problem that PCIe bandwidth was not high enough to connect the GPU to the CPU. A general overview of the interconnects used by various companies for GPUs is available in [**193**]. NVLink is a channel, and devices can have multiple channels and communicate through a mesh network.

Naturally, the second version (NVLink2) used in the V100 has a lot in common with NVLink3 in the A100. Various performance metrics for NVLink2 running with V100 are discussed in detail in [**194**]. NVLink2 (instead of the slower PCIe 3.0) is used to both communicate with IBM Power9 and provide cache coherence.

Like PCIe, NVLink2 is a packet bus, but NVLink2 is more efficient when transferring small packets — with 16 bytes of header, 256 payload bytes can be transferred. This interconnect uses a mesh topology for point-to-point connections, resulting in higher overall bandwidth compared to the tree topology in PCIe. Connections consist of multiple full-duplex links that exchange data at 25 GB/s in each direction. The device has up to six channels, and they can be combined into 3 channels with a total speed of 75 GB/s. Both PCIe and NVLink provide bidirectional transfers, but NVLink also has direct access to CPU page memory [**194**]. In [**194**] are given data on the achievable bandwidth and latency of NVLink2.

Using a switch to communicate between Nvidia GPUs not only improves performance scalability, but also provides an extremely important (especially for modern AI tasks) increase in the amount of available GPU memory. NVSwitch is a non-blocking switch (inernally — fully connected crossbar), has 18 NVLink ports and makes it possible to connect up to 16 V100 GPUs via NVLink2, providing each channel with a bidirectional bandwidth of 50 GB/s — and, accordingly, a total switch bandwidth of 900 GB/s [**195**]. For example, in implementations on motherboards containing 8 GPUs, each of them is connected to 6 switches on the board, which also have connections with switches on another board. GPU communications inside the board require one NVSwitch traversal, and with the GPU of the second board — two NWSwitch traversals. Cyclic Redundancy Coding (CRC) is used to ensure communication reliability on links, and ECC is used within switches (for example, for routing) (see [**195**]). for details). Bandwidth limitations on servers with V100 may occur more likely when transferring data over PCIe between the GPU and CPU.

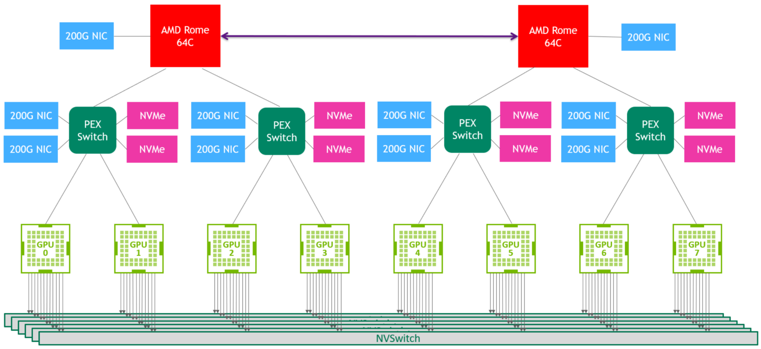


Figure 7. GPUs interconnect in the DGX A100 server (Figure from [73])

Multi-GPU servers with A100 already use NVLink3. Low-level details explaining why increased bandwidth and reduced latency relative to V100 with NVLink2 were obtained, are described in [94]. From the user's point of view, the main thing is that with no change in channel bandwidth, the number of channels has increased from 6 on the V100 to 12 on the A100.

In multi-GPU servers using NVLink, various specific topologies are accordingly possible [190]. For example, the AI-oriented DGX A100 server has 8 GPUs and 6 NVSwitch chips, and each GPU is connected to each NVSwitch using two NVLink3 links [196]. This is illustrated in Figure 7 (PCIe PEX switch is used to communicate with EPYC Rome) [73].

NVLink3 also provides improved error detection and recovery features [190]. Although, as stated above, the A100's single NVLink channel has the same throughput as the V100 — 25 GB/s in each direction — but it uses half as many signal pairs per channel compared to the V100. Doubling the number of channels in the A100 compared to the V100 gave a total throughput of 600 GB/s for the entire A100, compared to 300 GB/s for the V100.

In [3], on different servers with 4 V100 GPUs and on a DGX A100 server with 8 A100 and two 64-core AMD EPYC 7742 processors, the throughput of data transfers from the host to the device or vice versa was measured. For the A100 with PCIe-v4, it was about 25 GB/s (about three-quarters of the theoretical value of 32 GB/s for PCIe$\times$16), which corresponds to the authors' expectations [3]. This work also revealed a slight decrease in the bandwith of bidirectional data exchanges between CPUs with "remote" A100 compared to "local" ones (having a direct PCIe connection with the corresponding CPU), which was due to competition for work with PCIe.

It is clear that calculations with multi-GPU servers to achieve the expected high performance may probably require significant modernization of programs or applications that work with one GPU (and even perhaps more subtle optimization taking into account some “asymmetry” of different GPUs in the server). Examples include works [197, 198]. Such modernization also includes algorithms, and can cover the most basic things — for example, GEMM (see [199] and [200]).

It is clear that servers containing one or more A100 GPUs are being supplied by all leading server manufacturers. Multi-GPU servers are focused primarily on AI workloads, although HPC software is emerging in areas where GPUs have not been used much at all. For example, the QUICK application implements parallelization at the multi-GPU level, including for the most widely used quantum chemical method DFT in the Gaussian basis [198].

The DGX A100 is an AI-focused Nvidia famous “classic” server containing 8 A100 GPUs and 6 NVSwitches [201]. It uses the HGX A100 module for this, and gives a specific fixed SXM4 interconnect topology. The A100 has two variants with a memory size of 40 or 80 GB, respectively, there are two server options — DGX A100 320GB System and DGX A100 640GB System, with memory size of 1 and 2 TB per system, respectively. The DGX A100 has two 64-core EPYC 7742 Roma processors, which are connected to the A100 via PCI switches (with PCIe-4.0 $\times$16 buses). For communication between servers, Nvidia ConnectX-6 or ConnectX-7 adapters (Infiniband HDR /200 Gb/s Ethernet) are used [202].

The DGX A100 is a ready-to-use system supplied with an operating system (based on Ubuntu Linux) containing special management and monitoring tools. For details (including the implemented topology of the interconnects used), see [202].

In addition, Nvidia is reviving a class of workstations that has not been used for a long time — with the DGX Station A100 offering (for different types of DGX systems, see [203]).

Nvidia has gone even further in providing fast deployment computing systems and created the DGX SuperPOD platform, based on building blocks with racks containing of five DGX A100, with the delivery of cluster systems having from 4 to 28 racks (see, for example, [196]). This arrangement of building blocks was then modernized [204]. These AI-centric systems can be deployed in just a few weeks [196, 204]. For example, in the Top500, such a supercomputer Nvidia Selene based on DGX A100, installed in 2020, takes 9th place. Obviously, the DGX A100 and larger systems are aimed at simplifying work in their respective data centers.

Let's point out even more powerful supercomputers using the A100, which are included in the top ten Top500. The computing nodes of the Italian supercomputer Leonardo (installed in 2022), which ranks fourth in the Top500 [**205**], contain one 32-core Intel Xeon Platinum 8358/2.6 GHz processor (Ice Lake) and 4 GPU A100-SXM4-40GB, and for communication between nodes is used Infiniband HDR200 (Nvidia products with Dragonfly+ topology).

And the eighth place in the Top500 is occupied by the Perlmutter supercomputer installed in 2021 of the US National Energy Research Center (NERSC, and the operator is the American Lawrence Laboratory, LBNL, famous in the supercomputer world) [**206**]. Its main computing nodes use a 64-core EPYC 7763/2.45 GHz processor and 4 A100 GPUs (most nodes use A100-40GB, and another 256 have A100-80GB). Perlmutter uses the well-known HPE Cray EX235N supercomputer hardware and the corresponding HPE Slingshot 11 interconnect. Interestingly, these last two of the above-mentioned supercomputers, focused not only on solving AI problems, use 4 A100 per node.

### 4.1.3. *Nvidia SDK Tools for A100*

**General composition of SDK tools.** Like the A100 hardware discussed above, the SDK tools are also predecessors to the corresponding H100 tools. But SDKs are constantly evolving, and different versions of them can usually work for both base Nvidia GPUs and next-generation GPUs. After analyzing the A100 SDK here, in relation to the H100 SDK it is enough to point out the improvements.

Nvidia's SDKs have become well known due to Nvidia's clear dominance of the GPU market for many years. The basis for HPC is, of course, NVHPC tools — see the developer's website, [**207**]. There is also a complete list of components included in these software tools, most of the names of which (primarily mathematical libraries) already explain why they are needed. The ability to freely access NVHPC is a plus for Nvidia. At the time of writing this review, HPC SDK Version 23 was available [**208**].

NVHPC includes compilers supporting x86-64 (AMD and Intel), OpenPower and ARM platforms. They call the corresponding language compilers (with support for OpenMP and OpenACC tools for the CPU), assembler and link editor for target processors with command line parameters. These compilers are nvc (ISO C11), nvc++ (ISO C++17) and nvfortran. The latter (supports Fortran 2003 and a number of Fortran 2008 features) can use the parallelization features of Fortran, OpenMP and OpenACC to work with the GPU.

Finally, nvcc, a CUDA C/C++ compiler driver for Nvidia GPUs, uses GPU-specific options to generate optimized code for Nvidia GPUs and manage the host compiler. The latest version of CUDA at the time of writing was 12.2. nvcc hides the complex details of CUDA compilation from the developer, and redirects all non-CUDA compilation steps to the host compiler with special command line options [208]. The effective development of nvcc is facilitated by its basing on the famous LLVM compiler [209]. Further Section 4.2.5 about CUDA tools for the H100 describes the general design of different Nvidia compilers down to the runtime level, including new previously unused programming languages.

HPC SDK math libraries include, in particular, cuBLAS, cuSPARSE, cuRAND, and cuSOLVER (for solving linear algebra problems, including eigenvalue problems). cuSOLVERmp provides these functions when working with distributed memory in multi-node and multi-GPU systems. The cuFFTmp library complement the cuFFT library to perform similar advanced functions as in cuSOLVERmp.

The low-level cuTENSOR library is designed for linear algebra problems on tensor cores, and can be used not only for AI problems, but also in HPC areas. Two very important libraries provide communications. These are NCCL (Nvidia Collective Communications Library), which provides communication primitives for multi-node and multi-GPU systems with Nvidia GPUs, and NVSHMEM for PGAS parallelization using the OpenSHMEM standard. Here the memory can, in a sense, be integrated in a cluster with Nvidia GPUs in the nodes, using communications with NVLink, PCIe and Infiniband.

NVHPC includes a number of other tools, including the CUDA-GDB debugger and the Nsight Compute profiler (it is possible to use the NVTX profiling library to work with it) [208].

It should also be noted that traditional MPI parallelization tools on multi-node clusters containing GPUs can naturally also be used. But what may be important here is the use of such MPIs, which can use the long-standing hardware and software capabilities of Nvidia GPUs and CUDA tools — GPUDirect, which provide data transfer (RDMA) via PCIe bypassing the CPU — directly through the network card. Such capabilities have long been available, for example, in MVAPICH2 [210].

**CUDA Toolkit.** CUDA is the most famous and widespread API for working with GPUs, discussed in many publications. AMD has developed similar HIP tools for its GPUs [58]. Intel's OneAPI uses DPC++, but the overall parallelization design there also uses the `SIMT` model originally proposed by Nvidia, and oneAPI uses terms similar to those used in `CUDA` (see Table 1 above). The hardware of modern `GPU`s is closely connected with `SIMT`, so an ultra-brief explanation of the basics of `CUDA` is also necessary here (for a full description, see [**16**]).

Roughly speaking, in the `CUDA` programming model, sequential code is written, which is executed in parallel by many threads on the `GPU`, and each thread works with its own part of the common data, for which it receives its own identifier. Although a `CUDA` program running on the host can use not only C but also C++, the core software running on the device uses extensions relative to C.

In `CUDA`, the original task is divided into a set of independent subtasks, which are calculated on their own thread blocks. Each subtask has its own block of threads, and it is solved by all threads of this block. Threads can interact with each other only within the same block. A thread block is an array of threads that can be one-, two-, or three-dimensional.

Threads inside a block of threads use shared memory (ultra-fast — roughly speaking, at the cache level) and interact through barrier synchronization (when accessing this function, further work is impossible until all threads enter it). And global memory is used to exchange data between different blocks of threads.

The thread block is the central object for parallelization programming in `CUDA` (although with the advent of H100, a new level has appeared above the thread block in `CUDA`, the thread block cluster, this is unique to H100). The top level of the thread hierarchy — above the thread blocks — is the thread grid, a one-dimensional, two-dimensional or three-dimensional array of thread blocks.

All blocks of threads have the same dimensions and size (number of threads in a block). Thread grid size (number of thread blocks) and block size are built-in variables. Any block of threads in a grid has a block index in the grid. Each thread has an index specified by one, two or three non-negative integers.

For indexing threads and blocks, the software core uses, for the "most scalable" case, three-dimensional integer vectors — the built-in variables threadIdx and blockIdx.

Each block of threads is divided into warps, and all threads of a warp belong to one block. Only threads within the same warp are physically executing simultaneously. And threads at different warps can be at different stages of program execution. There is no need to exlicitly work at the warp level in a CUDA program. All Nvidia GPUs discussed in this review have warps of 32 threads. Working at the warp level may only be necessary to maximize the achievable performance in CUDA.

Nvidia CUDA is not limited to being based solely on C/C++. We will illustrate working with CUDA using the example of CUDA Fortran, which is attractive because Fortran remains popular in HPC (the ability to use Fortran to work with Nvidia GPUs is a significant advantage of the company). The newest (as of the June 2023 Top500 list) version of nvfortran was 23.3.

CUDA Fortran extensions allow you to write subroutines and functions in a Fortran program for execution on the graphical processor, including declaring variables allocated in GPU device memory and allocating dynamic memory in GPU device memory. Naturally, CUDA Fortran has support for copying data from host memory to GPU memory and vice versa, and calling GPU routines from the host. CUDA Fortran provides software access to tensor cores, cooperation with CUDA C, the use of asynchronous transfers between the host and the GPU (asynchronous transfer allows calculations to be performed on the device simultaneously with data transfer) and many other capabilities needed for modern GPUs [211].

Below is a simple example of how a program is built in CUDA Fortran (quoted from the Nvidia manual, [211]).

On the host

```
 1 program t1
 2 use cudafor
 3 use mytests
 4 integer, parameter:: n = 100
 5 integer, allocatable, device:: iarr(:)
 6 integer h(n)
 7 istat = cudaSetDevice(0)
 8 allocate(iarr(n))
 9 h = 0; iarr = h
10 call test1<<<1,n>>> (iarr)
11 h = iarr
12 print *,&
13 "Errors: ", count(h.ne.(/ (i,i=1,n) /))
14 deallocate(iarr)
15 end program t1
```

On the device

```
 1 module mytests
 2 contains
 3 attributes(global)  &
 4 subroutine test1( a )
 5 integer, device:: a(*)
 6 i = threadIdx%x
 7 a(i) = i
 8 return
 9 end subroutine test1
10 end module mytests
```

On the left is the code that runs on the host, and on the right is the code that runs on the device.

In the host code, the use of the |cudafor| module (line 2) provides interfaces to the CUDA host runtime library (in this program, to |cudaSet-Device()|, where the device number 0 is selected — this is the API call on line 7). Line 3 indicates the use of a module on the device (this module contains the test1 subroutine that is called). The 8th line of the program allocates the iarr array on the device, and the next line initializes data on both the host and the device [211].

The kernel running on the device is called on line 10; |«<1,n»>| here means that the kernel is executed by $n$ GPU threads (more generally, $n$ in this syntax indicates the number of threads in the block, and before the comma the number of thread blocks is indicated, which here is 1). On line 11, the results of the kernel execution are transferred to the host array, and on line 14, the CPU array is deallocated.

On the right side of the Fortran text running on the device, the prefix attributes(global) is used, which is an extension in the CUDA Fortran language. The global attribute means that the corresponding code runs on the device but is called from the host [211].

The 6th and 7th lines of code for the device are a replacement for the do loop in usual Fortran:

```
do i=1,n
a(i)=i
enddo
```

Since the test1 subroutine is executed on the GPU, it is executed in parallel by several threads on the GPU, each of which is identified by the built-in variable ThreadIdx (it is used as the index-of the array element).

Since the test1 subroutine is executed on the GPU, it is executed in parallel by several threads on the GPU, each of which is identified by the built-in variable ThreadIdx (it is used as the index-of the array element).

CUDA Fortran allows you to work in conjunction with other software. For example, it is possible to use OpenACC and CUDA Fortran together in one program; A program with OpenMP can use CUDA-managed memory [211].

Descriptions for CUDA Fortran are permanently available at the URL https://docs.nvidia.com/hpc-sdk/pdf/hpc*Vv*.pdf, where *Vv* are the all version digits of the SDK (e. g. *Vv*=233 for version 23.3 at the time of writing the review).

To conclude the brief information about CUDA Fortran, it should be noted the point of view of Nvidia employees, indicated in their famous book [212], that the use of CUDA Fortran will be aimed primarily at those programmers for whom it is important to ensure software portability, clarity and ease of maintainability of code when achieving acceptable performance. But the ease of writing code also has another important effect, increasing labor productivity (reducing program development time).

Naturally, other developers also have SDK-class software that can be used when working with Nvidia GPUs. For example, HPE Cray MPI [213] with CUDA support (this MPI correctly copies data from the device memory to the network card memory (and vice versa) by implicitly copying the data first to the host memory, and from there to the network card, or directly (bypassing the host memory) with support for GPUDirect RDMA [214], which is available in A100. MVAPICH2-GDR [215] can also interface with CUDA. AMD HIP tools, which can also work on Nvidia GPUs, were mentioned above. Such SDK tools that were not developed by Nvidia will not be discussed here, but they may be discussed in the sections of the review about new generation GPUs.

#### 4.1.4. *A100 Performance Data*

Although the review is aimed at new generation GPUs, it seems necessary to provide data on the achieved performance of the A100 and its acceleration relative to the V100. Information on the performance of the A100, which also provides a comparison with the new generation GPUs, is discussed further in the relevant sections about these GPUs, and may not be presented here.

Peak performance was discussed earlier, data from benchmarks and applications are discussed here. It must be kept in mind that deliveries of the A100 GPU from Nvidia, which began in 2020, were accompanied by such an active emergence of new A100 models, including due to the simultaneously awakening competition with the new generation GPUs from AMD emerging at that time, that the authors of some publications, where the latest A100 were used, the parameters of the model used were not always clearly indicated.

It is useful here to give a short introduction about the efficient use of tensor cores for dense matrix multiplication and the use of mixed precision. For working with AI, this is enough natural; For ordinary HPC applications FP64, that run efficiently on the GPU, the use of DGEMM is also often assumed. For a discussion of how actual is hardware support for DGEMM for mainstream HPCs, and whether lower mixed precision can be used, see [90].

HPC often requires DGEMM to work with large square matrices, and the A100 tensor core performs hardware operations on matrices of fixed small sizes (see above in Section 4.1.1). But there are many tensor cores in A100, and multiplication of large matrices boil down to multiplication of small submatrices. And programmers can simply use cublasDgemm, and for a large matrix size (compared to hardware-supported in tensor cores) $16384 \times 16384$, [216] indicates achieving 19.4 TFLOPS close to the peak performance for the A100 tensor cores. In [216], power consumption was also studied, and it was found that an increase in power does not always correlate with an increase in performance.

It should be noted that in [216] a matrix size multiple of $2^k$ was used. This is consistent with Nvidia's guide to working with matrices [217] on tensor cores (focused on AI areas) — it notes that performance is higher with matrix sizes that are multiples of 128 bytes (for FP64). In [217] describes how GEMM in cuBLAS is implemented by partitioning the output matrix into fragments that are assigned to blocks of threads, and discusses the optimal trade-off between the size of the submatrices to be multiplied and the requirement to utilize all GPU hardware resources through maximum parallelization. But these different proposed variants of GEMM are more important for not very large input matrices, and the results in [217] are demonstrated for the low precision typical for AI.

For the graphical processor Nvidia Titan RTX, where DGEMM was emulated using tensor cores due to their lack of hardware support for the FP64 format [218], a plot of the performance of DGEMM from cuBLAS versus the dimensions of square matrices shows certain jumps in the achieved performance. And for the A100, such jumps in GEMM performance when changing matrix dimensions when working with FP16 are demonstrated in [217]. Obviously, a similar dependence of the performance of A100 in DGEMM for the most relevant large square matrices for HPC also has this property, but the author is not aware of any articles demonstrating this.

So to work effectively with HPC on a GPU, not only sophisticated programming of possibly initially naturally sequential codes is required, but knowledge of the likely performance behavior at the BLAS level is also useful. In general, one gets the feeling that the use of tensor cores of these Nvidia GPUs for traditional HPC tasks in the FP64 format with conventional dense matrices remains to be studied — GPU hardware has developed too quickly, with an initial orientation not at all towards HPC.

A number of studies are devoted to the use of tensor cores and simply from the point of view of the applicability of mixed precision. Thus, in [219] various issues of such arithmetic were studied on V100 and A100, including from the point of view of rounding modes.

Work on software for matrix multiplication using tensor cores and their effective use continues. In [220], the possibility of using multiword arithmetic was explored — when matrices are represented as the unevaluated sum of two or more lower-precision matrices, and the product of the matrices is calculated by multiplying their lower-precision constituents. This was done, for example, on tensor cores of A100 and V100 for the input matrices **A** and **B** in formula (1) with FP16 numbers, and the accumulative matrices with FP32 numbers. Rounding errors were studied and the use of algorithms that reduce rounding errors was proposed. The purpose of this is to improve the trade-off between performance and precision [220].

For AI, it is relevant to work with matrices that have a specific sparsity. Data on such work with tensor cores for V100, A100 and H100 are available in [221].

The following is some of the information about the performance of the A100, which the author considers most relevant, especially for HPC tasks. The more high performance of the A100 relative to the V100 is obvious and confirmed in many articles. This is primarily due to FP64 support in tensor cores, higher peak performance, larger L2 cache size, and higher memory bandwidth in the A100. But this does not mean that the gain will be obtained in any code (especially not optimized), when using any version of CUDA, with any task dimensions, and so on. Thus, in [222] on some codes given by Nvidia as examples of using CUDA [223], the V100 outperformed the A100.

Well, as a kind of starting integral estimate of the acceleration of the A100 relative to the V100, we will use the data from the SPEC ACCEL v1.2 benchmark (using 19 different applications) [224], where the maximum results obtained by Lenovo for the A100-PCIe-40GB relative to the V100S-PCIe-32GB are better than approximately 1.7 times. Another integral assessment of the performance of the A100 is provided by data from the SPEChpc 2021 benchmarks, for which modern supercomputers with GPUs are also used [225]. However, the official results of these benchmarks [226] (as of June 14, 2023) do not contain data for computing systems with the same number of A100 and V100 GPUs. Comparative data on the performance of servers with multiple A100 is presented below in Section 4.2.6 on H100 performance.

**Microbenchmarks and low-level performance measurements, as close as possible to the A100 hardware.** Several publications can be roughly classified into this group of performance tests. In [227] stydied the performance impact of using asynchronous copy at the microbenchmark level and in the higher-level modernized integral Rodinia benchmarks suite (for non-AI applications at the University of Virginia).

In [79], microbenchmarks were used to measure the latency and bandwidth (performance of executing instructions) on tensor cores (at PTX level). In [228], the performance of kernels for memory-bound iterative methods for solving systems of linear equations was studied on the V100 and A100.

In [229], a low-level synthetic benchmark was developed and applied to determine the performance and power consumption of the DGX A100 server with 8 A100 GPUs and the DGX-2 server with 16 V100 GPUs using dynamic frequency and voltage scaling. To measure performance, it was used the Mandelbrot floating-point benchmark [230], implemented on CUDA PTX, which gives the closest approximation to peak performance due to high data parallelization, virtually no execution delays due to memory access and branches. These servers use 64-core EPYC 7742 due to their support for PCIe-4.0, which was not available in Intel Xeon. At optimal for power efficiency configuration, the A100 achieved power efficiency of 51 GFLOPS/W and 91 GFLOPS/W for the FP64 format when operating without and using tensor cores, respectively [229].

To ensure high application scalability, communication between GPUs in a multi-GPU server and between cluster nodes of such servers is important. A study [231] focused on Deep Learning Recommendation Models (DLRMs—widely used by Internet companies to predict what consumers might like when multiple options are available) obtained data on the bandwidth of such communications using NCCL and MPI for systems containing A100 and V100.

**Memory bandwidth benchmarks.** Data on traditional performance benchmarks can start with bandwidth benchmarks — it very often limits performance and is important for other benchmarks discussed below. Here we must keep in mind that due to the different programming models used, different numerical results may be obtained for the same benchmark. A notable example of this is *BabelStream* [232], which is a modification of the widely used STREAM benchmark for CPUs (see, for example, data for ARM processors [5]).

BabelStream does not take into account the transfer time between the host and the device, and it has implementations in almost all programming models used for GPUs. Compared to the classic standard STREAM benchmark [233] BabelStream has added a kernel with the scalar product of vectors, and some other modifications have been made [232].

In [234, 235], the obtained dandwidth data for all BabelStream kernels for different array sizes show a generally significant speedup of A100 relative to V100. For an 8.2 GB array, the V100 achieves 800 to 840 GB/s in these kernels, while the A100 achieves 1.33 to 1.4 TB/s. Although for small array sizes the A100 has lower bandwidth than the V100, for larger array sizes the A100 is about 1.7 times faster than the V100 for most kernels [234]. It should be borne in mind that these data were obtained shortly after the appearance of the A100 and obviously refer to the first model A100-40GB, and the CUDA 11 version was used.

Similar bandwidth data for all BabelStream kernels using different array sizes for the A100 and V100 are presented in [173].

In [42], data were obtained for the bandwidth (with the usual FP64 format) of all 5 BabelStream kernels: copy (a[i] = b[i]), mul (a[i] = b * c[i]), add (a[i] = b[i] + c[i]), triad (a[i] = b[i] + d * c[i]), dot sum (= sum + a[i] * b[i]) — and for each of them on 5 different programming models, for A100 (on a server with two 64-core EPYC 7H12 + 4 A100-40GB) and for V100 (on a server with two 20-core Xeon Gold 6230 + 4 V100 -32GB). One GPU was used in this benchmark. As a “unit of reference” that gives the maximum bandwidth of A100 and V100, the BabelStream results obtained with the CUDA version of this benchmark was taken.

Our analysis of data from [42] shows that in all BabelStream kernels using OpenMP and Kokkos on the A100 the "closing" to the CUDA values in percentage is more than on the V100 (with the exception of add and triad in Kokkos), and the achieved level of "close to the CUDA indicators" looks acceptable.

In many ways, BabelStream benchmark provide a convenient opportunity to compare the effectiveness of the implementation of different programming models of different GPUs. For example, in [236] a Fortran implementation of BabelStream was made using DO CONCURRENT tools from Fortran 2008 applicable to GPUs, which is in addition to other capabilities for working with GPUs in Fortran — using OpenMP, OpenACC or CUDA Fortran tools. Comparative performance data of different BabelStream implementations was obtained for A100-40GB and A100-80GB. Using NVHPC 22.7 for A100-80GB in C++ and Fortran, bandwidth of 1.7-1.8 TB/s was obtained in all 5 BabelStream kernels.

At the same time, the achieved bandwidth values on OpenMP and CUDA variants for C++ and Fortran practically coincided (the maximum difference is about one percent), with the exception of the dot kernel, where CUDA C++ was 13% slower than CUDA Fortran (after changing of the tuning parameter BabelStream, the C++ lag has decreased to one percent).

This review does not systematically examine the effectiveness of various GPU programming models, and these data were presented to illustrate the opinion of the authors [236] supported by the author of this review, about the lack of activity in the development of programming models using Fortran tools for heterogeneous architectures. The significantly higher results obtained in [236] for the A100 in BabelStream compared to [234, 235] are primarily associated with the use in [236] of a more modern A100 model (the A100-80GB has higher memory bandwidth; In [237] BabelStream triad kernel gave 1732 GB/s for A100-80GB versus 1399 GB/s for A100-40GB.

Achievable memory bandwidth data is no less important for AI than for HPC. In [231], a set of tests for DLRM tasks is proposed, which includes a memory bandwidth test specialized for DLRM, where it is evaluated when working with FP32 and FP16 formats. For the A100-40GB, a throughput of about 1.4 TB/s was achieved for FP32, for FP16 it was slightly lower. For V100 with FP32 more than 800 MB/s is achieved, and with FP16 — less than 800 MB/s [231].

Considering mixed-precision work in tensor cores, the corresponding memory bandwidth indicators are used to optimize the performance of the actively developed Ginkgo sparse linear algebra library portable to GPUs from different manufacturers [173]. The authors of [238] reports the performance of the A100 on a mixbench benchmark that uses kernels with mixed computational intensities (the ratio of floating point operations performed to bytes transferred) for different data formats and with different programming models [239], which gives and bandwidth estimates. This is actual mainly for AI and is not discussed here.

**Performance of Linear Algebra Codes (BLAS).** As with other GPU benchmarks, linear algebra tasks introduce a number of GPU-usable features that are different from traditional CPU use. This is mainly due to the frequent use of lower precision formats and support for operations with small matrices in tensor cores, which is primarily actual for AI. This led to BLAS with lower precision, as well as the development of batch linear algebra (solving many linear algebra tasks with small independent matrices; The batch size here is the number of matrices in it) [240].

Accordingly, linear algebra libraries specialized for GPUs have appeared (here we primarily mean not libraries from GPU development companies, but libraries focused on working with GPUs from different manufacturers) — and articles are appearing that study the performance of working with the tools of these libraries.

One thing to note here is the Ginkgo library for sparse linear algebra mentioned above; Data on the achieved performance on the A100 using it are available in [173, 234, 235, 237, 238]. For batch linear algebra, the well-known MAGMA library is applicable, the modules of which showed performance higher than the corresponding modules of the GPU manufacturers' libraries. Data on the performance of this library when running on the A100 are presented, for example, in [234] and [241]. For exascale computing, the libCEED library of the CEED Center (Center for Efficient Exascale Discretizations) of the US Department of Energy has appeared, where performance close to peak values is achieved on tensor cores of Nvidia GPUs [242], but the corresponding data in this work were obtained not for the A100, but for the V100.

It's natural to start analyzing performance for dense linear algebra problems with GEMM, since this is the only way to achieve maximum performance when working with A100 tensor cores. But due to the fundamental importance of GEMM and working with tensor cores, the general features of this for A100 have already been analyzed above at the beginning of Section 4.1.4 Publications that provide numerical estimates of the performance of GEMM with different data formats are reviewed here.

Benchmarks [222] reports the performance obtained for matrix multiplication of different data formats on the A100 and V100 for example CUDA kernels from Nvidia. There was no optimization goal here, small matrix dimensions atypical for traditional HPC problems were used, and this only illustrates the fact that you can use matrix multiplication code that will run faster on the V100 than on the A100.

Nvidia's tutorial on matrix multiplication for deep learning tasks [217] shows how GEMM's performance from FP16 on V100 increased dramatically when moving from cuBLAS v10 to cuBLAS v11, illustrating the rapid progress even in stable Nvidia software. This guide also shows typical performance dependences of the A100-SXM4-80GB for such GEMMs on matrix dimensions.

In [231] the benchmarks set for DLRM includes GEMM benchmark and uses GemmEx calls from cuBLAS. For V100 with FP32, this benchmark applies to working with FP32 vector cores, and for mixed precision with FP16 — with tensor cores. For the A100-40GB, the benchmark with tensor cores additionally includes work with TF32 and BF16. But the achieved performance values given in [231] refer to the batch version of GEMM (see Figure 8; BS in this figure is the package size), which is due to the small (indicated in the figure) sizes of different matrices. This figure shows that

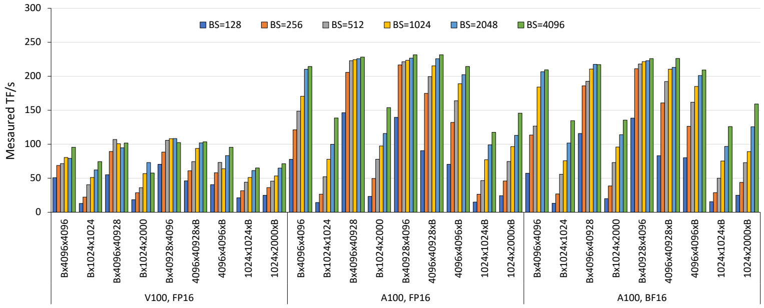


FIGURE 8. Performance of the GEMM batch implementation—with GemmEx calls from cuBLAS (Figure from [231])

the performance of the A100 with FP16/BF16 is several times greater than that of the V100 with FP16, which corresponds to the main focus of modern GPUs primarily on AI.

But the performance of batch variants of GEMM was also studied for the FP64 format. Data showing significant performance gains when running DGEMM in batch mode on a GPU are available in [242]. Batch variants of DGEMM for typical HPC tasks can be said to have not been used before—in the sense that relevant research has only begun to appear in the last ten years. Here it is worth noting the solution of the equation of hydrodynamics of compressible fluids with high-order finite elements [243] (see also [244, 245]).

In computational chemistry, the packaged DGEMM was used in the well-known CP2K software package for calculations using the quantum chemical DFT method with periodic boundary conditions in the basis of plane waves [246]. The effectiveness of batch matrix multiplication for calculating the matrix of pairwise distances used in molecular dynamics tasks was noted in [247].

The appearance of FP64 support by tensor cores in the A100 can significantly increase the relevance of batch DGEMM. But the use of these tools now largely relates to the development stage, figuring out where it can be used, which is also reflected in modern works. So, in [248] it was concluded that it is advisable to use batch DGEMM in the Nektar++ CFD application.

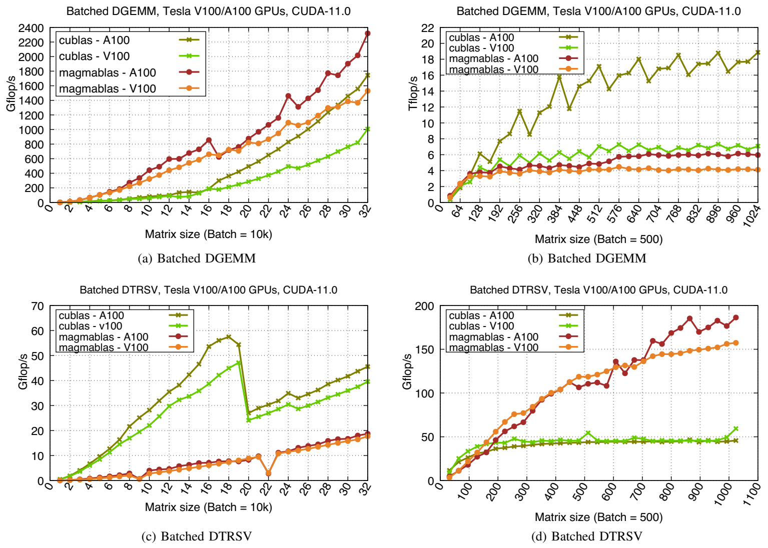


FIGURE 9. Performance of MAGMA and cuBLAS on DGEMM and DTRSV (Figure from [234])

In [249], calculations using the quantum Monte Carlo method were performed on nodes of the Vega supercomputer (at the Institute of Informatics, Slovenia), containing two CPUs (64-core EPYC 7H12) and four A100. Although the operation time of the batch GEMM was lower, in the Monte Carlo approach proposed in [249] using MPI parallelization of GEMM tasks, the overall performance of the entire simulation was much higher than that of the batch DGEMM. But in this work, tensor cores were not used.

Batch support for DGEMM is available in various libraries for the A100. In [234], the performance of batch DGEMM achieved on A100 was studied using MAGMA and cuBLAS (CUDA 11.0) tools. For small size workloads, MAGMA's batch DGEMM achieves 2.4/1.6 TFLOPS for A100/V100. This is 33/60% faster than cuBLAS, which is due to special optimization for small matrices in MAGMA. At the same time, the batch implementation of DGEMM in MAGMA did not use tensor cores (unlike cuBLAS). But on larger tasks, cuBLAS achieved 18/6 TFLOPS for the A100/V100 (for more details, see Figure 9, which also shows the performance of another BLAS routine, DTRSV); In general, on batch DGEMM, A100 is up to 1.5 times faster than V100 in MAGMA and up to 3 times in cuBLAS [234]. It can be seen that in the cuBLAS variant a good closing to peak tensor performance of the A100 for FP64 is achieved.

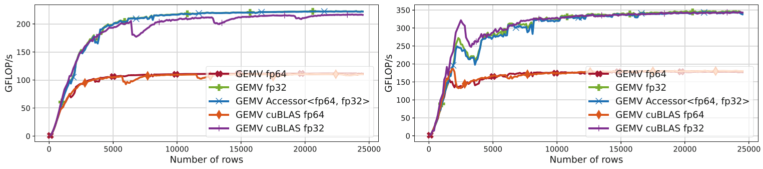


Figure 10. A100 performance for matrix vector multiplication (Figure from [238])

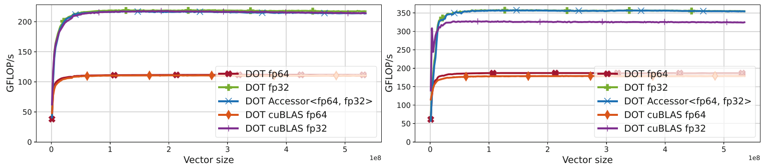


Figure 11. A100 performance with scalar product of vectors (Figure from [238])

In [234] also provides data on the performance of the A100 and V100 on MAGMA and cuBLAS implementations for batch BLAS DTRSV and LU and QR factorization (in LU and QR on the A100, MAGMA outperformed cuBLAS in all dimensions, and in DTRSV only in the middle ones).

To complete the discussion here of performance in batch linear algebra tasks on the A100, we should provide performance data for the QR factorization of dense matrices used in FP64 and FP32 formats [241]. The highest performance data, strongly superior to all alternatives, was achieved here using the MAGMA library. The asymptotic performance achieved (more than 2.6 TFLOPS for square FP64 matrices) is far from the theoretical peak values because the matrices for batch GEMM are not large enough.

From other publications regarding BLAS on A100, it is necessary to note the data [238] for matrix vector multiplication (GEMV kernel), which may be even more common (than GEMM) in HPC and for the scalar product of vectors (DOT kernel) — see Figure 10 and Figure 11. Here, the achieved performance for FP64 and FP32 using special memory access tools for different data formats of the Ginkgo library (marked as accessor in the figures) for A100 and V100 is compared with cuBLAS data. Here you can see how significantly the A100 outperforms the V100 in terms of performance across different BLAS implementations. Similar results were obtained in [238] for another matrix vector multiplication module from BLAS, TRSV.

As for working with sparse matrices, there are a number of publications for the A100 with data on the performance of sparse matrix vector multiplication (SpMV in BLAS), which is actual for various applications. [173, 234, 235] and [238] provide performance data on the A100 and V100 for SpMV in FP64 format. But the result here depends on the chosen sparse representation format, and on the specific matrices selected in the test, and, naturally, on the library used (cuSPARSE and Ginkgo were used here), and at the "macro level" it is less informative. Let us only note the maximum achieved performance of 220 GFLOPS and 135 GFLOPS on the A100 and V100, respectively [173], which is close to the upper limits was expected by article authors (230 GFLOPS and 140 GFLOPS, respectively—probably in roofline modeling).

Since choosing the optimal sparse matrix storage formats is also a non-trivial problem, machine learning was proposed to solve it on the A100 and V100 in [250].

**The fast ourier transform (FFT).** Important data on the performance of the A100 is provided by the achieved FFT performance, which is actual for various tasks, both HPC and AI. [251] demonstrates the performance of A100 using the developed SYCL FFT library compared to cuFFT. Although SYCL FFT is clearly inferior to cuFFT in terms of performance, the goal of the SYCL FFT developers is portability to GPUs from different manufacturers. At the same time, the lag of SYCL FFT is largely due to big overhead of the SYCL runtime (especially kernel dispatch), which is less important for large-scale problem.

Another FFT library aimed at working with GPUs from various developers is the open source VkFFT library, whose performance data on the A100-40GB is available in [252]. VkFFT supports double, single, and half precision data, and the performance achieved on the A100 is typically greater than with cuFFT [252].

In [253] developed the tcFFT library for one- and two-dimensional FFT using tensor cores. When running on FP16, tcFFT outperforms cuFFT (CUDA 11.0 was used) in 1D and 2D FFT on both the A100 and V100 (with the exception of low-dimensional 1D FFT). A significant superiority in performance in A100 relative to V100 for one- and two-dimensional FFT is shown; These data are illustrated in Figure 12 for a 2D FFT [253].

**Computational chemistry:**

(A) *Molecular dynamics.* Molecular dynamics problems began to act as classic problems, one of the first demonstrated examples of GPU

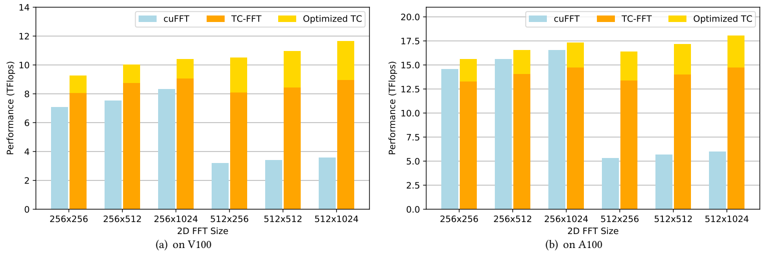


Figure 12. Performance of cuFFT and tcFFT on V100 and A100 (Figure from [253])

performance growth by their developers. Thus, in [190] an increase in performance in well-known molecular dynamics applications AMBER, GROMACS, NAMD and LAMMPS is indicated by 1.5-1.9 times on the A100 compared to the V100 (1.9 times applies to LAMMPS).

A detailed and in-depth study on the performance of LAMMPS in calculations of various molecular systems using different potentials and different GPUs, including the A100, is [254]. Here was used a cluster of multi-GPU servers (HPE/Cray EX on the Airforce Weather computing system, AFW HPC11). The A100-based nodes used one 64-core EPYC 7713 and four A100-40GB. LAMMPS uses Kokkos software to support code portability, and the paper aims to achieve the optimal choice of their parameters, and only the corresponding relative performance data is provided.

(B) *Molecular docking.* In [42], data were obtained on the performance of the miniBUDE mini-application [255] based on the BUDE (Bristol University Docking Engine) application targeted towards molecular docking for potential drug discovery tasks (prediction of the favored orientation of a ligand to protein at the form their stable complex. There, the change in the Gibbs free energy for the ligand-protein bond is considered extremely simple, empirically. BUDE was originally GPU-focused, and miniBUDE effectively ports to a wide variety of SDK for GPUs. The calculations here use FP32 and the A100 is about 30% faster than the V100.

Another molecular docking program, AutoDock, was upgraded to AutoDock-GPU in [256] to run on Nvidia GPUs. There, the A100 was found to achieve higher performance when using CUDA than when working with SYCL.

(C) *Quantum chemistry.* In [198] a radical modernization by the authors of the open source software package for computational chemistry (QUICK) is described, which makes it possible to carry out calculations using quantum chemical methods HF and DFT (the most common in the world today) in a Gaussian basis sets on multi-GPU servers, including c A100. A feature of calculations of all modern quantum chemical methods in such a basis set is the need to calculate $O(N^4)$ integrals (N-dimension of the basis), which cannot be reduced to widespread mathematical methods.

These calculations (as well as the corresponding gradients) require major calculation time using the HF or DFT method. For parallelization, CUDA C and OpenMPI tools were used here. Each of the above integrals from contracted basis functions is calculated by one thread at the PTX level, which made it possible to achieve high performance for such a complex task. Parallelization in QUICK is possible from the level of a single GPU to a cluster of multi-GPU servers. All calculations are performed in the FP64 format required for quantum chemistry; Tensor cores are not used due to the lack of need for matrix multiplication in these methods.

The presented results of run times for various molecules containing from several tens to several hundred atoms in an Infiniband HDR cluster with DGX nodes (4 V100-SXM2 with two Xeon Gold 6248 per node) showed high parallel efficiency and almost linear performance scalability from 1 to 16 GPUs. On a DGX A100 server with 8 A100, parallel efficiency is slightly lower than with V100 (V100 has less SMs), but the calculation time is much lower [**198**]. QUICK is developing rapidly [**257–259**], and the achieved very high acceleration in DFT calculations relative to conventional dual-processor x86-64 servers makes it attractive for fairly large-scale calculations.

An alternative approach to providing highly efficient Gaussian DFT calculations using GPUs was proposed in [**260**] and was implemented on the A100. In [**260**] new algorithms were developed for calculating the Coulomb and exchange contributions to the matrix of the one-electron DFT Hamiltonian, and a program was created that generates CUDA codes (the kernels are different for different angular momenta). On molecular systems ranging in size from a few hundred to over a thousand atoms, they achieved fairly good performance scaling using up to 128 A100 GPUs on the Perlmutter supercomputer.

Another demonstration of the ability to achieve a high level of parallelization scaling in a cluster of multi-GPU servers with A100 on quantum chemistry problems is work [**261**], where calculations were performed at the Samsung Electronics supercomputer center (on the SSC-21 supercomputer, ranked 20th in the Top500), the nodes of which used HPE servers with two 32-core EPYC 7543/2.8 GHz, 1 TB memory and 8 A100-80GB (nodes connected via Infiniband HDR200).

Here, using the TDDFT method (extended in comparison with DFT, with time dependence for calculating exited states), using Gaussian basis functions, calculations of a protein molecule containing over 40 thousand atoms were performed using up to 256 A100. The parallelization program used OpenMPI, OpenMP and CUDA Fortran tools (from NVHPC SDK 22.3). One MPI rank per node was used, where the processors (they are connected to the A100 via PCIe) generate a number of OpenMP threads equal to the number of GPUs in the node, and NVLink is used to communicate between the A100. The resulting acceleration does not deviate much from linear scaling up to 256 A100; Even with 256 GPUs, the parallelization efficiency exceeds 80% [261].

Another work carried out within the framework of the ECP project [262] presents the calculation times for the exchange-correlation component of the one-electron Hamiltonian of the quantum chemical DFT method using Gaussian basis functions using the new NWChemEx software package (developed on the basis of the well-known NWChem software package). The corresponding part of the code was done on CUDA. In the calculation of a protein (ubiquitin, more than a thousand atoms), the dependence of the calculation time on the number of GPUs used in parallelization on the Perlmutter (with 4 A100 in each node) and Summit (with 6 V100 in each node) supercomputers was studied. To optimize the calculation, code was developed at the PTX level, and NCCL/NVSHMEM tools were used to optimize the use of distributed memory. The calculation time for all program components decreases almost linearly as the number of GPUs increases to 32, and the deteriorating deviations from linearity when using V100 are slightly larger. And these times themselves on one A100 are one and a half to two times less than on the V100.

**Computational fluid dynamics (CFD).** Data on the performance of CFD applications are as "typical" in reports from GPU manufacturers as data on molecular dynamics, and accordingly, many publications with such data for the A100 have already appeared.

Thus, in [263], the calculation using the OpenCFD-SCU program on the A100 required 3.036 seconds per calculation step compared to 4.702 seconds on the V100.

In [264] provides performance data on the well-known Nek5000/RS code using the specialized OCCA (Open Library Concurrent Compute Abstraction) library for different types of GPUs. The kernels on the A100 there support 2.1–2.2 TFLOPS (FP64) for the Poisson operator and 3.1–3.8 TFLOPS (FP64) for the advection operator (higher numbers in ranges refer to higher dimensions). In the pressure preconditioner, the forward Poisson operator at coarser multigrid levels realizes 2.5–3.9 TFLOPS (FP32), and the Schwarz smoother supports 2.5–5.1 TFLOPS (FP32). The comparable values on the V100 are one and a half times lower.

Similar data on the higher performance of A100 relative to V100 by 1.55 times when running NekRS are available in [265].

In [190], an increase in productivity of 1.7 times is indicated on the A100 relative to the V100 of the famous NASA software package, FUN3D [266]. It is clear that this requires additional information about specifications of calculations.

FUN3D is a software package for solving CFD problems with an unstructured grid, originally written in Fortran, part of the source code of which was transferred to C++ CUDA in the form of kernels of the FLUDA library, and even a mini-application was developed. Calculations here are performed in FP64 format for most variables, and with mixed precision FP32/FP64 for linear algebra problems [267].

At the 2023 AIAA Science and Technology orum, there were a number of presentations that provided data on the A100's performance in various CFD tasks, including in comparison to the V100. In [268], it was found that three quarters of the total execution time of FUN3D using the A100-SXM-40GB on a 3.7 million point grid was used by memory-bound kernels. Achievable performance correlates with bandwidth, and calculation on the A100 by the NASA common research model (CRM) in transonic conditions using the Reynolds-averaged Navier–Stokes equations was 1.67 times faster than the V100-SXM-16GB calculation.

In [269], in an aerodynamic analysis for electric VTOL aircraft using a large eddy simulation program on a system with 8 A100, the calculation time using FP32 was 1.4 hours versus 2.9 hours on a system with 8 V100.

In [270], large eddy simulations of flow instability were performed on multi-GPU systems with 4 V100 and 8 A100. Performance vs. number of GPUs data for A100 and V100 systems shows that with the same number of GPUs (up to 4), A100 performance is several times higher when working with both FP64 and FP32, and good performance scaling up to 8 A100 is achieved.

Other CFD performance data for A100 and V100 servers at this forum is presented in [271], and in [272] performance data is obtained for systems with 4 A100 or V100 GPUs.

Report [273] presents data on the performance of solving CFD problems using the lattice Boltzmann method on the A100 and V100. The work [274] implements the lattice Boltzmann methods improved by the authors, compares the performance on the A100-40GB and V100 using FP32 and FP64 and, naturally, shows a significant increase in the performance of the A100 compared to the V100.

In [275] provides data on the performance achieved within the URANOS program when using the A100 or V100. URANOS is designed for compressible flow solver for high-fidelity modeling of compressible wall flows (solving the system of Navier-Stokes equations in a three-dimensional Cartesian system from low to high Mach and Reynolds numbers), is implemented in Fortran 90 and parallelized using OpenACC and MPI. The calculations were performed on two different Italian supercomputers with V100 in nodes, and on DGX-A100, where maximum performance was obtained. In all computing systems, maximum performance is obtained when using not the standard MPI implementation, but the supporting GPU from Nvidia, which uses direct data exchange between devices without accessing the host.

And in [276], for the tasks of accurate modeling of high-speed flows in various solution schemes, the efficiency of different Nvidia GPUs, including the A100 and V100, was compared using not only the FP64 format as the main one, but also FP32. Simulating supersonic jet noise (13 million cells, 400 000 iterations) on a single A100 using CUDA gave an run time of 34.5 hours. We calculated the ratios of various calculation times on A100 and V100 given in [276], and obtained a range of accelerations of A100 relative to V100 from 1.4 to 1.9 times.

**Artificial intelligence.** Regarding the A100's performance for AI problems, only two more important data sources (from the author's point of view) are considered here. Firstly, this is, of course, data on the performance of machine learning — MLPerf training benchmarks [277]; At the time of writing the review, the data was current for version 2.1 [278]. The more comparable "closed" section of these benchmarks is discussed below. Within it, possible data is divided into available cloud, available on-premise and preliminary. Table 14 shows locally available performance data for servers running the A100According to the rules, results can be provided for individual benchmarks from the general list. But in our review, the goal was maximum comparison, and the individual maximum performance values (obtained by different companies and achieved for individual of the benchmarks), which were not accompanied by many corresponding data for other benchmarks, are not shown in the table.

Table 14. Machine learning performance in MLPerf Training v2.1 benchmarks [278]. Data: Available on-premise

| Task | Type and number of GPUs | Calculation time, minutes | Submitter |
|---|---|---|---|
| Image classification | A100-SXM-80GB, 4 | 54.956 | ASUSTek[1] |
| | A100-PCIe-80GB, 8 | 30.756 | ASUSTek[2] |
| | A100-SXM-80GB, 4 | 54.231 | Dell[3] |
| Image segmentation (medical) | A100-SXM-80GB, 4 | 49.446 | ASUSTek[1] |
| | A100-PCIe-80GB, 8 | 25.855 | ASUSTek[2] |
| | A100-SXM-80GB, 4 | 47.253 | Dell[3] |
| Object detection, light-weight | A100-SXM-80GB, 4 | 161.699 | ASUSTek[1] |
| | A100-PCIe-80GB, 8 | 89.137 | ASUSTek[2] |
| | A100-SXM-80GB, 4 | 222.199 | Dell[3] |
| Object detection, heavy-weight | A100-SXM-80GB, 4 | 78.535 | ASUSTek[1] |
| | A100-PCIe-80GB, 8 | 43.081 | ASUSTek[2] |
| | A100-SXM-80GB, 4 | 83.712 | Dell[3] |
| Speech recognition | A100-SXM-80GB, 4 | 59.989 | ASUSTek[1] |
| | A100-PCIe-80GB, 8 | 32.534 | ASUSTek[2] |
| | A100-SXM-80GB, 4 | 55.086 | Dell[3] |
| Natural languages processing | A100-SXM-80GB, 4 | 33.431 | ASUSTek[1] |
| | A100-PCIe-80GB, 8 | 24.186 | ASUSTek[2] |
| | A100-SXM-80GB, 4 | 32.792 | Dell[3] |
| Recommendation | A100-SXM-80GB, 4 | 3.147 | ASUSTek[1] |
| | A100-SXM-80GB, 4 | 4.309 | Dell[3] |
| Reinforcement Learning | A100-PCIe-80GB, 8 | 161.647 | ASUSTek[2] |

[1] with EPYC 7773X, GPU TDP 400 W;
[2] with 2×EPYC 7763, GPU TDP 300 W;
[3] with 2×EPYC 7763, GPU TDP 500 W;

The table contains only data whose submitter provided results for 7 benchmarks out of 8 available in MLPerf training 2.1. There are no V100 performance data for version 2.1, but the strong acceleration of A100 relative to V100 can be indirectly estimated from MLPerf training HPC 2.0 data [279]. The resulting benchmark performance may depend not only on the GPU characteristics itself, but also on the specific computing system and software used.

This is not discussed further here, but below, in Section 4.2.6, we present data on the performance of the A100 in the new versions of the MLPerf benchmarks — training v.3.0 and another set of benchmarks for machine learning inference, inference datacenter v.3.1. There, the performance of the A100 is compared with the H100 (this data became available after the practical completion of the review).

Another publication with important information on A100 performance metrics for various AI domains [**231**] is narrowly focused on DLRM. It has been noted that these models have atypical requirements compared to other types of deep learning models. In [**231**], using A100 and V100, various performance data were obtained, including on a cluster of GPU servers, which is actual for large, highly scalable data centers, and recommendations were made on possible improvements to DLRMs.

All of the above performance data naturally and unambiguously indicates the advantage of the A100 over the V100, including in performance, which should be most pronounced in HPC applications that require FP64 matrix multiplication and/or large memory capacity. The V100 lag is smaller in HPC applications with FP64 that do not use matrix multiplication in tensor cores: both GPUs have the same number of FP64 vector cores per one SM.

## 4.2. New Nvidia H100 GPUs

### 4.2.1. *H100 — microarchitectural implementations of Hopper*

The main metrics that characterize the H100 versus the A100 are shown in Table 10, and Figure 6 illustrates the overall SM structure in the H100, providing a framework for understanding the H100 architecture and scaling performance with the number of SMs. In addition to the H100 models discussed in this review, Nvidia began producing H800 GPUs, which also have Hopper architecture and are not subject to US sanctions against China due to the reduced performance level in the H800 to an acceptable level. These GPUs are not covered in this review; information about them is available, for example, in [**185**].

All information provided later in this section about the Hopper microarchitecture and its implementation in the H100 is based on a general description by Nvidia [**78**], and additionally, if more detailed information is needed, references are provided to relevant sources.

Thanks to the above discussion of the A100 and the Ampere architecture, the H100 analysis can only focus on the improvements in the H100 relative to the A100, since the overall build of the H100 and A100, as well as their SMs, are roughly the same. The general hierarchy of GPU construction from the SM level to the full GPU level is no different for the H100 and A100, and the terminology used does not change: from two SMs a TPC cluster is formed, and from a set of TPCs GPC clusters are formed, of which there are 8 on the GPU. But different H100 GPUs in this hierarchies differ in quantitative indicators.

Like the A100, the H100 has the highest available metrics supported by the Hopper architecture (these are specified for the GH100). In fact, Nvidia

Table 15. Macro-level characteristics of GH100 and H100 models

| GPUs | GH100 | H100-SXM5 | H100-PCIe |
|---|---|---|---|
| Number of SMs | 144 | 132 | 114 |
| Number of TPCs | 72 | 66 | 57 |
| Number of GPCs | 8 | 8 | 8 |
| L2 cache | 60 MB | 50 MB | 50 MB |
| Number of HBM stacks | 6 (HBM2E or HBM3) | 5 HBM3 | 5 HBM2E |

offers two models (with different form factors) that implement the Hopper architecture, and their quantitative specifications in this hierarchy are shown in Table 15.

The GH100 has 9 TPCs in each GPC, but this is not required in real H100 models. The GH100, H100-SXM5 and H100-PCIe each use 10 memory controllers (512 bits each), giving a total memory width of 5120 bits (see Table 10). The most important thing is that H100 models with different form factors differ significantly in performance simply due to the different number of SMs, as well as due to different memory. This can be seen in more detail not in Table 15, but in Tables 10 and 12. Peak performance in the H100 can be calculated in the same way as was done above for the A100 — the number of SMs has simply increased in the H100, and every of 4 partitions of SMs in the H100 has twice the number of FP64, FP32 and INT32 vector devices than in A100 (see Figure 6).

Compared to the A100, in addition to the quantitative increase in the H100 in the number of available SMs and the number of vector cores contained in each SM, and at the general level of the H100 — the size of the L2 cache and other specifications, a number of very important improvements in the H100 hardware are closely related to CUDA and will be discussed further in the analysis of CUDA extensions for H100. This applies to both asynchronous execution and a new module, the Tensor Memory Accelerator (TMA), for efficiently transferring large blocks of data between global and shared memory (see Figure 6).

Finally, the H100 introduces or improves upon the A100 with a very large number of new hardware enhancements that are beyond the scope of this review in the narrow sense of its HPC focus and with limited consideration of AI. Even listing new such tools can take more than one indent — these are numerous improvements in MIG technology (especially actual for working with cloud technology), security tools, and so on. New instructions have been added to ISA, for example, for current genomics problems — this has a very important, but narrow meaning. A lot of what has been added relates to video and image processing and AI. Of the latter, we note only the appearance of hardware for transformer models used for NLP. As with the A100 above, interconnect hardware is covered in a separate section on H100 servers and computing systems.

Table 16. Nvidia GPUs Interconnect Specifications according to [78, 280]

| GPUs | V100 | A100 | H100 |
|---|---|---|---|
| Number of lines (differential pairs) per NVLink port | 8 | 4 | 2 |
| Single/bidirectional bandwidth of NVLink link, GB/s | 25/50 | 25/50 | 25/50 |
| Number of NVLink links per GPU | 6 | 12 | 18 |
| Total bandwidth of NVLink links | 300 | 600 | 900 |
| Total aggregate NVSwitch bandwidth, TB/s | 2.4 | 4.8 | 7.2 |

### 4.2.2. *Computing systems with H100*

This section describes Nvidia's H100-based computing systems (from servers to supercomputers) using x86-64 server processors, as well as the H100 interconnects and Nvidia's Grace ARM processors that will be used in H100-based servers.

**NVLink network for H100.** To scale performance with the growing number of GPUs used in servers (which has become a current characteristic trend) and quickly exchange GPU data with the CPU and with other GPUs, interconnects are critical. From the author's point of view, here Nvidia managed to achieve a very striking breakthrough (see Table 16).

NVLink4 in H100, as a high-speed, low-latency interconnect with support of fault-tolerant features, provides a bandwidth of 900 GB/s — 1.5 times more than NVLink3 [78]. It is important that this bandwidth is ideally combined with other bandwithes of the new Nvidia superchips using ARM architecture, which will be discussed below.

NVLink4 uses two differential pairs in each direction (two times less than there were in the NVLink3 channel with the same bandwidth) to form a single channel with an effective bandwidth of 25 GB/s in each direction. Therefore, the H100 now has 18 NVLink4 channels versus 12 channels in the A100 [78].

But more importantly, NVLink4 now supports the NVLink network, allowing up to 256 H100s in different nodes (servers) to securely communicate with each other using a fat tree topology. The corresponding cluster has 32 nodes with 8 H100s each. The NVLink network has a network address space and uses address translation hardware in the H100, ensuring isolation of the network address space and the separate GPUs address spaces (previously on the NVLink network all GPUs share a common address space [78].

The NVSwitch switch initially provided the ability to combine the memory of several GPUs. Powered by the H100, the NVSwitch3 provides hardware-accelerated multicast collective operations, delivering up to two times the bandwidth and reduced latency of NCCL on the A100 for small block size collectives. This significantly reduces the load on SM during collective communications [78].

Using the NVLink network and NVSwitch3 allows you to create large-scale NVLink Switch System networks with very high levels of bandwidth. Nodes in such a network are connected through a second ("external" to the nodes) level of NVSwitches, which reside in switch modules outside the nodes and connect multiple nodes together. Connected nodes are capable of delivering 57.6 TB/s of all-to-all bandwidth [78].

This results in building H100-contained NUMA systems with very high memory scaling levels achievable, in line with the trends of today's large AI training models.

Servers and clusters with H100. H100 GPUs can be hosted on a server using appropriate Nvidia modules. Nvidia supplies HGX modules (roughly speaking, an analogue of a graphics processor board) that contain the H100 and can be used by other companies to create a server containing the H100. HGX H100 is a unit containing 4 or 8 H100. The 4 H100 HGX configuration has fully interconnected point-to-point connections, while the 8 H100 configuration provides full bandwidth between the H100s via the NVSwitch.

Another type of module with H100 from Nvidia is the H100 CNX, where in addition to the H100 there are Nvidia ConnectX-7 SmartNIC capabilities that provide 400 Gb/s throughput in the Infiniband NDR400 or Ethernet 400 variant [78]. It is clear that the HGX modules are primarily focused on AI, while the CNX modules are primarily focused on HPC.

The server configuration as it relates to H100 is determined by these modules. Therefore, here we will note only the DGX H100 servers that have already been supplied for use in supercomputers, focused primarily on AI (although various companies have already announced their servers with H100-PCIe and HGX, for example, Supermicro and Gigabyte).

The DGX H100 with 8 H100 system contains two independent data processing units (DPU) Nvidia Bluefild-3 and 8 ConnectX-7 adapters [78].The DGX H100, in addition to being almost completely ready to solve AI problems, is ideal for working with AI in cloud technology, support for which starts at the level of one H100.

In the DGX H100 SuperPOD cluster (minimum configuration — 32 DGX H100 nodes), compared to the SuperPOD A100, instead of two levels of Infiniband switches, one level of NVSwitch3 switches is used, and the maximum cable length from switch to switch is increased from 5 meters for DGX A100 to 8 meters for DGX H100.

The June 2023 version of the Top500 list has 5 supercomputers using H100-PCIe in x86-64 based servers. The highest performance among these supercomputers (14th place in the Top500) was achieved in the predecessor of the EOS supercomputer announced by Nvidia [281]. This cluster uses 128 DGX SuperPOD nodes. The DGX H100 servers run Ubuntu 22.04 and contains a 56-core Xeon Platinum 8480/2 GHz (Intel Sapphire Rapids, fourth generation Xeon Scalable CPU) and NVidia ConnectX-7 for Infiniband NDR.

Other Top500 supercomputers with H100 are in the second hundred of the Top500 list and beyond. They use servers from other companies and other server CPUs, including AMD EPYC. Of these supercomputers, it is worth noting Henri, which ranks 255th and also tops the similar June Green500 list. Henri uses Infiniband HDR and the nodes use 32-core Xeon Platinum 8362/2.8 GHz (Intel Ice Lake) processors. It should also be noted the expected Kestrel supercomputer [282].

The very fact that the DGX H100 is being used today in a supercomputer suggests its likely focus on AI and working with cloud technologies.

### 4.2.3. *Nvidia ARM hardware to work with H100*

The use of ARM processors in high-performance servers with the H100 GPU should be a vivid real-life illustration of ARM's success in competition with traditional x86-64 server processors, which were used in Nvidia servers with the H100 shipped at the time of writing this review. But the use of ARM in servers with Nvidia GPUs began earlier: multi-core ARM processors are already used in servers with the A100 (for example, from Gigabyte [283]; these servers are now supplied to Russia and Uruguay, and Gigabyte already offers servers with the H100).

In 2023, Nvidia began trial deliveries of modules containing ARM processors to work with the H100. Information on the architecture and performance of Nvidia's ARM CPU (Grace) for running the H100 is still very limited; Available benchmark and application level performance estimates for Nvidia's Grace were more about expectations.

Therefore, the discussion of Nvidia's ARM hardware (the Grace superchip module, which has begun trial deliveries, and the Grace Hopper superchip, where the Grace superchip is integrated with the H100) is very brief here.

TABLE 17. Main technical characteristics of the Grace superchip

| | |
|---|---|
| Architecture | Neoverse V2 (ARM 9.0-A) with 4×SVE2(128 bit) |
| Number of ARM cores | 144 (two Grace processors) |
| Peak performance | 7.1 TFLOPS (FP64) |
| L1 cache | 64 KB I-cache and 64 KB D-cache (per core) |
| L2 cache | 1 MB (per core) |
| L3 cache | 2××117 MB |
| Memory type | LPDDR5X (with ECC) |
| Memory bandwidth | up to 1 TB/s |
| Memory capacity | 240/480/960 GB |
| NVLink-C2C bandwidth (bidirectional) | 900 GB/s |
| PCIe links | 8×PCIe-v5 ×16 with splitting capability |
| Their total (full duplex bandwidth | 1 TB/s |
| TDP | 500 W (with memory) |

Data from: [284, 285]

**ARM Grace superchips.** The Grace superchip is essentially an implementation of an SoC containing two ARM processors and memory, with PCIe support. As noted in [13], it was created specifically for supercomputers and HPC.

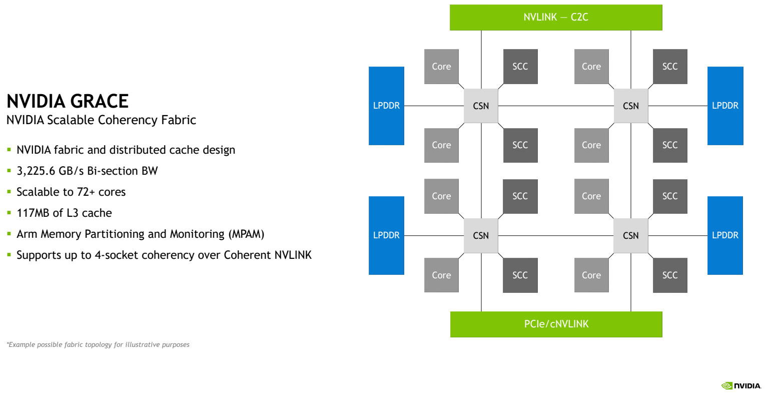
The Grace data analyzed below is based on [284], and additional references are provided only for information from other sources. The main technical characteristics of Grace are collected in Table 17.

The first thing to note is that Grace is based on the latest ARM cores for HPC, Neoverse V2 [286], which were announced in the fall of 2022. These are 64-bit ARM cores with OoO instruction execution, supporting SVE2 instructions with 128-bit vectors.

The Grace superchip consists of two 72-core Grace processors connected by an NVLink-C2C channel supporting memory coherence with a throughput of 900 GB/s, which eliminates interaction between sockets of conventional servers as a bottleneck [13]. At the same time, the peak performance (FP64) of the superchip, 7.1 TFLOPS, is not much lower than that of the A100 without the use of tensor cores.

It should be noted that in modern Neoverse V2 cores, aimed at CPUs with a large number of cores, the L1 I-cache is protected by parity checking, and the L1 D-cache is protected by ECC codes. The L2 D-cache of ARM Neoverse V2 cores is also protected by ECC codes and can have a capacity of 1 or 2 MB [286]; in Grace, Nvidia chose the 1 MB option. The L3 cache is naturally also protected by ECC codes; or other details of the Neoverse V2 microarchitecture, see [286].

An interesting and very important feature of the Grace superchip developed by Nvidia is the Scalable Coherency Fabric (SCF), which has



**NVIDIA GRACE**

NVIDIA Scalable Coherency Fabric

- NVIDIA fabric and distributed cache design
- 3,225.6 GB/s Bi-section BW
- Scalable to 72+ cores
- 117MB of L3 cache
- Arm Memory Partitioning and Monitoring (MPAM)
- Supports up to 4-socket coherency over Coherent NVLINK

*Example possible fabric topology for illustrative purposes

Figure 13. General construction of the Grace microarchitecture (Figure from [13])

a mesh structure over which the cores and L3 cache partitions (SCCs in Figure 13) are distributed, and ensures data flow between the cores, NVLink-C2C, memory and system I/O with a total half-bandwidth of more than 3.2 TB/s. This is illustrated in Figure 13. SCF is designed to scale beyond a single Grace CPU to form a super chip with 144 cores. Cores and SCCs are distributed across the mesh, and CSN (Cache Switch Nodes) act as the interface between the CPU cores, the cache, and the rest of the system [13].

The Grace superchip also has special tools that support partitioning of hardware resources, which can be effectively used when working in a cloud environment [284].

Nvidia developers explain their choice of 32-channel LPDDR5X memory technology as the “golden mean” between HBM2E and DDR5 in terms of capacity, bandwidth, cost and power consumption [13]. The maximum theoretical bandwidth of LPDDR5X in the Grace superchip (1 TB/s) is slightly higher than the bandwidth of NVLink-C2C (see Table 17).

It should also be noted that Grace supports four PCIe-v5 $\times16$ channels (in addition, for various tasks there are two more low-speed PCIe-v5 x2 channels) [13]. This immediately makes this superchip the leader among CPUs in terms of I/O performance.

But the TDP of the Grace superchip (this is a dual-processor SoC) is comparable to the TDP of modern GPUs; this is higher than the A100 and H100 PCIe (but lower than the H100 SXM). To work with the Grace superchip, air or liquid cooling can be used [284].

Nvidia in [284] also provides expected estimates of some of the Grace benchmarks. These relate to STREAM test data (up to 400

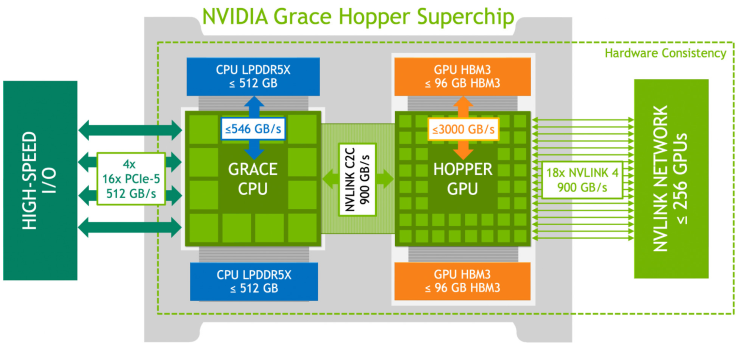


FIGURE 14. Grace Hopper with support for working with NVLink4 (Figure from [287])

GB/s for one Grace CPU and up to 800 GB/s for the entire superchip), SPECrate2017_int_base (370 and 740 for one Grace CPU and superchip, respectively), and significant speedup values in some applications (including areas CFD and weather forecasting) compared to a dual-processor server with AMD EPYC 7763 (Milan). Other initial data have appeared, but they are still completely insufficient.

Nvidia talks about a twofold increase in power efficiency compared to traditional CPUs used in data centers [284]. Plans to create supercomputers based on Grace were announced by the US Los Alamos National Laboratory and the Swiss National Computing Center [14].

### 4.2.4. *Grace Hopper superchips*

In addition to Grace, Nvidia also announced Grace Hopper, in which Grace (two 72-core processors) is integrated with the H100. The main basic indicators of this superchip are approximately the sum of the indicators of the H100 and Grace discussed above, and depend on which variants of each of these two integrated components is used in the Grace Hopper.

In version 1.01 of the document [287], on which further analysis of Grace Hopper superchips is based, it is indicated that the LPDDR5X size for Grace in the Grace Hopper superchip is up to 512 GB, size of HBM3 memory for H100 is up to 96 GB, and about the presence of two options Grace Hopper superchip — with support for working with NVLink4 (see Figure 14) and without it (then systems with such superchips are connected via Infiniband NDR). It is clear that the integration of two CPUs, H100 and memory gives a high TDP, 1000 W is specified in [287].

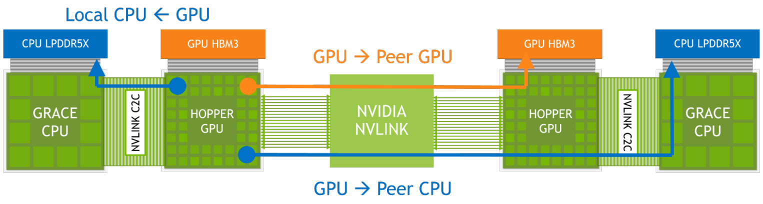


Figure 15. General microarchitecture of Grace Hopper (Figure from [287])

After completing work on this section of the review at the same URL [287], Nvidia posted new versions of this document (the latest version known to the author is 1.11), where the superchips in question are now called GH200 Grace Hopper. The most important changes there are associated with the emergence of new GH200 models, in which instead of HBM3, HBM3E with a capacity of 141 GB with a bandwidth of 4.8 TB/s will be used (versus 4 TB/s when working with HBM3). The main characteristics of the GH200 Grace Hopper architecture, of course, have not changed. But since due to the rapid development of GH200 and related computing systems in 2023, new modifications appeared in later versions of this document, and sometimes even the names used were modernized, the analysis of GH200 in this review was carried out in a limited scope and is further based on version 1.01 of the document.

As the interconnect between the Grace processor and the H100 uses NVLink-C2C, which provides high bandwidth and low latency (see Figure 15) [287]. NVLink-C2C provides memory coherence and hardware support for system-wide atomic operations, which improves the performance of synchronization primitives. Memory coherence allows developers to transfer only the data they need (rather than entire pages of memory) from Grace to H100 and back again, and naturally simplifies programming heterogeneous applications. The performance of non-local memory accesses also improves (for example, when CPU and H100 threads access memory located on another device) [287].

A key feature of the GH200 superchip is the use of extended GPU memory (EGM). Each GPU from a multi-node computing system with an NVLink network based on GH200 has access to both the LPDDR5X memory of all Grace processors and the HBM memory of all GPUs, i.e. The total memory capacity for the GPU is dramatically expanded. A unified

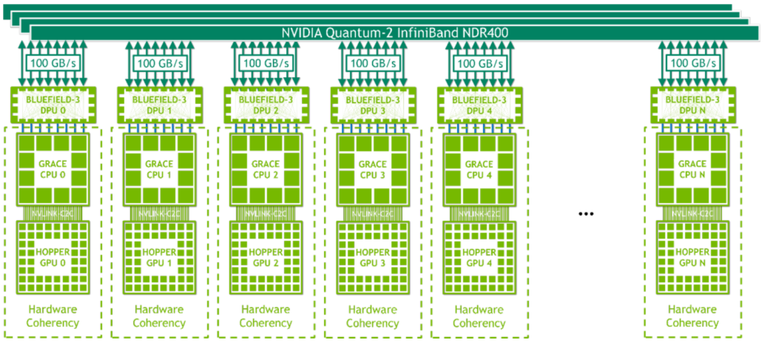


FIGURE 16. General construction of interconnections GH200 with NVLink support (Figure from [287])

memory is formed with common page tables, a common virtual address space. The speed of GPU access to local memory is set by NVLink-C2C, and to remote memory by NVlink4 [287] (see Figure 14 and Figure 15).

Servers with GH200 are already offered, for example, by Gigabyte [288]. Nvidia offered computing systems with the GH200 — the MGX GH200 and DGX GH200 platforms. The MGX GH200 modular platform (there are options with HBM3 or HBM3e) is as similar as possible to a server with one GH200 superchip, without support for the NVLink switching system. Nodes with MGX GH200 can form traditional clusters, where Infiniband or Ethernet can be used for communication, working through the DPU.

DGX GH200 is a supercomputer for AI; there, 256 superchips are combined using the NVLink interconnect, giving a memory address space of up to 144 TB. A summary of these Nvidia AI-focused platforms is provided in version 1.11 of the document [287].

From the point of view of the possibility of building the most powerful supercomputers, the basis for ultra-high levels of performance scaling is provided by the use of NVSwitch3 with NVLink4 channels in conjunction with the GH200, since this provides Nvidia's original two-level scaling. Its capabilities are illustrated in Figure 16, which relates to working with the GH200 variant with NVLink support [287].

At the first level, it's possible to combine a set of GH200 superchips into a single domain, connected through the NVSwitch3 switching system (with a total bandwidth of all GPU threads in the domain up to 900 GB/s per superchip). The use of these interconnects makes it possible to form a common memory of the entire domain, including both the LPDDR5X memory of the Grace processors and the HBM3 memory of the H100 [287]. In late 2023, Nvidia announced another single-rack compute system (GH200 NLV32) — with a much smaller domain than the top-end DGX GH200 configuration. Already at this level a very high level of performance is achieved; Nvidia also calls this system a supercomputer (see, for example, [289]).

At the second level, clustering of domains is possible using the most modern, widely used interconnects. For this, PCIe-v5 channels supported in Grace are used. Figure 16 shows a possible variant of such formation of a cluster of domains using the Bluefild-3 DPU. For communication with Grace, this DPU has 32 PCIe-v5 lanes, which has a total bidirectional bandwidth of 256 GB/s (this DPU has another 32 GB of own memory). And to form a cluster of domains, the DPU has 1 or 2 additional ports with speeds up to 400 Gbps (they can work with Infiniband NDR400 or Ethernet 400) [290], as shown in Figure 16. The use of DPU removes the Grace need to manage transferring data over the cluster interconnect.

It is clear that scaling on such a system will be a difficult task even for AI. But the author does not know, even through the announcement, about the possibility of such an "option" and the level of scaling of systems with GPUs from other companies. But in general, it is clear that Nvidia with the GH200 is focused on delivering more work-ready computing systems.

### 4.2.5. *CUDA Extensions for H100*

Everything stated below in this section is based on the description of the H100 hardware [78]. But first, it is advisable here to point out the general structure of the CUDA platform in its part related to compilers, since over time new programming languages appear that can also be used to obtain kernels executed on Nvidia GPUs. At the same time, the CUDA platform provides different compilers with a unified software stack, and creating a compiler for a new programming language basically comes down to writing only the front end part of the compiler.

These compilers are based on LLVM (in [78] the use of its 7th version is indicated). In addition to the capabilities of the C++ and Fortran ISO standards used for working on GPUs, as well as their CUDA extensions, back in 2021 in CUDA 11.4 Nvidia introduced CUDA-Python [291], and there are also developments for Julia and other programming languages.

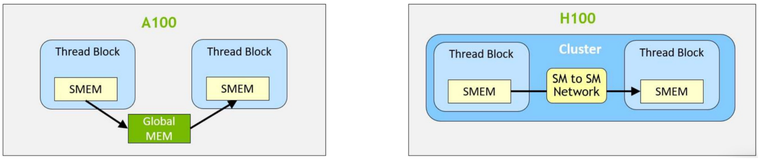


FIGURE 17. Working with DSMEM in A100 and H100 (Figure from [78])

The corresponding front-end components of these compilers generate an intermediate representation (IR) called NVVM IR [292], which is based on the famous LLVM IR. Next, using the libNVVM library, a PTX code is generated, from the virtual ISA of which a kernel is obtained that runs on the GPU [78].

The most important thing for achieving high performance of GPGPU is a provision of data locality, which gives high-speed access to nearby memory levels, and asynchronous execution, when the calculations themselves are performed independently (simultaneously) with data transfer.

Previously (and for the A100), the CUDA programming model included blocks of threads, from which a grid of threads was formed. IIn H100, the number of SMs has increased significantly and another level has been added to the overall thread hierarchy, a thread block cluster, which includes a set of thread blocks spanning multiple SMs. All this is reflected in the H100 hardware implementation with a new larger level of memory localization, since clusters of thread blocks in the H100 work simultaneously on SMs inside the GPC, ensuring fast data exchange between threads in the cluster.

Cluster threads can directly access the shared memory of other SMs using load, store, and atomic operations (which cannot be partially executed). For this purpose, DSMEM (Distributed Shared Memory) is formed. Figure 17 shows how data is exchanged between thread blocks in A100 and in H100 with DSMEM. A thread block cluster has access to more shared memory capacity than a single thread block—the virtual address space of "shared" memory. Compared to using global memory, DSMEM speeds up data exchange between blocks of threads by approximately 7 times. [78].

On the other hand, from the author's point of view, the appearance of a of thread block cluster on a new GPU model is a clear demonstration that widely used programming models for GPUs, which require the highest possible performance, automatically become poorly portable to other GPU models.

Other notable extensions in CUDA for H100 relate to asynchronous execution, allowing for overlapping (simultaneous execution) of data movement, computation, and synchronization [78]. To achieve this, the H100 has a new asynchronous memory copy module TMA and a new asynchronous transaction barrier. TMA can transfer large (up to the shared memory capacity) blocks of data and multidimensional tensors from global memory to shared memory and back.

The TMA operation is asynchronous and uses shared memory-based asynchronous barriers like the A100. In addition, one thread in a warp can be selected to perform an asynchronous operation, and then multiple threads can wait for the data transfer to complete using a barrier. TMA frees threads to do other independent work, since after a thread creates a so-called copy handle before running TMA, TMA itself performs all other functions as part of the H100 hardware.

Asynchronous barriers first appeared in the A100, and the H100 introduced a new primitive for asynchronous memory copies, the asynchronous transaction barrier. The write command to shared memory transfers not only the data, but also the number of transactions. The asynchronous transaction barrier blocks threads on a wait command until all producer threads have completed the "Arrive-On" operation and the sum of all transaction counters has reached the expected value. See [78] for more details. These capabilities are naturally used to work with thread block clusters.

### 4.2.6. *H100 Initial Performance Data*

Data on the H100's performance in benchmarks and HPC applications at the time of the June 2023 Top500 list was virtually non-existent, except for the starting data in the Hot Chips 34 report [293]. We will also note the available (as of June 14, 2023) data from the tiny suite of the SPEChpc 2021 benchmark [226], where results for the Lenovo ThinkSystem SR655 V3 server are presented with one and two H100-PCIe-80GB (with EPYC 9654P). We compared these data with Lenovo's SPEChpc 2021 results for one and two A100-PCIe-80GB on another ThinkSystem SR670 V2 server (with Xeon Platinum 8380). For the base variant of the test (the peak variant for the A100 was not carried out), the resulting performance on one H100 was 1.33 times greater than that of the A100, and on two H100s the same acceleration was 1.51 times. But should be kept in mind that for parallelization, OpenACC tools (plus MPI) were used here.

Other interesting data on the performance of H100 were obtained for the well-known molecular dynamics application Amber version 22 [**294**]. There, for typical test molecular systems for this application, calculations were carried out that gave performance estimates (the number of simulated nanoseconds per day of calculation) on different graphics processors. The relative performance calculated from these values shows the acceleration of H100 relative to A100 by 1.11-1.28 times for dihydrofolate reductase (23 558 atoms), 1.27 times for nucleosomes (25 095 atoms), 1.42-1.43 times for a protein of 90 906 atoms, 1.40 times for cellulose with a large number of macromolecules (408 609 atoms), 1.35 times for a satellite of the tobacco mosaic virus (1 067 095 atoms).

In calculations of smaller molecular systems, the accelerations obtained on H100 were also and greater than those indicated above, but we do not present these data here. But it is useful to note that for all the molecular systems calculated in [**294**], the performance of the Nvidia RTX 3090 is close to the A100, and the Nvidia RTX 4090 is close to the H100.

There is also other data on Amber22 performance on the H100 compared to performance on the A100, but these are discussed below in Section 5.3.2, with comparison to performance on the MI250.

But first of all, new generation GPUs are usually focused on solving AI problems, so the initial performance data appears here. For the H100, first became available Nvidia's preliminary results for one of the classic AI (machine learning) benchmarks suites, MLCommon — MLPerf Training in version 2.1 [**278**].

The calculation times presented in [**278**] for H100 (in the "closed"/"preliminary" section of MLPerf Training) were obtained on the DGX H100 system, containing, in addition to 8 H100, two Xeon processors (56 cores) and running the Ubuntu 20 distribution. Data analysis for the DGX A100 system with 8×A100-SXM-80GB in benchmarks that give comparable results to the H100 shows that the calculation time on the H100 is about two times less than on the A100, with the exception of the NLP benchmark, where the H100 is 3.8 times faster.

At the end of June 2023, data on the performance of H100 and A100 appeared on the new version of MLPerf Training v3.0 [**295**], and in September — on the new version of MLPerf inference datacenter v.3.1 [**298**], which are illustrated in Table 18 and Table 19.

Table 18. Performance data for H100 and GH200 in MLPerf inference datacenter v. 3.1 benchmarks (all data in the category "closed"/"available"); data have been selected for maximum comparability for 99% accuracy and rounded to 2 significant digits

| GPUs | 1 | 2 | 4 | 8 |
|---|---|---|---|---|
| Image Classification/ResNet | | | | |
| H100-PCIe-80GB | 47(55)[1] | 106(115)[2] | 206(200)[2] | 368(443)[1] |
| H100-SXM-80GB | 73(89)[3] | | 312(354)[3] | 584(704)[4] |
| GH200-96GB | 77(93) | | | |
| A100-PCIe-80GB | | | 147(158)[5] | |
| A100-SXM-80GB | | | | 305(340)[6]; 290(326)[7] |
| NLP/BERT | | | | |
| H100-PCIe-80GB | 4.6(5.7)[1] | 9.1(12)[2] | 18(23)[2] | 35(46)[1] |
| H100-SXM-80GB | 7.3(8.8)[3] | | 29(36)[3] | 56(70)[4] |
| GH200-96GB | 7.7(10) | | | |
| A100-PCIe-80GB | | | 12(13)[5] | |
| A100-SXM-80GB | | | | 25(28)[6]; 25(28)[7] |

The performance data (number of querues in the server scenario and number of samples - in parentheses - in the offline scenario) is **reported per millisecond instead than per second**.
All calculations were carried out using TensorRT 9.0.0 and CUDA 12.2.
[1] Data from Nvidia on Gigabyte G482-Z54, with 2×EPYC 7742;
[2] Data from Dell on Dell PowerEdge R760x with 2×Xeon Platinum 8480+
[3] Data from Dell on Dell PowerEdge XE9640 with 2×Xeon Platinum 8468
[4] Data from Nvidia on DGX H100 with 2×Xeon Platinum 8480C
[5] Data from Dell on Dell PowerEdge R750x with 2×Xeon Gold 6338
[6] Data from HPE on HPE ProLiant XL675d Gen10 Plus with 2×EPYC 7763
[7] Data from Oracle on Oracle BM.GPU.A100-v2.8 with 2×EPYC 7J13

Data for these tables were selected to provide the greatest possible comparison of performance, but since they were actually obtained after the completion of data selection for review, they are not analyzed in detail here.

Table 18 shows selected data from the MLPerf inference datacenter version 3.1 benchmarks suite. The data in this table shows the H100's clear performance gains over the A100, the H100-SXM's more subtle performance gains over the H100-PCIe, and the 96GB GH200's increased performance over the H100-SXM (and in more so over the H100-PCIe). This demonstrates the increase in performance for AI area when working with a single H100 GPU in the Grace-integrated GH200 variant with 96 GB HBM-memory. In addition, these data show good scalability at the number of GPUs used in the server increases from 1 to 8.

TABLE 19. Machine learning performance in MLPerf Training v3.0 benchmarks [295] in minutes

| GPUs | 2 | 4 | 8 |
|---|---|---|---|
| Image classification | | | |
| A100-PCIe | | 58,[1] 61[2] | 32[3] |
| A100-SXM | | 54[4] | 27[5] |
| H100-PCIe | 82[6] | 45,[2] 39[7] | 20,[8] 21[9] |
| H100-SXM | | 27,[10] 26[11] | 13,[12] 13[13] |
| Image segmentation (medical) | | | |
| A100-PCIe | | 47,[1] 48[2] | |
| A100-SXM | | 47[4] | 23[5] |
| H100-PCIe | 67[6] | 32,[2] 31[7] | 19,[8] 18[9] |
| H100-SXM | | 22,[10] 22[11] | 12,[12] 12[13] |
| Object detection, light-weight | | | |
| A100-PCIe | | 171,[1] 176[2] | |
| A100-SXM | | 222[4] | 79[5] |
| H100-PCIe | | 114,[2] 107[7] | 54,[8] 56[9] |
| H100-SXM | | 72,[10] 72[11] | 37,[12] 37[13] |
| Object detection, heavy-weight | | | |
| A100-PCIe | | 86,[1] 81[2] | 47[3] |
| A100-SXM | | 83[4] | 38[5] |
| H100-PCIe | | 62,[2] 55[7] | 28,[8] 28[9] |
| H100-SXM | | 40[10] | 19,[12] 20[13] |
| Speech recognition | | | |
| A100-PCIe | | 63,[1] 64[2] | |
| A100-SXM | | 55[4] | 29[5] |
| H100-PCIe | 77[6] | 51,[2] 45[7] | 23,[8] 28[9] |
| H100-SXM | | 27,[10] 27[11] | 19,[12] 19[13] |
| Natural languages processing | | | |
| A100-PCIe | | 45,[1] 51[2] | |
| A100-SXM | | 32[4] | 15 |
| H100-PCIe | 43[6] | | 10,[8] 10[9] |
| H100-SXM | | 11,[10] 11[11] | 5.4,[12] 5.4[13] |
| Recommendation (DLRM) | | | |
| A100-SXM | | | 8.4[5] |
| H100-SXM | | 8.8[10] | 4.3,[14] 4.3[12] |

[1] Data on ESC4000-E11 with 2×Xeon Platinum 8462Y+
[2] Data on R750x with 2×Xeon Gold 6338
[3] Data on ThinkSystem SR670 V2 Server with 2×Xeon Platinum 8360Y
[4] Data on XE8545 with 2×EPYC 7763
[5] Data on XE9680 with 2×Xeon Platinum 8480+
[6] Data on D54Q-2U with 2×Xeon Gold 6430
[7] Data on R760x with 2×Xeon Platinum 8480+
[8] Data on ESC8000A-E12 with 2×EPYC 9654
[9] Data on AS-4125GS-TNRT with 2×EPYC 9554
[10] Data on SYS-421GU-TNX with 2×Xeon Platinum 8460H
[11] Data on XE8640 with 2×Xeon Platinum 8468
[12] Data on XE9680 with 2×Xeon Platinum 8470
[13] Data on AS-8125GS-TNHR with 2×EPYC 9634
[14] Data on G593-SD0 with 2×Xeon Platinum 8480+

All data in the table refers to "available on-premise" and was obtained on a GPUs with a memory capacity of 80 GB and **rounded to two significant digits**. For information about the AI models used in the benchmarks and the underlying datasets used, see [295].

Table 19 shows a sample of approximately one-sixth of the MLPerf Training v3.0 benchmark data table presented in [**295**], where are, for example, also results for a larger number of GPUs used. Such a large amount of data presented in this table allows to compare the performance of the H100-SXM, H100-PCIe and A100-SXM/PCIe with different numbers of GPUs used in the server, using different software and different hosts.

The data in this table shows that possible reasonable variations in the host hardware (of course, CPUs containing dozens of cores were used here) and software chosen by server manufacturers for these benchmarks do not have a strong impact on the achieved performance and are therefore not analyzed here.

The data in this table in almost all benchmarks shows good scaling with the number of GPUs in the server up to 8. As a zero approximation for comparing performance, we can assume that the dependence of performance on the number of GPUs is linear, although this is not always the case — for example, when moving from 4 to 8 H100-PCIe speedup in the medical image segmentation benchmark was only 70 percent.

The table data shows that noticeably higher performance is achieved when working with the H100-SXM compared to working with the H100-PCIe (for example, one and a half times more for 4 GPUs in image classification and image segmentation benchmarks); or that the performance of the H100-SXM is twice as high as that of the A100 on the same benchmarks with the same number of GPUs; or that H100-PCIe is one and a half times faster than A100 under the same conditions, and so on. It also happens that scaling with the number of GPUs is rather insufficiently high for AI (for example, when moving from 4 to 8 H100-PCIe in the medical image segmentation benchmark), but such an analysis, for obvious reasons, is not carried out here.

## 4.3. Nvidia GPUs Summary

Despite the emergence of alternative Nvidia GPUs from AMD and Intel, there are no signs of weakening in the position of Nvidia GPUs in hardware and software terms — their use in quantitative terms will continue to develop rapidly. Software tools for Nvidia GPUs are the most widely available, and new software tools are usually released for them earlier than for competitive GPUs. New generation GPU architectures from Nvidia provide a high level of compatibility with earlier GPU models. Accordingly, the portability of the software to newer models is significantly higher than that achieved when switching to work with competitor GPUs.

While Nvidia's GPU market share could theoretically fall in the global market due to the introduction of AMD and Intel GPUs, the global increase in AI efforts will boost the adoption of Nvidia GPUs. The possible performance advantages of Intel and AMD GPUs need to be proven in real applications. From an HPC perspective, it's worth paying attention to Nvidia's increasing focus mainly on AI area.

## 5. Next generation GPUs from AMD

The new generation GPUs in this review include GPUs that appeared after the basic Nvidia V100 and A100 GPUs. AMD has come a long way in recent years not only in the field of successful competition with Intel x86 processors (AMD EPYC is pushing aside Intel Xeon not only in the market of traditional servers, but also in the market of servers with GPUs). The introduction of the MI100 [296] at the end of 2020, and the MI200 family the following year [297, 299] showed that competition with Nvidia GPUs is starting. It was the GPU MI100 and MI200 that became the first representatives of GPUs produced, classified in this review as a new generation.

The MI100 was sometimes seen as a precursor to AMD GPUs in upcoming supercomputers (the Spock cluster with MI100 in nodes is a predecessor to Frontier [254]; the MI100 began to be used in the nodes of LUMI, which took third place in the June Top500 list, when its nodes already began using the MI250X [42]). The MI100 architecture has already been discussed in publications — see, for example, the review [21]. By analogy with the V100, here we will limit ourselves only to the summary technical characteristics of the MI100 (see Table 20) and Table 21, but we will analyze data on the achieved performance of this GPU, including in comparison with other GPUs considered in the review.

The relevance of the analysis of GPUs from the MI200 family is increased by the fact that they have now been used to build not only the No. 1 supercomputer in the Top500, which for the first time in the world crossed the 1 EFLOPS barrier (Frontier), and the third in the Top500 (LUMI) list, which belongs to the European Union (since LUMI is installed in Finland — this also demonstrates significant European success in the supercomputing field): MI250X GPUs are used in 2% of supercomputers from the Top500, which also reflects another success of the manufacturer, HPE/Cray, in whose hardware the MI250X was placed.

Table 20. Specifications of modern AMD and Nvidia GPUs

| GPUs | MI100 | MI210 | MI250 | MI250X | A100 with PCIe | A100 with SXM4 | H100 with PCIe | H100 with SXM5 |
|---|---|---|---|---|---|---|---|---|
| TSMC technology, nm | 7 | 6 | | | 7 | | 4 | |
| Number of active cores (stream processors)[1] | 7680 | 6656 | 13312 | 14080 | 6912 | 6912 | 14592 | 16896 |
| Number of active CUs[2] | 120 | 104 | $2 \times 104$ | $2 \times 110$ | 108 | 108 | 114 | 132 |
| GPU base/boost clock (MHz) | 1000/1502 | 1000/1700 | | | 765/1410 or 1065/1410 | 1095/1410 or 1275/1410 | 1095/17554 | 1590/19804 |
| Peak performance: FP64 (TFLOPS) | 11.5 | 22.6 | 45.3 | 47.9 | 9.7 | 9.7 | 25.6 | 33.5 |
| FP64 with tensor cores (TFLOPS) | No | 45.3 | 90.5 | 95.7 | 19.5 | 19.5 | 51.2 | 66.9 |
| FP32 (TFLOPS) | 23.1 | 22.6 | 45.3 | 47.9 | 19.5 | 19.5 | 51.2 | 66.9 |
| FP32 with tensor cores (TFLOPS) | 46.1 | 45.3 | 90.5 | 95.7 | No support: lower precision inputs | | | |
| FP16/BF16 (TFLOPS) | 184.6/ 92.3 | 181 | 362.1 | 383 | 78/312[3] | | 205[3] | 268[3] |
| INT8 (TOPS) | 184.6 | 181 | 362.1 | 383 | 624 | | 1513 | 1978.9 |

[1] Low-level SIMD cores with support for multiply-and-add commands are also called CUDA processors by Nvidia (matrix/tensor cores are not taken into account here);
[2] for Nvidia — streaming multiprocessors (SM);
[3] when using sparsity — twice as high; Performance with FP16, BF16 and INT8 relates to tensor kernels.

Performance with FP16, BF16 and INT8 refers to tensor cores.

Table 21. Specifications of modern AMD and Nvidia GPUs (continuation)

| GPUs | MI100 | MI210 | MI250 | MI250X | A100 with PCIe | A100 with SXM4 | H100 with PCIe | H100 with SXM5 |
|---|---|---|---|---|---|---|---|---|
| GPU memory type | HBM2 | HBM2E | | | | | | HBM3 |
| Memory bus, bit | 4096 | | 8192 | | 5120 | | | |
| Memory capacity, GB | 32 | 64 | 2 × 64 | | 40 or 80 | | 80[5] | |
| Memory clock, MHz | 1200 | 1600[6] | | | 1215 or 1512 | 1215 or 1593 | 1593 | 1313 |
| Its bandwidth, GB/s | 1229 | 1638 | 3277 (2 × 1638.4) | | 1555 or 1935 | 1555 or 2039 | 2039 | 1681 |
| Interconnect of GPU with GPU or with CPU | Infinity Fabric 2.0 | Infinity Fabric 3.0 | | | with CPU: PCIe v4 | NVLink-3.0 | with CPU: PCIe v5 | NVLink-4.0 |
| Its bandwidth, GB/s | 3 × 92 | 3 × 100 | 6 × 100 | | with CPU: 64 | 600 | with CPU: 128 | 900 |
| L1 cache, KB | 16 on CU | | | | 192 on SM | | 256 on SM | |
| L2 cache, MB | 8 | 16 | | | 40 or 80 | 40 | 50 | |
| TDP, W[4] | ; 300 PSU: 700 | ; 300 PSU: 700 | 500; 560 PSU: 900 | 500; 560 PSU: 900 | 250 or 300 PSU: 700 | 400 PSU:800 | 350 PSU: 750 | 700 PSU: 1100 |

[4] There is a model with HBM3/96 GB and a different GPU clock.

[5] for tensor operations with reduced (less than FP32) precision, the boost clock is lower;

[6] TDP: the number after the semicolon for AMD is peak, PSU: suggested power for Power Supply Unit;

This table data are taken from the database [185] and from manufacturers websites.

## 5.1. GPUs AMD MI200

### 5.1.1. *Microarchitecture and technical characteristics of different MI200 models*

The MI200 family includes three different models — MI210 [300], MI250 [301] and MI250X [302], of which the top model MI250X appeared and was the first to be supplied to the Frontier supercomputer [303], the leader of the Top500 list.

As AMD transitioned from producing its traditional Radeon Instinct graphics processors to next-generation GPUs, the architecture also changed. If earlier AMD graphics processors used the GCN (Graphics Core Next) architecture, then CDNA (Compute DNA) was developed for MI100 (Compute DNA) [304], and in MI200 it was upgraded to the second version CDNA 2 [305]. When these architectures change, naturally, a certain continuity is maintained, primarily in the computing units, the improvement of which largely occurs in technological and quantitative indicators.

For the analogue of the basic computing unit SM in Nvidia GPUs for AMD GPUs, the term CU is used (see Table 1), which is also used in BR100. Sufficiently detailed information on the construction of the CU and its main blocks for CDNA is available in [304]. The CU in CDNA 2 contains, in particular, a dispatcher for working with threads, files of ordinary and vector registers, cores (including matrix ones, see below), L1 cache and LDS (Local Data Share) memory [305]. LDS shared memory (often also called LSM, Local Share Memory) is used by threads inside a warp (wavefront in AMD terminology, see Table 1) [305].

The L1 cache capacities for the MI100 and various MI200 models, as well as other important indicators of these AMD GPUs, in comparison with the A100 and H100 are shown in Table 20 and Table 21. Taking into account the delays in PVC supplies and the possible freeze in production of the BR100, the greatest interest in this review and for comparison purposes GPUs in general currently represent the GPUs from this table.

But first of all, we should point out the technological indicators. MI100 began to be produced using TSMC 7 nm technology (like the A100), and MI200 — using 6 nm technology (only in the latest H100 that Nvidia introduced, used 4 nm technology). The MI100 contains 25.6 billion transistors with die size of 750 mm$^2$ — and the MI250X already has 58.2 billion (with a smaller die size 724 mm$^2$) [185]. These indicators are important, among other things, because they are associated with possible cost indicators and TDP values.

These data show the similarity of such indicators for MI100 and V100 (see also Table 10). At the same time, in MI250X the number of transistors is slightly larger than in A100 (there are 54.2 billion of them), and the MI250X die size is noticeably lower (in A100 — 826 mm$^2$). Accordingly, ideas arise about the possible comparability of MI100 and V100, as well as MI250X and A100 — which has been tested in a number of publications with performance data. As for the comparison of the technological indicators of the MI250X and A100, it requires fundamental clarification, which will be discussed below.

To compare the MI250X with the A100, we should add a comparison with the H100, which will appear in 2023. Here, the new 4 nm technology level allowed Nvidia to place 80 billion transistors with a reduced die size (compared to the A100) 814 mm$^2$.

The main indicators of GPU MI100, MI200, A100 and H100 are given for comparison in Table 20, 21, and Figure 18 shows a higher (than CU) level of computing power scaling in CDNA 2 — GCD, which also has 4 Compute Engine's (CE), each containing 28 CUs (two columns of 14 CUs each in the figure [**305**]. Two CUs of the 112 physical CUs are disabled [**41**] — accordingly, Table 20 indicates 110 CUs.

The GCD die is of fundamental importance for AMD technology: the top MI250 and MI250X models each contain by two GCDs [**305**], connected as a chiplet [**299**].

As GCD interconnect the Infinity Fabric 3.0 (discussed in the next section of the review) is used, but its bandwidth is much lower than the GPU memory bandwidth (see Table 21). When programmed, MI250 and MI250X will be treated as two different GPUs [**305**]. Accordingly, returning to the technological comparison with the A100 and H100 GPUs, MI250/MI250X can also be perceived as GPU pairs. And per one GCD in the MI250X, the number of transistors in it, roughly speaking, is 2 times less than in the A100.

If we move from the CDNA 2 architecture to the supplied MI200 models, they differ in the number of active CUs (see Table 20). According to [**305**], each CU contains 64 cores (64 shader cores or stream processors in AMD terminology, equivalent to four SIMD blocks with vectors of 16 numbers) — these are some analogues of CUDA cores in Nvidia GPUs. Each such core can perform multiply-and-add commands with FP64 numbers. Then the peak GPU performance (FLOPS) is twice the product of the number of such GPU cores and the clock frequency. According to the established tradition, manufacturers use the accelerated frequency in calculations of peak performance — such numbers are given in this table.

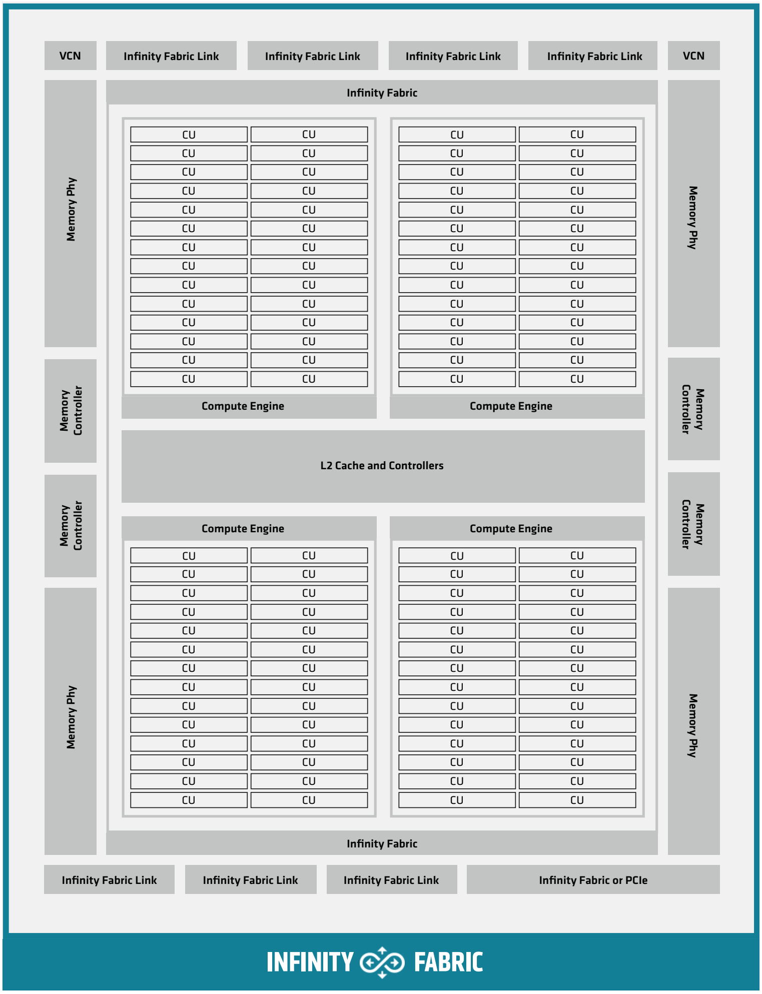


Figure 18. General construction of a GCD die (Figure from [305])

In CDNA2, the ALUs located in the CU became 64-bit, and it became possible to pack two FP32 numbers in place of one FP64, and work with them, for example, in multiply-and-add instructions [305]. The MI250X's peak performance with packaged FP32s doubles to 95.7 TFLOPS without using matrix cores.

In CDNA (in CUs), matrix cores have appeared (analogues of tensor

TABLE 22. The number of results obtained per clock cycle on one CU in CDNA (MI100) and CDNA 2 (MI200)

| Format | By vector units | | By matrix units | |
|---|---|---|---|---|
| | CDNA/MI100 | CDNA 2/MI200 | CDNA/MI100 | CDNA 2/MI200 |
| FP64 | 64 FLOPS | 128 FLOPS | — | 256 FLOPS |
| FP32 | 128 FLOPS | 128 FLOPS | 256 FLOPS | 256 FLOPS |
| FP16 | — | — | 1024 FLOPS | 1024 FLOPS |
| BF16 | — | — | 512 FLOPS | 1024 FLOPS |
| INT8 | — | — | 1024 numbers | 1024 numbers |

Data from [307]. It also lists the number of clock cycles required to perform each type of MFMA operation. Running mixed format MFMA with FP32 results gives 1024 FLOPS per clock cycle.

cores in Nvidia GPUs), which can execute MFMA (Matrix-Fused-Multiply-Add) commands — “multiply-and-add” on small-sized matrices using mixed precision, but there the maximum precision is FP32 [304], as in the V100. And in CDNA 2 — and accordingly in MI200 (similar to A100) MFMA can also work with the FP64 format. Acceptable sizes of matrices with FP64 ($M \times N \times K$ in formula (1)): $16 \times 16 \times 4$ or $4 \times 4 \times 4$ [299].

It is useful to note here the peculiarity of the MFMA command set (it is a set, since for each combination of number formats used in different matrices there is its own version of the MFMA command) and similar WMMA commands in Nvidia GPUs. They perform the operation expressed by formula (1) with $\alpha = \beta = 1$, but the result is placed in another matrix **D** instead of **C** on the left side of this formula. There is no support for working with specific sparse matrices for AI area (as in the A100) in CDNA 2. ISA documentation for CDNA 2 is available at [306].

To calculate the peak performance of MI100 and MI200, you need to know the number of results obtained per clock cycle in one CU; this data is presented in Table 22. For example, each CU in CDNA 2 has 4 matrix cores, each of which produces 64 FP64 results per clock cycle [305], which after multiplying the total number of matrix cores in the GPU by 64 and by the clock frequency gives the peak performance shown in Table 20 when working with matrix cores.

In general, if we compare the peak performance data presented in Table 20, we can see that when working with vector cores, the MI100 is ahead of the A100 (and naturally the V100, see Table 12) for FP64 and FP32 in this indicator, giving this an potential advantage for all (working with vectors) HPC applications (unless performance is limited by matrix multiplication). There is no support for FP64 in the MI100 matrix cores, and with an accuracy below FP32, the MI100 is competitive with the V100 in this performance indicator, being much inferior to the A100.

On vector cores, the MI250 and MI250X models containing two GCDs each outperform the A100 in peak floating point performance. The same occurs when working with matrix/tensor cores (with the exception of FP32). However, A100 for the field of AI, when using special sparsity of matrices with tensor cores when working with reduced precision relative to FP64, is ahead in terms of peak performance.

The actual performance achieved fundamentally also depends on other factors, primarily the memory hierarchy — but first you need to decide how the comparison is made: logically, the MI250 and MI250X models are presented as two GPUs. If we divide the peak performance of these two models presented in Table 20 by two, then for FP64 all MI200 models are still significantly ahead of the A100, and for FP32 their advantage is small (for tensor cores the A100 also shows higher performance, but the original data are not supported in standard FP32 format).

When compared with one GCD, the newly released H100 already surpasses the MI200 in peak performance with FP64, and at lower precisions this superiority is large. It should be noted that the peak performance of one GCD MI250X for the FP64 format is very close to that of the H100-PCIe (see Table 20). A systematic comparison of these GPUs based on the actual achieved performance during the preparation of the review was impossible due to the lack of relevant publications, and in practical terms, a comparison of the H100/GH200 with the expected MI300 may become relevant in the near future.

There are other execution units in CDNA 2 that are relevant to working with images and video, but they are not discussed here — for example, GCD has two VCN blocks for machine learning tasks in working with images and video (see Figure 18).

Now we should turn to the memory hierarchy — the second main component that determines GPU performance. The most detailed data on the entire MI200 memory hierarchy is presented in [**299**]. The L1 cache here has a structure traditional for AMD GPUs, including not only division into an instruction cache and data cache, but also division of the latter into a vector cache (for working with a vector register file) and a scalar cache. And it is impossible to directly compare the capacity of the L1 D-cache in CDNA 2 with the capacity of such a cache in the A100 due to the fact that CDNA 2 also has a separate LDS shared memory with a capacity of 64 KB per CU [**41**]. And in V100 and A100, the corresponding cache (the size of which is given above in Table 21) is unified, and includes L1 and shared memory.

The L2 cache located in the GCD has a capacity of 8 MB and is a 16-way set-associative cache. The L2 cache in GCD has a total data transfer rate with lower level caches of 4096 KB per clock [299]. More precisely, the L2 cache in GCD consists of 32 slices, each of which transfers 128 bytes/cycle, or a total of 6.96 TB/s [41]. Through the 3rd generation Infinity Fabric interconnect, L2 cache data can communicate with HBM2E memory at a speed of 2 KB per clock. Attached to a single GCD, the HBM2E memory has a capacity of 64 GB with a bandwidth of 1.6 TB/s [299].

Models MI250 and MI250X each have two GCDs. Accordingly, at the level of the entire model, the indicators listed in the previous paragraph are doubled, as shown in Table 21. And Infinity Fabric in CDNA 2 began to ensure cache coherence between GCDs, and the HBM2E memory, as well as the L2 cache of both GCDs, are shared [299]. If the mapping to the A100 is done relative to a single GCD, then the capacity and throughput of HBM2E in CDNA 2 is higher than that of the A100-40GB. But Nvidia then began producing A100 models with double the capacity of the HBM2E (80GB), which have more capacity and bandwidth than the single GCD in the MI250/MI250X.

Comparing the listed memory indicators in Table 21, we can say that they are comparable in different AMD and Nvidia GPU models (MI100 and V100, MI200 and A100; H100 is not taken into account here). However, the striking difference is in the capacity of the L1 and L2 caches: in the A100 they are much larger than in the GCD, which can have an important impact on the performance achieved. The MI250/MI250X L2 cache capacity per GCD remains the same as the MI100 (8 MB), slightly larger than the V100 (6 MB) [185]. But in the A100 this capacity was sharply increased and became several times larger than that of the MI200. As for the L1 cache, in the MI100 its capacity was only 16 KB (plus 64 KB LDS) on the CU [262] versus 128 KB on the SM in the V100 (where this memory is partially used as shared memory) [185]. The MI200 CU also has the same L1 cache capacity as the MI100 (and also has a separate LDS memory).

But the capacity of the L1 cache is 8 times less than that of the unified V100 cache, and additional LDS may not compensate for this — even after allocating part of the capacity of the unified V100 cache to shared memory, the remaining capacity for the L1 cache may be much larger than in MI100 and MI200. This has already caused performance problems (see Subsection 5.3.2.2 on MI250/MI250X performance further on this), so compared to the A100, the MI200 cache is potentially bottlenecked place.

In any case, the performance publications discussed below have already noted the shortcomings of the cache memory in the MI100 and MI200. It is possible that to optimize applications for MI200, it is advisable to use not a simple roofline model, but an empirical one taking into account cache memory.

If we compare the TDP (see Table 21), then for the MI200 GPUs compared (to the A100), these indicators are also quite close, although the TDP of the A100-SXM models is higher. It should also be noted that the numbers indicated in this table in the TDP line for AMD GPUs refer, as indicated in documents [300–302], to TBP (Total Board Power).

In general, everything already discussed above suggests that at the hardware level, the performance indicators of the MI100 are comparable to the V100, and those of the MI200 (per GCD) are basically comparable to the A100, which makes it interesting and relevant to compare the corresponding actually achieved indicators of their productivity.

The last thing to discuss in this section is PCIe support in the MI100 and MI200 GPUs. Previously, this bus was used for communication between the device and the host, and its limited bandwidth could become a bottleneck for GPU performance. When multi-GPU servers began to be offered, this bus could also be used for communication between GPUs, which would also become a bottleneck in the server. In the GPUs discussed in this section of the review, they all have a special high-speed interconnect for communication between GPUs (Infinity Fabric for AMD, NVLink for NVidia). But to communicate between a host and a device, the use of such a specialized interconnect requires its hardware support by the host processor.

This has always been done on AMD GPUs since Infinity Fabric was originally focused on communication between AMD CPUs. This is not the case for NVLink — and required the use of PCIe to communicate with conventional x86-64 server processors. Therefore, the advantage of IBM Power9 was its support for an interface with NVLink for communication with the V100, thanks to which Power9 is also used in such famous supercomputers as Summit and Sierra. All GPUs compared in the Table 21 (except H100) have support for PCIe-v4 $\times 16$ with a bandwidth of 64 GB/s.

But here there is a fundamental difference between these GPUs from AMD and from Nvidia: PCIe in AMD GPUs is not used for communication with the EPYC CPU, but is a commonly used PCIe interface to which network cards are connected, for example, for communication between cluster nodes. The use of Infinity Fabric in AMD GPUs gives them an advantage in connection bandwidth to the CPU, including compared to the A100-PCIe. And the PCIe interface available in the MI100/MI200 is aimed at using it in conjunction with MPI versions that support these GPUs for exchanging messages between servers without involving the CPU — this is used, for example, in the SLATE dense linear algebra library, focused on similar systems with distributed memory [309].

The consideration of GPU interconnects in this review is carried out in sections devoted to GPU-based computing systems and is considered as additional special hardware for the GPU, as well as possibly special processors that support the corresponding interface. Infinity Fabric is therefore discussed in the next section.

#### 5.1.2. *Computing systems based on MI200*

First of all, an analysis of the Infinity Fabric interconnect is required here. It was originally close to PCIe and is currently used for communication between EPYC processors in servers. Like NVLink, new versions of Infinity Fabric appeared as development progressed; in CDNA 2 this is already the third generation [299].

The MI200 has different Infinity Fabric connection options, selected for optimal work in different physical conditions (for example, for short connections between GCDs within the same GPU, a special interconnect is used). One Infinity Fabric channel is 16 bits wide in one direction (this was also the case in MI100). The bandwidth of one such channel in CDNA 2 is 50 or 100 GB/s in one or two directions, and between two GCDs in one GPU 4 channels are used with a total bandwidth of 400 GB/s [299] (see Figure 19 with communication topology in the nodes of the LUMI supercomputer [41]).

For communication between GCDs from different GPUs, a multi-GPU server uses one or two Infinity Fabric channels with bidirectional bandwidth of 100 or 200 GB/s, respectively. Finally, to communicate the MI250X with EPYC CPU, uses another Infinity Fabric variant with 72 GB/s bidirectional bandwidth to communicate with each GCD; In addition, host interconnects such as the Cray Slingshot-11 [41, 305] (see Figure 19).

The MI200-containing servers themselves may use a single or dual EPYC CPU option, and different interconnect topologies are possible there [305]. Figure 19 [41] shows the topology for a server with 4 MI250X and one EPYC — this corresponds to the LUMI [41] and Frontier [303] supercomputer nodes used.

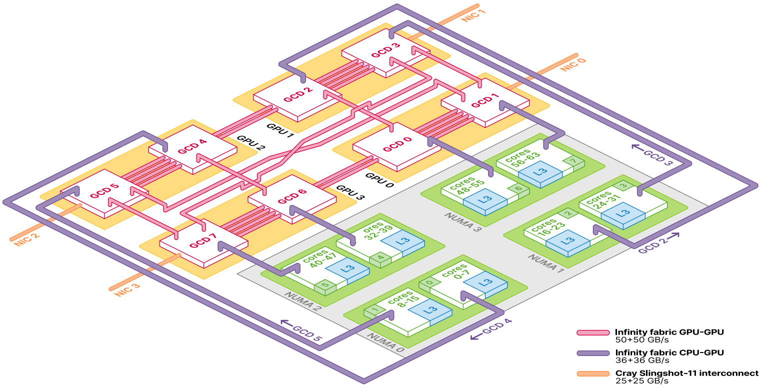


Figure 19. Using Infinity Fabric in a LUMI supercomputer node with 4 MI250X (Figure from [41])

Figure 19 corresponds to the flagship node topology for HPC; for it, [299] reports a total Infinity Fabric bandwidth of 1.54 TB/s. [305] presents another server topology with 4 MI250s coupled to two EPYC CPUs via a PCIe switch as a "mainstream" for HPC and AI, and illustrates another topology for a flagship server for AI with 8 MI250s and two EPYCs.

The MI210, which has one GCD, has its own modifications to work with Infinity Fabric. The MI210 provides 64 GB/s of bandwidth to the CPU without the need for PCIe switches, and the MI210-MI210 interconnect can use 3 Infinity Fabric lanes for a total of 300 GB/s of bandwidth. Finally, for a multi-GPU server with MI210, flagship and “mainstream” options are also offered [305].

Different topologies can also occur in Nvidia multi-GPU servers. However, the presence of different types of Infinity Fabric connections can complicate the optimization of highly scalable programming in multi-GPU servers with MI200. However, in [308], which examined various Infinity Fabric interconnect options on a server with 4 MI250X GPUs, no significant NUMA effects were identified on the CPU-GPU communication bandwidth at the HIP API level.

The Frontier supercomputer nodes use one 64-core EPYC 7A53 processor (operating at 2 GHz) with 4 MI250X. This CPU, codenamed Trento, was created specifically to working with the MI250X in Frontier nodes. The L3 cache capacity of this CPU is 32 MB for each of the eight 8-core groups (256 MB total per CPU). The CPU is configured as 4 NUMA nodes, each of which is connected to 128 GB of memory, so its capacity in the server is 512 GB (8 channels of DDR4-3200) [41, 299, 303]. The possible topology for using Infinity Fabric in multi-GPU servers with MI250X was discussed above (see Figure 19); [41] points out the importance of proper NUMA node binding to the GPU, which can be critical to optimizing application performance. Frontier was built by Cray/HPE and uses the Slingshot-11 interconnect to connect the server nodes [299].

The third-place Top500 supercomputer, LUMI, uses the same hardware as Frontier [41, 42]. Here we must also point out the Crusher system, which is actively used in research, containing 192 of the same nodes as Frontier (9408 nodes) and LUMI (2560 nodes), and is used for testing and development. And even earlier, the predecessor of Frontier was considered the Spock cluster, which had 36 nodes containing by 4 MI100 [310] (see Table 23 below).

Naturally, almost all the world's leading server manufacturers supply models containing the MI200 GPU. Their use in modules of the standard OAM (OCP Accelerator Module) form factor [305] contributes to the rapid and widespread adoption of such servers. Servers are available in a variety of sizes from 1U to 4U, as well as 6U and 10U, containing from 1 to 8 GPUs (as well as 10 and 20) MI100, MI210 and MI250. These servers can have 1–2 AMD EPYC CPUs of Zen 2 and Zen 3 architectures, as well as 2nd and 3rd generation Intel Xeon Scalable processors. A list of such servers is available at [311].

To achieve maximum performance when working with the MI250/MI250X GPU, liquid cooling is required (without it, it is usually impossible to have more than 500 W). A classic example of this is the HPE Cray accelerator blade EX235a [46], used in related supercomputers.

## 5.2. SDK for MI100 and MI200

The discussion of SDKs in this section of the review is very limited, since most of the corresponding components from AMD are direct analogues of the SDKs from Nvidia described above, the names of which are also often very similar. AMD's SDK for the GPUs in question is called ROCm (Radeon Open Compute, and m stands for "multi-GPU computing" at the heart of these tools [312]). One advantage of AMD's SDK is that it is available in source code [312], while Nvidia's SDK is simply freely available.

If in 2020, when MI100 appeared, version ROCm 4.0 was available [312], then by mid-2023, before the appearance of the June Top500 list, version ROCm 5.5.1 was already available [313]. If the basis of NVidia's SDK should be considered the CUDA API, then the basis of AMD is the HIP API[58]. HIP as a programming model is extremely close to CUDA, and similar terms are used there (see Table 1). In HIP, when working with AMD GPUs, a warp (wavefront in AMD terminology) is used with a size of 64 threads [313], and not 32 (as for SM in Nvidia), which is related to the CU architecture.

But programs from Nvidia GPUs began to be portabled to AMD GPUs regardless of the appearance of the MI100. Thus, information on the achieved performance when porting various software systems from a popular area of application of modern GPUs, classical molecular dynamics, is presented in [314].

As an advantage of HIP over CUDA, we can point out the possibility of using HIP when working on Nvidia GPUs, which makes it possible to port software for AMD GPUs to Nvidia hardware. But here two important clarifying points arise. Firstly, we need data on the comparison of the performance achieved when running HIP relative to that achieved using CUDA (this will be discussed below). Since HIP uses certain CUDA backends when working with Nvidia GPUs, this eliminates possible problems.

It is generally believed that using HIP causes almost no performance degradation compared to direct CUDA encoding [42]. Data actually demonstrating this for one of the algorithms for solving a CFD problem with an unstructured mesh are available in [168]. But the new version of CUDA for H100 has extensions, the most striking of which can be considered a change in the hierarchy of the used sets of threads (a cluster of thread blocks has appeared). Accordingly, potential HIP performance issues for Nvidia GPUs may remain.

The ROCm software stack at the lowest level includes drivers. Currently, ROCm support is provided on Linux (drivers are included in the kernel module, which can be installed on Ubuntu, RHEL and SLES distributions). It is clear that ROCm includes all the typical components for an SDK. There is a compiler, now called ROCmCC (with support for HIP, OpenMP and OpenCL), which is based on Clang/LLVM [313]. Naturally, there is a profiler, debugger and performance tracing library.

Math libraries come with the prefix roc or hip. The roc prefix is used in the names of libraries optimized for AMD GPUs, and the hip prefix is used for libraries where tools for Nvidia GPUs are used as a back-end [313]. Given the prefix the full names of these libraries make their purpose obvious and are not given here. There are also communication libraries similar to the corresponding tools from Nvidia. For example, Nvidia's ROCm counterpart to NCCL is RCCL.

Even due to the syntactic similarity between HIP and CUDA, it seems natural that ROCm would have a conversion tool from CUDA to HIP, HIPIFY [313]. But in the general case, this should be considered as the first semi-automatic step towards such a transformation, which may require further manual modification, if only because not all CUDA APIs are supported in HIP (see, for example, [42]). Here we should point out a certain analogy between HIPIFY and Intel DPCT tools.

It is important to point out here the open source hipSYCL project (now called *AdaptiveCpp*) at the University of Heidelberg (Germany), an implementation of SYCL targeting CPUs and GPUs from different manufacturers. And in [165] an attempt has already been made to implement oneAPI without the DPC++ compiler, using hipSYCL. A comparison of the performance of MI250X using different implementations of SYCL, DPC++, as well as HIP and OpenMP on different applications is carried out in [176].

Another important project for working with AMD GPUs is the GPUFORT project [315]. These tools transform one source code into another source code. There are two possible options for such transformation of the source text. First, from CUDA Fortran (or possibly OpenACC) to a Fortran variant with OpenMP version 4.5 directives. The resulting text can then be compiled using AOMP (a Clang/LLVM-based compiler with a Fortran front end). Secondly, it is possible to use HIPFORT tools, which provide a Fortran and HIP runtime interface and access to math libraries. If compiled to run a program on an Nvidia GPU, HIPFORT provides an interface to the CUDA runtime tools and associated math libraries [316].

As for Fortran, it should be noted that there is support for the AMD GPUs under consideration outside of ROCm — in the famous HPE Cray Fortran, which also has support for OpenACC, which is missing in ROCm.

It is clear that the operation of the more traditional parallelization tools MPI and SHMEM is also ensured. In [317], the performance of a number of MPI variants that support working with CDNA and CDNA 2 — OpenMPI, Cray MPICH, MVAPICH2-GDR (as well as RCCL tools) was studied, including when using the Cray Slingshot-10 interconnect in the famous Spock cluster (it has nodes with 64-core EPYC 7662 and four MI100).

We're not talking about the AI frameworks included with ROCm here, since they're AI-specific and the review is primarily focused on HPC — but ROCm is naturally integrated with the major frameworks [313].

In conclusion of this section, it can be noted that both the hardware and software of the GPU MI100 and MI200 do not show such a strong focus primarily on AI (and on HPC — rather secondary) as in the GPU H100 (and partly in the A100). At Nvidia, this it shows up not only in documentation, but also in the new supercomputers that are emerging: unlike the more "traditionally oriented" Frontier with AMD MI250X, the H100 is used in supercomputers from the Top500, focused on AI. Nvidia produces AI-ready, high-performance computing systems up to the supercomputing level.

## 5.3. MI100 and MI200 performance data in benchmarks and applications

By the time the June Top500 list appeared, many articles had become available that examined the performance of these GPUs in different benchmarks and on different applications, including performance comparisons with respect to the V100 and A100. In most cases, when there were sufficient publications for the specific AMD GPU models under consideration, this review prioritized HPC-related works for analysis (but many data on AI are also provided). Among these publications, we also selected articles related to performance data on widespread and relevant mathematics problems. From publications on applications in this review, traditional areas known in HPC, for example, computational chemistry or CFD problems, were also selected, with priority given to world-known applications.

But here it is necessary to highlight the work on the ECP project and the summary data of a review nature on more than 20 applications (more precisely, projects) specifically focused on exascale — for example, [262]. There is a large number of performance comparison data for the MI100, MI250X, V100 and A100. Since performance estimates in the limiting case are interesting for a specific application or at least an application in the relevant domain, in many cases it is useful to look at the data in [262], where performance is also presented at the single GPU level.

However, it must be borne in mind that the ECP does not mainly use globally used applications, but their special versions (possibly parts) made to focus on exascale, or new developments. And this may be completely ineffective and does not provide accurate estimates for performance in more typical applications, for small computing systems, or for calculations of not so complex objects. In addition, this is just the initial data from ongoing work on various projects using early versions of ROCm in AMD GPUs, and the results will clearly be improved (some clarifying articles related to specific projects have already appeared). Accordingly, a very large number of criticisms appeared here regarding many ROCm components of different versions from 4.2.0 to 4.5.0, which AMD tried to quickly fix

Since much of the following data on the performance of the MI100 and MI200 GPUs was obtained using well-known supercomputers (including those from among the leaders in the Top500), brief information about such computing systems (including those used for comparison with Nvidia GPUs) is summarized in Table 23.

### 5.3.1. *Performance of MI100*

Comparative performance data of the MI100 compared to the V100 and A100 will be discussed here (everything relative to the MI200 is discussed below).

As for comparing the performance of GPU MI100 and A100, although the data in Table 20 shows slight advantages of MI100 in terms of peak performance (for example, with FP64), limited attention should be paid to these indicators for GPUs — no less often performance is correlated with memory bandwidth, and more precisely, more important is to look at the roofline model data. The available data from articles generally clearly indicate large (often several times) performance advantages of the A100 relative to the MI100, although there are exceptions.

Table 23. Computing systems whose nodes were actively used in publications on GPUs performance cited in this review; for supercomputers from the Top500, their number in the list is given in parentheses

| Computing system | Number and type of GPUs in the node | Number and type of CPUs in the node | Interconnect (between nodes)[1] |
|---|---|---|---|
| Frontier (1) [303] | 4×MI250X | 1×EPYC 7A53(Trento) | 4×HPE Slingshot-11 |
| LUMI (3) [318] | 4×MI250X | 1×EPYC 7A53(Trento) | 4×HPE Slingshot-11 |
| Crusher [319] | 4×MI250X | 1×EPYC 7A53(Trento) | 4×HPE Slingshot-11 |
| Spock [320] | 4×MI100 | 1×EPYC 7662(Rome) | 1×HPE Slingshot-10 |
| Leonardo (4) [205] | 4×A100-40GB | 1×Xeon Platinum 8358 | 4×Nvidia Infiniband HDR |
| Perlmutter (8) [206] | 4×A100-40GB | 1×EPYC 7763 | 4×HPE Slingshot-11 |
| ThetaGPU [321] | 8×A100-40GB | 2×EPYC (Rome) | 8×Infiniband HDR |
| Polaris (19) [322] | 4×A100-40GB | 1×EPYC 7543P (Milan) | Slingshot-10 |
| Summit (5) [323] | 6×V100-16GB | 2×Power9 | 2×Mellanox Infiniband EDR |

[1] The interconnects used in the publications cited in the review are indicated.

Therefore, we will present here only a few confirmations of this, mainly in an integral sense (with a wide scope of applications) — and not based on individual specific examples. For example, in [45] data corresponding to the above are presented on 6 different applications and mini-applications (from different fields of science) included in the ECP project or related mini-applications. According to data from [45] we calculated how many times the achieved performance of A100 was higher than MI100: for AMR-Wind (CFD part of the ExaWind, Exascale Predictive Wind Plant Flow Physics Modeling project) — on three different kernels in 1.7-5.3 times; in the quantum chemical mini-application GAMESS RI-MP2 — 5.8 times; in the TestSNAP mini-application for the SNAP quantum potential, which is then used in the LAMMPS molecular dynamics application (from the EXAALT project for nuclear fusion problems) on three different kernels — 2.2-7.9 times, and so on.

As another example of performance data covering even more projects and applications from ECP, we point out slightly more recent publication [262] (see also [324]); comments about [262] are given above. Our calculations of relative performance based on data from [262] show that the A100 was about two times faster on CFD problems (NekRS), and three times faster

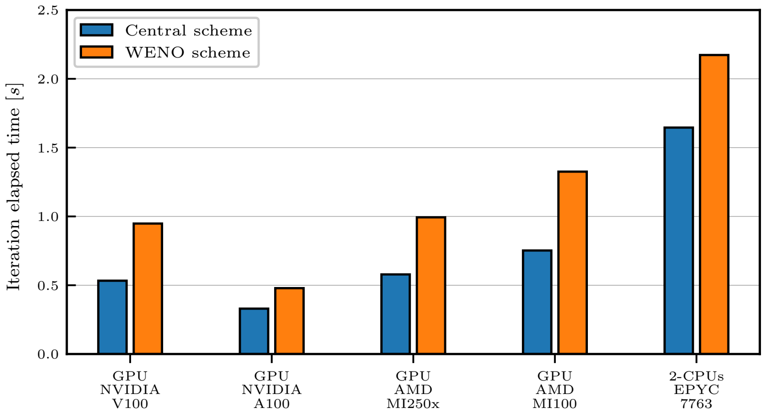


FIGURE 20. Time per iteration (seconds) for STREAmS-2 with a $420 \times 250 \times 320$ grid for two calculation schemes: the central flow estimation scheme and the WENO scheme (figure from [325])

on quantum chemistry (Gamess, Fock matrix construction) and molecular dynamics problems (LAMMPS). For TestSNAP, the A100 was 2.8 times faster than the MI100.

A newer version of ROCm 4.5.2 on another CFD application, STREAmS-2 [325], also produced similar MI100 performance lag values (see Figure 20) [325]. These data are discussed in more detail below in english-sec:5.3.2.2]Subsection 5.3.2.2. It is unlikely that progress in new versions of ROCm can completely eliminate such a lag.

In [225] performance using MI100, A100, and V100 was examined on SPEChpc 2021, which contains components from applications in different HPC fields, but the data for MI100 was obtained in a different "small" suite of SPEChpc than was not used for other GPUs.

Application performance is the most practically relevant, but corresponding data on the performance of MI100 relative to A100, showing similar data above, is also available for specific mathematical methods—for example, for FFT [251], for QR decomposition of square matrices using the MAGMA library for single and double accuracy [241], for BabelStream bandwidth benchmarks [42]. There is also other data on BabelStream benchmarks for their implementation using DO CONCURRENT Fortran tools [236], where the memory bandwidth of MI100 was slightly different from the C++ version of BabelStream (sometimes Fortran gave better results). The bandwidth for both A100-80GB and A100-40GB in [236], was naturally found to be much greater than that of MI100.

Modern GPUs, their SDKs, and the applications that use them are complex applications require different parameters for optimization for different GPUs, and there are exceptions to all rules. Thus, in [241] data on the same QR decomposition with rocBLAS show the advantage of MI100 over cuBLAS with A100. And in [326] within the AnySeq/GPU code for bioinformatics tasks using FP32 cores in MI100, V100 and A100, the performance (in trillions cell updates per second, TCUPS) was 3.8, 1.7 and 3.3 TCUPS, respectively. But in general, in publications, the strong lag between the MI100 and the A100 in terms of performance is shown quite clearly.

But comparing the performance of MI100 with V100 seems more relevant, including as a possible addition or basis for comparing A100 and MI200. Here the achieved performance is indeed quite comparable, and the peak FP64 performance of the MI100 is still noticeably higher than that of the V100. This gives the MI100 a chance to achieve higher performance for compute-intensive applications (in roofline model terminology). Let us immediately give a striking example of this in [262], where, using the quantum chemical application NWChemEX in calculations with the computationally intensive method of coupled clusters (CCSD), a comparison of the performance of Spock nodes (4 MI100 each) and Summit (6 V100 each) showed the advantage of Spock. NWChemEx used CUDA on Summit and HIP (ROCm 4.3.0) on Spock.

The acceleration values of Spock relative to Summit that we calculated (according to Table 11 in [262]) show that on one Spock node the total time of CCSD calculation is 1.5 times less than on the Summit node, and the time of an additional more complex calculation of the correction T (for triple excitations) are 1.9 times fewer than on the Summit node.

As the number of nodes increases to 4, the performance advantage of Spock at an equal number of nodes quickly decreases, and at 4 nodes Spock is only faster in terms of computation time of T [262]. It must be said that probably most modern GPU applications are more memory-bound, and the V100 is less behind the MI100 in memory bandwidth.

The article [262] provides comparative data on the performance of deep learning for MI100 and V100 within the framework of two ECP projects — CANDLE (precision medicine for oncology) and ExaLearn (the goals of the project are clear from its name). In CANDLE, BERT performance is moderately greater on the MI100 than on the V100, but not to the extent that the relative performance of these GPUs would indicate (obviously in [262] referring to peak hardware performance). And for ExaLearn, [262] indicates a much lower performance of the MI100.

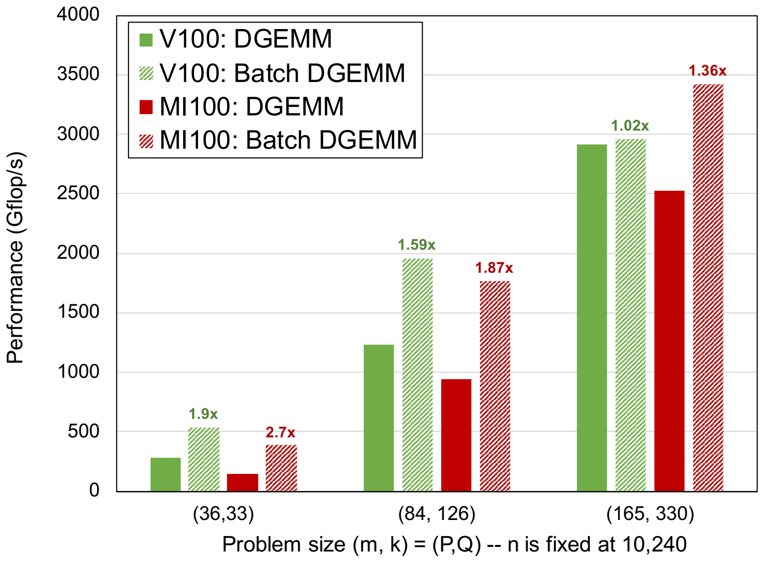


FIGURE 21. Performance of MI100 and V100 on DGEMM (Figure from [242])

It is important that in addition to higher peak performance, MI100 showed higher memory bandwidth than V100 on all components of the BabelStream test using a wide range of different programming models [42]. In the same work, the performance of the miniBUDE (molecular docking) application on MI100 turned out to be higher than on V100-32GB and almost coincided with the performance of A100-40GB.

A very relevant comparison is the performance of MI100 and V100 on DGEMM, which was carried out in [242] for batch and conventional implementation of matrix multiplication. The corresponding data is presented in Figure 21 using ROCm 4.2 (hipBLAS) for MI100 and CUDA 11.2 (cuBLAS) for V100. The parameters m,n,k on the abscissa here correspond to the dimensions of the matrices $M$, $N$, $K$ in formula (1). It can be seen that MI100 can be ahead of V100. [242] also provides data for CEED benchmarks from ECP, showing comparable performance between MI100 and V100, and performance on NekRS, where MI100 was 15% behind V100. The same 15% lag is indicated in [265].

In [327] calculations were carried out using the QUICK quantum chemical application in combination with the AMBER molecular dynamics application (according to the QM/MM scheme). In the starting tests, the MI100 gave performance two times less than the V100, due to the excessive use of registers in AMD kernels. After improvements in QUICK, in the calculations of large-sized systems, the performance of MI100 and V100 turned out to be comparable (on small systems, V100 was much faster). The kernel, which calculates the gradient of the exchange-correlation potential, calculated much faster on the MI100 than on the V100 [327].

ECP-related work [328] compared the performance of MI100 (on Spock) and V100 (on Summit) on the well-known CFD benchmark HipBone (which is GPU-focused, based on the well-known C++-based NekBone benchmark, but does not cover all its functionality [142]), and compared the performance at different degrees of the polynomial of the kernel of the Poisson operator, which actually determines the performance of HipBone. HipBone uses FP64, but some preprocessing subtasks use FP32.

On one GPU, the performance of MI100 (with ROCm v4.5.0) obtained in [328] is approximately the same as the performance of V100 (with CUDA v11.0.3) at all degrees of the polynomial — sometimes MI100 is faster, and sometimes V100. The highest performance is achieved with the highest polynomial degree used (N=15): for V100 — 2101.4 GFLOPS, for MI100 — 2135.2 GFLOPS; good scalability was also shown on Spock and Summit nodes with the number of MPI ranks up to 32 and 48, respectively [328].

In report [262] based on data for some ECP projects, the comparability of the performance of MI100 and V100 is noted — this is said, for example, for AMR-Wind data (our estimate of the acceleration of V100 relative to MI100 according to Figure 20 in [262] is 1.4 times).

In the ExaSMR project (simulation of a small modular reactor core by combining two parts, neutron physics and CFD — NekRS is used for the latter), the Shift code with HIP (ROCm 4.2) for the first part was at first inferior to V100 with CUDA by a factor of two, but after optimization the lag was only 20%. In the NekRS part of ExaSMR, in some kernels, the MI100 was only a few percent behind the V100 on Summit [262]; our calculation according to tables 42 and 43 from [262] gives an acceleration of V100 relative to MI100 by 10–20%.

In other software components of other ECP projects, [262] indicated a more severe performance lag in MI100. On the QMCPACK application (quantum Monte Carlo calculations), nickel oxide supercell calculations achieved very different lags in different cases, but overall the MI100 gave significantly lower performance, and AMD compilers were criticized. In TestSNAP, MI100 in Spock lags behind V100 in Summit by approximately 2 times (our calculation using Table 19 in [262]). In the ECP project with the Gamess quantum chemical application, for ock matrix calculation in different basis types with lengths from 4 to 12 thousand orbitals, MI100 is 1.4-2.6 times slower (our calculation according to Table 15 in [**262**]). Naturally, there is a lot of other data available in [**262**] comparing the performance of V100 and MI100.

Authors of [**329**] compared the performance of MI100 and V100 using mini-application, MiniMDock for molecular docking. In variants optimal for GPU performance (with HIP and CUDA), MiniMDock, when using medium and large ligands, MI100 (Spock, ROCm 4.5) was inferior in performance to V100 (Summit, NVHPC 21.11) by 32 and 39%, respectively, but on small ligands MI100 was slightly faster than V100.

Authors of [**330**] optimized some critical important kernels for the CFD application FUN3D, whose performance on the A100 was discussed above. For all of them except the viscous flow kernel, MI100 (with ROCm 4.2.0) was 18–53% slower than V100 (with CUDA 11.2).

Naturally, MI100 is used in a variety of areas, not only in HPC, but also for AI, for example, for deep learning in the DLRM model [**331**].

We can say that the MI100 is comparable in performance to the V100, although the MI100 often lagged behind. Much of the MI100 performance data reported above refers to preliminary results obtained shortly after the MI100 became available, and these results may still be quite significantly improved on new versions of ROCm (CUDA versions seem much more stable). But due to the clear advantages of all indicators of MI210 relative to MI100, not only those indicated in Table 20 and Table 21, but also all others known to the author at the time of writing the review, interest in MI100, in the author's opinion, relates to the use of MI100 as already purchased, and the acquisition of a new MI100 is not practical compared to MI210. Therefore, a more complete review of the available performance data for the MI100 is not provided here.

### 5.3.2. *Performance of MI200*

Looking at the data in Table 20 and Table 21, it is clear that on a per-GCD basis, the MI210 and MI250 models are almost the same, but the GCD in the MI250X has a slight increase in the number of CUs, which should provide a slight increase in performance. Therefore, although we will begin the analysis of the achieved performance with a number of current articles known to the author (at the time of preparation of the review) with data on the MI210, this may also be of interest for evaluations of two others models of the MI200 family.

**5.3.2.1. Performance of MI210.** As a basic indicator of the achieved performance of the MI210, we first point out the data Dell presented for its PowerEdge R750x servers — for double and single precision matrix multiplication (DGEMM and SGEMM) [332], including in comparison with data for the MI100. Dell's calculations on the MI210 used tensor (matrix cores and ROCm 5.1.3. A performance of 28.3 TFLOPS was achieved for DGEMM, and 32.2 TFLOPS for SGEMM. These data [332] indicate a large acceleration in MI210 relative to MI100 for DGEMM — 3.6 times (but in MI100 there is no FP64 support in tensor cores), and a low achieved percentage of the peak performance in MI210 (62.5%).

For SGEMM on MI210, performance was 9% higher than on MI100 [332]. But FP32 is supported in the MI100 tensor cores, which gives the MI100 even slightly higher peak single-precision performance than the MI210 (see Table 20 and Table 21).

In the HPL benchmark, the performance of MI210 reaches 18.2 TFLOPS per 1 GPU and grows almost linearly up to 4 GPUs (up to 72.6 TFLOPS), while the performance of MI100 per 1 GPU is several times less and grows more slowly with the number of GPUs in the server [332].

For HPL, there is also data from AMD itself, which allows us to demonstrate the possible impact of the ROCm version [333]. Here, using the newer ROCm 5.2.0 and rocHPL 6.0.0 on a single GPU, the HPL benchmark gave 21.07 TFLOPS (1.16 times faster than Dell), on 4 GPUs — 81.097 TFLOPS (1.12 times faster), on 8 GPUs — 159.73 TFLOPS. As noted in [333], performance may also vary depending on the driver version. We assume that the influence of differences in other components of the servers used in the benchmarks in [333] and [332] here is negligible.

On the HPL-AI benchmark with mixed precision FP16/FP32/FP64 on 4 GPUs in [333], a performance of 444.77 TFLOPS was obtained.

Dell [332] provides data on the performance of MI210 relative to MI100 in the LAMMPS molecular dynamics application for its several different types (including reactive molecular dynamics) and potentials — when working with FP64, MI210 is 18–30% faster, and with FP32 — by 12%. Data on the performance of LAMMPS for reactive molecular dynamics are also reported by AMD using a more newer version of ROCm [333].

The first articles with MI210 performance data began to appear for applications in various fields. For example, [334] provides performance data on the Uni-Dock molecular docking application developed in this paper, which is superior in performance to the well-known AutoDock-GPU application (although detailed Uni-Dock performance data is provided in [334] for V100 and MI100).

Among the works that compare the performance of MI210 and Nvidia GPUs, we point out CFD data [**268**] for the FUN3D application mentioned above in the section on A100 performance with its FLUDA library for GPUs. Based on the data in Table 4 in [**268**] we find that the A100-40GB is 1.44 times faster than the MI210 (and the MI210 is 1.29 times faster than the V100-16GB).

In [**335**], the performance of MI210 and A100-40GB was compared when running the PyTorch deep learning framework (PyTorch 2.0-20230102 was used) using ROCm 5.4.2 and CUDA 11.8 stacks. This article proposed TorchBench, a new set of benchmarks for the PyTorch software stack. In comparison, the classic MLPerf deep learning benchmark suite, whose GPU results were discussed earlier in this review, includes 8 deep learning models, while TorchBench has 48. TorchBench is open source and, thanks to its wide variety of representative models, allows for e.g. and identify situations with decreased performance when running PyTorch on a GPU.

Comparison made in TorchBench's 32-bit (default) configuration. The A100's TF32 support gives increased performance at the cost of reduced precision, so not all models can use it. For deep learning and inference of a trained neural network with FP32, the MI210 was faster in some models and the A100 in others, which does not give a clear performance advantage to one of these GPUs [**335**].

In general, MI210 immediately began to be used for tasks related to AI. For example, [**336**] developed a DLRM-oriented technique for overlapping communication and computation and created a kernel for MI210. And in [**337**], an improved algorithm was proposed and implemented on MI210 to improve the performance of a convolutional neural network (CNN), which, using the MIOpen deep learning library and ResNet50 source data, allowed to obtain 74% of the peak single-precision performance of MI210.

**5.3.2.2. Performance of MI250/MI250X.** As for the MI250/MI250X GPUs, each containing 2 GCDs, their potential competition with the A100 in terms of performance was demonstrated by the developers themselves. In June 2022, Nvidia cited data demonstrating the superior performance (and power efficiency) of the A100-SXM-80G relative to the MI250 in single- and quad-GPU servers (counting the MI250 as one GPU) on classic molecular dynamics applications AMBER, GROMACS, LAMMPS, NAMD, and OpenMM [**338**], and in August 2022, AMD provided opposite performance data for all of these applications except GROMACS — but instead of showing data for this program, it showed a performance advantage on the HPCG benchmark [**339**]. It is clear that such a comparison requires additional data about the versions of applications and SDKs of the companies used, those used in the calculations of molecular systems and types of potentials, and so on. The performance analysis of the MI250/MI250X below also includes published data on molecular dynamics performance.

That the performance of the MI250X is high is evident from the success of AMD GPUs in the Top500, where Frontier and LUMI supercomputers take first and third places with HPL performance of 1194.0 and 309.1 PFLOPS, respectively, and HPCG performance of 14.1 and 3.4 PFLOPS respectively [4]. A fairly broad comparison of the performance of 8 different applications from different areas of HPC (mostly not from those widely used in the HPC world) on Frontier and Summit was carried out in [**340**], where showing the advantages of Frontier, a general starting comparative assessment of the performance of MI250X and V100 is given.

But since such successes on supercomputers are largely associated with a high level of scaling with the number of nodes, we present here just one more illustration based on a comparison of the performance of nodes with the A100 and with the MI250X — in the Perlmutter supercomputer and the Crusher cluster (brief data on these computing systems is available in Table 23). Their nodes contain 4 GPUs A100 and MI250X, respectively.

In [**341**] the performance of X-ray tracing code using Kokkos tools (as well as a version with CUDA) was studied on different numbers of Perlmutter and Crusher nodes (there, CUDA and HIP were used as back ends, respectively). Most of the computing time on the GPU there is taken up by the nanoBraggSpots kernel; Using Kokkos, nanoBraggSpots achieved over 60% performance improvement per Crusher node compared to Perlmutter [**341**]. The data in Figure 2 of this article shows that the performance of nanoBraggSpots with an equal number of nodes from 32 to 128 is several times greater on Crusher. And the use of Kokkos gave noticeably higher performance than the usual use of CUDA.

But then we need to keep in mind which comparison of Nvidia and AMD GPUs is most interesting. AMD MI250 and MI250X each have two logical GPUs (2 GCDs each), while the A100 is a single GPU. From a programming point of view, MI250 or MI250X are two GPUs, and it is interesting to compare the A100 with one GCD, which was often (but not always) done in publications. And then the question arises about costs — if they are comparable for the A100 and MI250X, it would be interesting to compare them with the whole MI250X. But here there is no comparison of cost indicators or TCO (it not only depends on where the GPU is purchased and used, but also changes over time).

**Basic benchmarks and math libraries performance tests.** The most important baseline benchmarks for the Frontier and Crusher nodes are available on the website of the US Oak Ridge National Laboratory, which owns both these supercomputers and Summit [342]. Some of this data also applies to the host CPU (EPYC 7A53). To work with MI250X (with one GCD), ROCm 5.3.0 is used there.

Article [342] provides information on the achieved bandwidth of MI250X memory, CPU-to-GPU and inter-GPU communications. It presents data on bandwidth in all five kernels of the BabelStream benchmark (classic version with FP64) using HIP and OpenCL, depending on the dimension of one-dimensional used in benchmarks arrays.

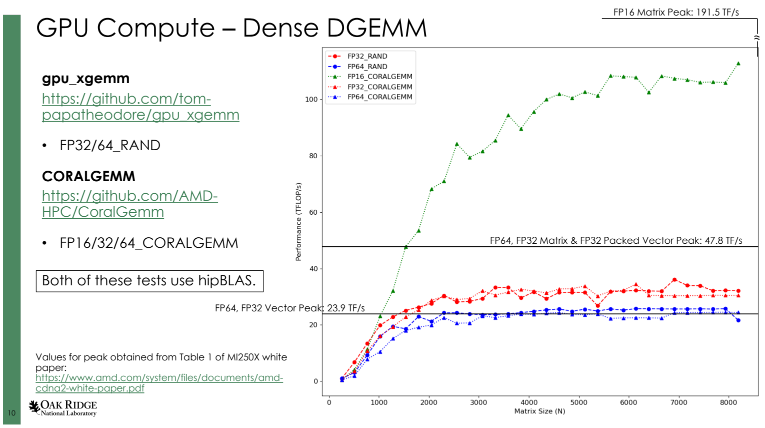
According to [342], the HBM bandwidth achieved in BabelStream is 77–86% of the peak value (82–90% is achieved for a CPU with its DRAM). The highest values were achieved here when running the copy (slightly less in mul) kernel of BabelStream using hipcc, a maximum of about 1.38 TB/s. But in some kernels of BabelStream, using OpenCL gave higher bandwidth than hipcc, and with the dot product kernel, hipcc (unlike OpenCL) gave abnormally low bandwidth.

The memory bandwidth of MI250X (one GCD) in [324] was also studied by executing SpMV from the Ginkgo library using different sparse matrix storage formats.

CPU-GPU bandwidth (measured by hipMemcpy) in [342] was 25.6 GB/s (71% of peak), and between GPUs in the osu_bw MPI test from OSU microbenchmarks [343] (using Cray MPICH 8.1.23) — from 37.6 GB/s (75.2% of peak) on one Infinity Fabric channel to 145.3 GB/s (72.7% of peak) on four Infinity Fabric channels [342]. In [342], the latest measurements are for one-way transfers—and Table 19 of that review gives a peak value of 100 GB/s per channel for two-way transfers.

Article [342] also provides data on the performance of MI250X on GEMM (using hipBLAS), including DGEMM. For GEMM with FP16, a specific test CORALGEMM [344] was used, the performance of which increased up to the highest matrix dimension of over 8 thousands. For FP32 and FP64, in addition to this test, another specific test gpu_xgemm [345] was used in [342] (see Figure 22).

The choice of one or another specific test does not give fundamental changes in performance here — it is mainly determined, of course, by the choice of FP32 or FP64. The use of hipBLAS does not give any outstanding results (such as achieving 90% or more of the peak value). But it's interesting whether the jumps in the performance curves depending on the matrix size are related to its "optimal" parity.



## GPU Compute – Dense DGEMM

**gpu_xgemm**
https://github.com/tom-papatheodore/gpu_xgemm

- FP32/64_RAND

**CORALGEMM**
https://github.com/AMD-HPC/CoralGemm

- FP16/32/64_CORALGEMM

Both of these tests use hipBLAS.

Values for peak obtained from Table 1 of MI250X white paper:
https://www.amd.com/system/files/documents/amd-cdna2-white-paper.pdf

10

Figure 22. GEMM performance on one MI250X GCD (Figure from [342])

Article [342] also presents mixbench test data, but these results, as above in the section for A100, are not considered here, since they are more interesting for AI.

Benchmarks [333] presents data on the performance of MI250 (with ROCm 5.2.0) in the HPL (with rocHPL 6.0.0), HPL-AI (with HPL-AI-1.0.0) and HPCG 3.0 benchmarks. On one GPU the achieved performance was 40.45 TFLOPS (about 90% of the peak vector value), and on 4 GPUs it was 161.97 TFLOPS. Note that the results in [333] are for full GPUs with two GCDs. On HPL-AI (module with FP16/32/64) on 4 GPUs, performance reached 930.44 TFLOPS. On HPCG, 488.8 GFLOPS were obtained for one GPU and 1927.7 TFLOPS for 4 GPUs.

Computational Chemistry. Considering that AMD and Nvidia actively compared the performance of the MI250X and A100 on molecular dynamics problems, we will begin our presentation here with this area of computational chemistry. It should be noted right away that achieving high performance when porting molecular dynamics applications from Nvidia GPUs to AMD GPUs is not an easy task. The transfer of such software packages from CUDA to HIP is discussed in [314] (although the models analyzed in this review were not considered there).

AMD reports its LAMMPS 2022 molecular dynamics application performance across several famous of this application benchmarks on the MI250 and MI210 GPUs and compares them to results on the A100 [346].

A single MI250 outperforms the A100 on all benchmarks (while the MI210 containing a single GCD tends to lag behind the A100-PCIe-80GB). AMD has also demonstrated good performance scaling of LAMMPS up to 4 GPUs per server [346].

Authors of [254] conducted a detailed and in-depth study aimed at improving the performance of LAMMPS after porting it from Nvidia GPU hardware (Summit with V100 and AFW HPC11 computing system with A100-40GB used in the article) to MI100 GPUs (in the Spock cluster) and MI250X (in the Crusher cluster). LAMMPS uses the Kokkos library to support work with different hardware. To optimize the performance (studied for the normal FP64 format) of LAMMPS on Nvidia and AMD GPUs, roofline models were used for each kernel, and performance estimates involved 6 different potentials and calculations of liquid, metallic, granular, biological and polymer systems up to size 55 million atoms.

Support for LAMMPS operation on GPUs with long-range interaction-oriented PPPM potential was implemented in [254] using FFT, for which rocFFT tools were used on AMD GPUs. When running at ReaxFF potential on a 256K atoms molecular system, about 20% (5.6 TFLOPS) of the peak FP64 performance of a single GCD in Crusher was achieved.

In [254], despite a large amount of careful and extensive research done, the obtained performance estimates are indicated as preliminary, which is probably due to the active development of ROCm noted in this article (ROCm v4.5.0 was used in this work).

The pointed out different performance behavior when changing Kokkos parameters for AMD and Nvidia GPUs. This can be taken as an illustration of the difficulty of porting software from one type of GPU to another while maintaining efficient code optimization.

In [262] the performance of the SNAP quantum potential (as part of the EXAALT ECP project), then used as part of the work with LAMMPS, is analyzed. Although more performance information was obtained here for the MI100, which the MI250X was, of course, far ahead of, the GCD MI250X alone at the very beginning of 2021 was significantly behind even the V100 in this calculation, and much more behind the A100. This was due to the lack of L1 cache capacity in the MI250X CUs — the V100 and A100 used a capacity of 96 KB per SM for SNAP, 6 times more than the MI100 and MI200 per CU (see Table 20 and Table 21).

Table 24. Performance (ns/day) of AMBER 22 on different GPUs

| Test | A100-PCIe performance[1] | MI250 performance[2] | H100-PCIe performance[1] |
|---|---|---|---|
| JAC Production NVE 4fs | 1199.22 | 1871 | 1479.32 |
| JAC Production NPT 4fs | 1194.5 | 1794 | 1424.90 |
| STMV Production NPT 4fs | 52.02 | 80.65 | 70.15 |

[1] data from [349] (data as of 03/21/2023). Used Amber 22 Update 1, AmberTools 22 Update 1, Nvidia CUDA 11.4

[2] data from [333], used ROCm 5.2.0 and Amber container 22.amd_100

Work on optimizing the SNAP kernels continues [262], but although these were rather preliminary results, further progress is expected primarily in terms of the use of Kokkos here. [262] states that SNAP's L1 cache hit rate is (for the largest SNAP kernel) 66% for the MI250X versus 90% for the V100.

In [347], AMD demonstrated good scalability of the NAMD molecular dynamics application on the famous APOA1 and STMV NVE benchmarks using up to 8 MI250.

Above, at the very beginning of sec:5.3.2.2, it was noted that there was no data provided by AMD on GROMACS performance on MI250/MI250X in 2022, which was obviously due to temporary problems with porting the GROMACS code, and was quickly corrected.

In [333], there is data on the performance of MI250 on the most famous molecular dynamics applications AMBER, GROMACS, LAMMPS and NAMD. Let us give a number of examples. For GROMACS, in the well-known STMV (Satellite Tobacco Mosaic Virus) benchmark, a performance of 34.2 ns/day was obtained on one MI250X, and 89.26 ns/day on 4 MI250Xs. For NAMD 3.0, in the standard STMV NVE benchmark, a performance of 19.87 ns/day was obtained on one MI250X, and 77.132 ns/day on four MI250Xs.

The GROMACS 2023.1 version uses SYCL as a back-end for the MI200, what has shown good scaling of the achieved performance in the STMV benchmark when using up to 8 MI250X. It has been discovered that not all the features of CDNA 2, which can provide a significant increase in GROMACS performance, are yet used by the AMD compiler [348].

For AMBER22, here are performance data on 8 known AMBER benchmarks. Thus, on the Cellulose Production NPT 4fs benchmark, a performance of 227.2 ns/day was obtained. We have compared the data from [333] with the results available in [349], and summarized these data in Table 24.

But here we need to make important clarifications about comparing the performance data of different molecular dynamics applications on different GPUs. The calculation data provided relate primarily to the molecular system being calculated (where the total number of atoms is important), which is determined by the name of the benchmark itself. But the calculation time also depends on other parameters — the potential used, the time-step (often measured in femtoseconds, fs in Table 24) and the cutoff radius of interactions. Therefore, the data provided on the achieved performance may not be comparable, and the use of the same parameters is required, which are not even always given. The author is not aware of such a coincidence of parameters in different data sources of Table 24.

It should be borne in mind that in this table the data from [**349**] was obtained with a far from new version of CUDA; AMD itself compared in [**333**] with CUDA 11.6, the old version of CUDA may not be effective enough for the H100 — in the publications cited in this review, CUDA 12.2 was also used. But these data, as well as those presented above in this review, indicate that the MI250 has real competition in molecular dynamics performance with the A100 (and maybe with the H100), perhaps even ahead in performance, and the dependence on the versions of the SDKs used is obvious.

Another article [**350**], which examined the performance of MI250, is at the intersection of molecular dynamics and AI. In [**350**], using the V100, A100 and MI250 GPUs, a fairly large number of calculations were carried out using the DeePMD-kit software package for molecular dynamics simulation using machine learning potentials (instead of using the QM/MM method for this). DeepMD-kit allows these potentials to be integrated with LAMMPS, Amber, OpenMM and GROMACS applications. Based on Table IV from [**350**], we computed the performance speedups of A100-80GB over a single MI250 GCD. For LAMMPS molecular dynamics calculations for FP64 they ranged from 6% to 2.5 times. The lack of details in [**350**] about the software versions used (for example, about the LAMMPS version ported to the AMD GPU, used for molecular dynamics calculations) allows these values to be attributed rather to preliminary estimates, but still providing some information about the performance of the MI250.

The paper [**351**] can also be partly (in a certain mathematical sense) considered of interest for molecular dynamics problems — here, to calculate the Euclidean minimum spanning tree using an algorithm improved by the authors, A100 (with NVCC 11.5) and MI250X (1 GCD, with ROCm 4.5) and Calculations were carried out on 12 different data sets. Based on the obtained performance data in Figure 6 from [**351**], we calculated the relative speedup of A100 compared to 1 GCD, it is in the range of 1.5-1.7.

Table 25. Calculation time of correction, T (in seconds) on GPUs [352]

| GPU | Programming model | | |
|---|---|---|---|
| | SYCL | CUDA | HIP |
| MI250X (1 GCD) | 17.41 | — | 15.56 |
| MI250X (2 GCD) | 8.97 | — | 8.12 |
| A100 | 18.23 | 16.14 | — |

In general, it seems that stable reliable data on comparing the performance of the MI200 with Nvidia GPUs in molecular dynamics problems is still rather absent (in the sense that new and improved indicators are appearing).

In [352], data on the achieved performance within the framework of the NWChemEX project in ECP using MI250X and A100 in Crusher and Polaris supercomputer nodes, respectively, were studied (see Table 23). NWChemEX is also focused on calculations using the high-precision quantum chemical method CCSD(T), in which the main calculation time is spent on the correction due to triple excitations (T), which is a perturbation to CCSD, requiring $O(n^7)$ calculations, where n is the dimension of the basis. The kernel for calculating this correction was implemented in different versions using HIP, CUDA and DPC++; or this, the compilers clang-16, gcc-10.3.0 and CUDA-11.4.4/ROCm-5.1.0 were used (for Polaris/Crusher, respectively), and for parallelization between nodes, Cray-mpich 8.1.16 was used.

Table 25 shows the obtained times for calculating the correction $T$ for the molecular fragment of ubiquitin (these are data from Table III in [352]) on one GPU. From these data it follows that one GCD in the MI250X was slightly faster than A100-40GB (both on HIP versus CUDA and using SYCL), and on two GCDs the performance of the MI250X increased by 1.9 times. At the same time, the performance achieved using DPC++ turned out to be quite close to that obtained using CUDA or HIP.

But in [352] on the A100 it was also found that in the SYCL implementation, the kernel for calculating $T$ places very high demands on the L1 and L2 caches, and the generated PTX code showed a large number of loads and stores into local (private in SYCL terms) memory. Adding one clang key gave a 2.5 times increase in performance. Changing another #pragma option to clang increased performance by another 20%. But this PTX code did not use FMA multiply-and-add operations, and after adding another clang key, the additional performance improvement was about 20% [352]. The results in Table 25 include optimization.

In addition, scalability up to a large number of nodes in Crusher and Polaris was demonstrated, which is highly dependent on the basis types used.

Table 26. Strong and weak performance scaling (in TFLOPS) of the Dirac operator HISQ on Perlmutter and Frontier nodes

| GPU in a node | Number of GPUs (for MI250X — number of GCDs) | Performance (strong scaling) | Performance (weak scaling) |
|---|---|---|---|
| A100-40GB | 1 | — | 1.35746 |
| | 4 | 5.06892 | 4.53759 |
| MI250X | 1 | — | 0.92974 |
| | 2 | 1.63049 | 1.51439 |
| | 4 | 3.00675 | 2.89454 |
| | 8 | 4.69513 | 5.57654 |

For strong scaling, a global lattice was used $96^4$; for weak scaling — local lattice $32^4$. The table use data from [354].

In [353], using the still "pilot" state of the LUMI-G supercomputer (LUMI subcluster with MI250X GPU in the nodes), calculations were carried out using the well-known quantum chemical application CP2K with the authors' improved methods for the exchange part of the Hartree-ock method and correlation methods with periodic boundary conditions, but using a Gaussian basis, which makes it possible to use block-by-block sparsity. Good performance scalability and parallelization efficiency up to 32 LUMI-G nodes were obtained.

**Lattice quantum chromodynamics (LQCD).** In [354], the performance of the SIMULATeQCD application for quantum chromodynamics, available on GitHub, was studied when running on one or several nodes (multi-GPU servers) of cluster systems, including Perlmutter and Frontier (see Table 26). Although the MI250X performance data here, as in many other publications, is considered preliminary, the comparison made here with the performance of the A100-40GB is interesting (although it is clear that using more modern versions of ROCm and additional optimization of the program code the performance achieved may increase significantly in the coming times).

SIMULATeQCD uses CUDA or HIP codes as a backend. [354] provides performance data using up to 256 A100s and GCDs; We will limit ourselves here to data from benchmarks of the Dirac operator HISQ with scaling within one node (see Table 26).

Although the results for the MI250X were reported as preliminary, the performance achieved was lower than what the authors expected based on the MI250X specifications. In general, here, as a first approximation, we can say that one GCD is somewhat behind the A100 in terms of performance, while the full MI250X (2 GCDs) is somewhat ahead of it.

In [355], the performance on nodes of the Big Red 200 (4 A100-40GB per node) and Crusher (4 MI250X per node) supercomputers was compared using the well-known QUDA program for lattice quantum chromodynamics. Due to poor scaling with the number of nodes in both systems, we will only point out performance data for one node — a node with an MI250X (with 8 GCDs) was 16% faster than a node with an A100. The advantage of the MI250X in [355] was attributed to the memory-bound performance of QUDA (with the MI250X has 2×the memory bandwidth over that A100 model, see Table 21).

**Computational fluid dynamics (CFD).** We begin the analysis of data on the performance of MI250X and MI250 in CFD applications with AMD data [333] on the performance of MI250 in the standard (for OpenFOAM CFD application) HPC Motorbike benchmark (662.3 sec. on one MI250 and 209.84 sec. on four MI250).

In [356], for the numerical simulation of realistic combustion devices, a special PeleLMeX solver was used, the performance of which was studied on Crusher with MI250X and Summit with V100. With a small number of GPUs used, the performance of the MI250X was found to be one and a half times greater than that of the V100.

In [357], multi-particle collision dynamics calculations were carried out to particle-based description of hydrodynamic interactions using the Cabana 1.0-dev library based on Kokkos 3.5.00, using A100 and MI250. A comparison of multi-GPU servers containing 4 A100 or 4 MI250 showed that with smaller sizes of the calculated system, MI250 is 7% faster, and with a larger size, A100 is 19% faster (according to data from Figure 3 in [357]). This allows us to talk about the comparability of the performance of a full (with two GCD) MI250 and one A100.

A very interesting and relevant article comparing the performance of MI250X, MI100, A100 and V100 [325] concerns the solution of the Navier-Stokes equations for compressible flow using finite difference discretization— for wall bounded turbulent flows. There, the implementation of the STREAmS-2 application, portable to x86-64 CPUs, AMD and Nvidia GPUs, developed on the basis of the object-oriented (using Fortran 2008) application STREAmS-1, was considered.

To work with Nvidia GPUs, [325] uses CUDA Fortran tools (HPC SDK 22.11), and with AMD GPUs, HIPFORT (ROCm 4.5.2 on MI100 and ROCm 5.0.2 on MI250X). To port STREAmS code from CUDA Fortran to HIPFORT, [325] created its own PyconvertSTREAmS tools, since GPUFORT, according to the authors of [325], was rather at the research stage of development. But for optimization, the code obtained in PyconvertSTREAmS may also require subsequent manual modification.

Calculations with analysis of performance scaling on several nodes were carried out in [325] in *Euro*HPC *JU* clusters: for MI250X — on LUMI, for A100 — on Leonardo (see Table 23).

When comparing calculations on one GPU in MI250X, one GCD was used; data on the corresponding calculation times for two different calculation schemes in STREAmS-2 are shown above in Figure 20. The dimensions of the grid used in the calculations were chosen to be maximum for possible placement in the V100 memory capacity [325].

The data in Figure 20 indicates the similarity of the achieved performance between one GCD MI250X and V100; The MI100 was significantly slower than the V100, and the A100 was significantly faster than the GCD alone. As noted in [325], this does not correspond to the peak performance ratios (with FP64) of these GPUs, and there was a more thorough analysis of the execution times of individual kernels, which showed similar results for kernels with the A100 clearly leading in performance. Exmples of unexpected increases in GCD performance achieved with small changes in kernels are also given in [325].

An analysis in [325] using a roofline model (considering only the HBM) found that calculations on the GCD, as expected for CFD, were always memory-bound, and on the A100 were often in the computationally intensive region. Data movement for GCD was found to be significantly higher than for A100, meaning the A100 fetches registers and caches more often than GCD. In addition, [325] pointed out the large use of registers in the weighted essentially non-oscillatory (WENO) scheme used.

It is advisable to add additional comments to the above-described data from article [325]. As for the data on the higher performance of the V100 relative to the MI100, this seems rather natural (this was mentioned above in the section on MI100 performance). However, the V100 comes in two models [185] — with a memory capacity of 16 GB and (which appeared later) with 32 GB, and the V100 model used in [325] with 16 GB when working with a large grid size (but fits in 32 GB MI100) will simply be unacceptable.

Regarding the MI250X, firstly, the assumption of possibly weaker caching is consistent with the MI250X cache disadvantages noted above compared to the A100 (see also Table 20 and Table 21). Secondly, [142] points out a disadvantage of the AMD compiler (in ROCm v4.5.2) compared to the Nvidia compiler (in CUDA v11.0.3): CUDA uses significantly fewer registers per warp than the AMD compiler and provides much higher warp load on SM, which contributes to higher performance of Nvidia GPUs.

Table 27. NekRS performance data on a single GPU using CUDA and HIP for Nvidia and AMD GPUs respectively

| System | Device | Relative Performance |
|---|---|---|
| Summit | V100-16GB | 1.00 |
| ThetaGPU | A100-40GB | 1.57 |
| Perlmutter | A100-40GB | 1.62 |
| Polaris | A100-40GB | 1.62 |
| Spock | MI100-32GB | 0.84 |
| Crusher | MI250X-64GB (1 GCD) | 1.32 |

Thirdly, it is also actual to compare the performance of the full MI250X (the performance of the cheaper MI250 should probably be close) with the A100, which was not carried out in [325]. If we make a natural assumption about the good scalability of STREAmS-2 within the whole MI250X, then dividing the calculation time with MI250X in Figure 20 by two as an initial approximation, we will obtain calculation times comparable to the A100. In addition, one must keep in mind the lack of sufficient information about the level of optimization achieved when working with HIPORT. All this does not contradict the fact that STREAmS-2 achieves a higher percentage of the peak performance of the A100 compared to GCD.

Preprint [325] also obtained data showing the high scalability of STREAmS-2 performance — for example, efficiency of more than 80% when using up to 16 Leonardo nodes and up to 32 LUMI nodes, and good results on a larger number of nodes, giving for STREAmS-2 high potential for using in compressible fluid dynamics.

Argonne National Lab [144] conducted performance studies of the Nek5000/RS application on a range of supercomputers using MI250X, A100 and V100 GPUs on nodes, scaling from a single GPU to many nodes. Calculations were carried out using an upgraded version of Nek5000 for GPU operation (NekRS version 22.0) as part of the ECP ExaSMR project to generate virtual nuclear reactor simulation datasets with high-fidelity coupled physical models of reactor phenomena. Table 27 shows the performance of NekRS for singlerod simulation on a single GPU.

According to this table, a single GCD on a Crusher provides 1.32×the performance gain for solving the Navier-Stokes equation compared to a single V100 on a Summit. The A100 outperforms a single GCD, and overall, according to [144], the performance of a single GCD is about 85% of that of a single A100.

In [144], in particular, a comparison was made of performance scaling on Frontier (with ROCm 5.1.0 and Cray-mpich 8.1.17) and Crusher (with different versions of the SDK — ROCm 5.1.0, ROCm 5.2.0, Cray-mpich 8.1.16 and Cray-mpich 8.1.19). On Crusher, ROCm 5.1.0 gave performance 2–5% faster than ROCm 5.2.0, and performance on Frontier was better than on Crusher.

In [358], for large-eddy simulation problems, the performance of the NekRS and AMR-Wind codes for modeling flows in the atmospheric boundary layer using the Summit and Crusher supercomputers (with nodes containing V100 and MI250X, respectively) was studied in strong and weak scaling variants. For NekRS, a single GCD MI250X on a Crusher has been shown to provide performance comparable to a single V100 on a Summit.

In [142] presents the performance of the MI250X (compared to the MI100 and V100) using the NekBone-based HipBone benchmark (the HipBone proxy application was discussed above in the A100 and MI100 performance sections) using the libParanumal finite element library (developed within the ECP ) with OCCA tools for abstraction between different parallel programming models — for example, OpenMP, OpenCL, CUDA, HIP and SYCL. HipBone, like Nekbone, uses a regular mesh of hexahedral elements.

The calculations in [142] were carried out on the Summit, Spock and Crusher computing systems (see Table 23) using CUDA 11.0.3, ROCm 4.5.0 and ROCm 4.5.2 in their nodes, respectively. Testing was conducted on a single GCD, allowing for potentially more than half of the MI250X's total compute capabilities and higher clock speeds than running the same workload on both GCDs simultaneously. However, there was no significant difference in performance between kernels running on one GCD or on both GCDs simultaneously in the MI250X. It was also found that the Nvidia compiler uses significantly fewer registers per warp compared to the AMD compiler, resulting in much higher warp utilization on the SM.

Article [142] also carried out a detailed study of strong and weak scaling in the used computing systems, which is not discussed here.

Report [359] indicates performance data on MI250X, MI100, V100 and A100 for the NekRS application and the HipBone proxy application. Computing systems Summit, Spock, Crusher and ThetaGPU were used for calculations (see Table 23).

The performance of the Poisson operator (as well as that of other operators in mesh methods — calculating the result of the operator's apply to a function at mesh points), which determines the calculation time of HipBone, for different degrees of the polynomial used for one GCD is 20–30% greater than for V100 (see Figure 43 in [359]).

All devices achieved the highest HipBone performance at the highest polynomial degree used (15) — 2101.4 GFLOPS in V100, 2135.2 GFLOPS in MI100, 2774.9 GFLOPS in a single GCD [142, 359]. For NekRS, one GCD on Crusher on different kernels gave from 57% to 88% of the performance of A100, in terms of total calculation time — 71% [359].

A number of publications have been discussed above regarding the performance of GPU-oriented software NekRS and subsequent NekBone and HipBone. All are focused on being portable across different types of GPUs, are heavily involved in ECP work, and use C++ tools (although NekBone has a version with CUDA Fortran [358], and all are based on Nek5000, which used Fortran).

But this direction using C++ has an alternative, which is also based on Nek5000 and is focused on portability to different types of accelerators, and it was also used in studies of the achieved performance using the GPUs discussed in the review. In [171], for the field of direct numerical simulations of turbulence with applications in sustainable shipping, a simulation of the flow around a lettner rotor (at Re = 30000) and its interaction with a turbulent boundary layer was carried out on clusters with multi-GPU servers in nodes using A100 and MI250X.

For this purpose, Neko software for working with unstructured meshes, also based on Nek5000, was used and modernized in [171]. Neko uses the object-oriented capabilities of modern Fortran standards to control memory allocation and provide multi-layered abstractions of the solver stack, enabling computation on a variety of architectures from conventional CPUs to different types of accelerators.

In [171], a device abstraction layer is used to manage device memory, data transfer, and kernel execution from Fortran, and CUDA and HIP backends are developed. The calculations here were performed on an Alvis cluster having by 4 A100-SXM4 per node (CUDA version 11.1.1 was used) with a Mellanox ConnectX-6 interconnect ($2 \times 400$ Gb/s) — using OpenMPI 4.0.5, and on an HPE Cray EX cluster, having by 4 MI250X per node (version ROCm 4.5.2 was used) with an HPE Slingshot 10 interconnect — using Cray MPICH 8.1.14. On the hosts in Alvis gcc 10.2 was used, in HPE — Cray cce 13.0.

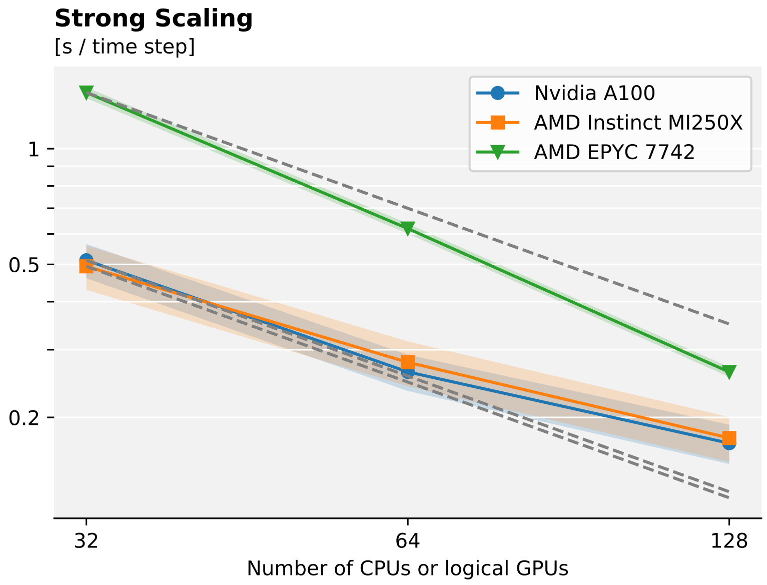


Figure 23. Performance in time per time step: shaded areas refer to standard deviation. The orange and blue lines represent GPU systems. There are two logical GPUs in each MI250X (picture from [171])

The hosts in both clusters had 512 GB of DDR4 memory (with two Intel Xeon Gold 6338 per Alvis node and an AMD EPYC 7A53 per HPE node).

Neko calculations were memory-bound, which the authors believe allows compute-intensive applications to benefit from the higher peak FP64 performance of the MI250X [171]. [171] concluded that the two GCDs in the MI250X match the two A100 in terms of performance. The average time per time step differs by less than 5% when comparing two A100 to one MI250X (see Figure 23). But this contrasts with data discussed above, for example [359], that a single MI250X GCD has performance closer to 71% of that of a single A100.

A number of publications that provide performance data for the MI250 or MI250X relate to magnetohydrodynamics. The article [360] presented a new Idefix code for nonrelativistic fluid dynamics and magnetohydrodynamics based on Kokkos, which in benchmarks for magnetohydrodynamics on a server with 4 MI250X showed performance 2.57 times faster than on a server with 4 V100. Also [360] shows much higher power efficiency with the MI250 than with the CPU.

In [361], calculations were carried out using the Athena++ program (more precisely, the AthenaK code was used to work with the GPU) for general relativistic magnetohydrodynamics using Crusher nodes (with 4 MI250X per node) and Polaris (with 4 A100 per node). Compared to the same number of logical GPUs (i.e., comparing the GCD number for the MI250X to the A100 number), Polaris is, roughly speaking, a couple of times faster for any number of logical GPUs (see Figure 32 in [361]), and still scales good in both computing systems. This suggests that when comparing the A100 to the entire MI250X, they are competitive in performance.

Article [362] provides data on the performance of MI250X, MI100, A100-40GB and V100-16GB in cluster systems using PARTHENON-HYDRO — a mini-application (based on the astrophysical magnetohydrodynamics code Athena++), consisting of about 1.4 thousand lines of C++ for 1D, 2D and 3D compressible hydrodynamics on uniform and multilevel meshes.

The obtaining performance (here considered simply as some relative value) is 5.7 — for the MI250X (two GCDs, with ROCm 5.1.0); 4.2 — for A100 (with CUDA 11.5); 2.7 and 2.15 — for V100 and MI100, respectively. This means that the full MI250X is well ahead of the V100 and MI100 in performance, and faster than the A100 (although when calculated on a single GCD, the MI250X lags behind the A100). Data on the achieved strong and weak scalability in the systems used for calculations were also considered in [362], but this is not analyzed here.

**Plasma physics.** In principle, this area also belongs to CFD, but is highlighted here as a separate part, since there have been a number of publications in plasma physics that have used the MI250/MI250X and obtained interesting performance data.

Thus, in [363], a higher performance of the MI250 compared to the V100 was shown in solving the Vlasov kinetic equation for the dynamics of plasma charged particles.

In [364] studied the performance of the CGYRO code, which solves five-dimensional gyrokinetic-Maxwell equations describing the evolution of plasma microturbulence in magnetic fusion devices. To work with GPUs (MI250X in Frontier and A100 in Perlmutter), OpenACC tools were used (from HPE Cray Fortran for MI250X and from Nvidia Fortran for A100), as well as the cuFFT and hipFFT libraries for A100 and MI250X, respectively.

In the initial test, CGYRO on the MI250X node was significantly slower than on the A100 node, but after optimization it became faster with the MI250X. But in [364], calculations were compared with the same number of nodes and GPUs of these supercomputers; accordingly, the full MI250X of two GCDs turned out to be faster than the A100.

In [365], the calculation time of the HiPACE++ application for modeling plasma accelerators with a quasi-static particle-in-cell algorithm was compared for the A100-80GB and MI250X (one GCD). In this case, all the main data for the calculation is located entirely in the GPU memory during the calculation. CUDA 11.0 was used on the A100, and early test access to Crusher was used for calculations on GCD (probably with HIP). FFT was used in the calculations, and rocFFT, according to [365], then had poor performance at grid sizes other than a power of two. The calculation time on the GCD at two different types of calculation stages ranged from 0.7 to 1.6 and from 0.9 to 3.7 times compared to the time on the A100.

AMD cites the PIConGPU application, which also uses the particle-in-cell algorithm, as an example of a plasma physics application adapted to run on the MI200 GPU by using the ALPAKA (Abstraction Library for Parallel Kernel Acceleration) backend running on top of HIP/ROCm [366].

The particle-in-cell method is generally widely used in plasma physics applications, and it was also used in calculations using MI250X. In [367] 3D modeling of laser-matter interaction was studied on the massively parallel WarpX PIC code using GPUs on Frontier, Summit and Perlmutter (a similar study was also carried out in [368]).

Preprint [369] analyzed the performance of the linear iterative solver TFQMR using Kokkos (including a faster batch version) available in the PETSc (Portable Extensible Toolkit for Scientific Computing) library. This is interesting for plasma physics when working with the Landau collision operator. The performance of the MI250X and A100 here was evaluated using Perlmutter and Crusher cluster nodes.

The above-mentioned articles [365, 367, 368] are directly related to projects from ECP. Like many of the other publications used in Subsection 5.3.2.2, the latest data obtained in the newly appeared computing systems with GPU MI250/MI250X are used here; this data was often considered preliminary due to the possible rapid progress of new versions of the SDK from AMD.

**Artificial intelligence tasks.** Although AMD didn't put AI exactly first place of potential applications for the MI200, these GPUs were immediately use for AdI applications. Thus, in [28], the integration of HPC and deep learning was used: for the Monte Carlo application (for thermodynamic calculations in materials science), a set of surrogate deep learning models was developed and calculations were performed on many nodes of two supercomputers: Summit c V100 (using the CUDA stack) and Crusher with MI250X (using the ROCm stack). On different model sizes, one GCD was up to 15% faster than the V100 (and the whole MI250X was correspondingly faster by up to 2.3 times). Crusher also showed better performance scaling relative to Summit.

These GPUs have become used for AI tasks in a wide variety of fields that use AI. Thus, in [370], MI250X was used for mitochondrial segmentation tasks.

And after the deployment of the European supercomputer LUMI with MI250X, probably due to the use of different languages in Europe, publications on natural language processing on MI250X began to actively appear there, including using their own modified BERT models (see, for example, [371–374]).

## 5.4. AMD GPUs Summary

The above performance data for the MI200 GPUs — mostly the MI250 and MI250X — certainly suggests that these GPUs are somewhat competitive with the A100. Both AMD with ROCm and application developers are actively pushing forward to achieve higher performance.

It's clear that the performance data already reported often doesn't meet expectations based on the higher peak performance of these AMD GPUs compared to the A100 (meaning even a single GCD). Although in many cases the benchmarks were memory bound rather than compute-intensive in the roofline model, this still does not correlate with the many performance gaps relative to the A100, as well as the sometimes identifiable performance "sinks" of the MI250X (see, for example, data [328]).

In a number of publications, this is associated with the insufficiently large cache size (mainly L1) relative to the A100, and the use of too many registers in the codes generated by AMD compilers compared to Nvidia CUDA tools, although in some cases the reasons for insufficient performance compared to expected performance remain unclear. The existing comments regarding AMD's SDK software correlate with the clear indication in many of the above publications that the performance data obtained is preliminary, therefore suggesting improvements due to the rapid development of the SDK.

Since there are also data on higher performance of MI250/MI250X, the fact that the comparative performance of these GPUs relative to the A100 is not sufficiently predictable at this point in time should be considered important.

Here one cannot try to be based on a “statistical analysis” of comparative performance data, but must be based on data for specific applications and objects of study (which, to a first approximation, are also available in this review), as well as on similar (in purely mathematical sense) methods used, and take into account the already discovered shortcomings of MI250/MI250X and their SDK (the latter may possibly be corrected in new versions).

But it is necessary to keep in mind the possible opposite conclusions about the efficiency of the MI200 when taking into account comparative prices and TCO, and accordingly choose to compare the full MI250/MI250X or individual GCDs with the A100.

At the time of writing this review, the MI300 and the EI Capitans supercomputer are expected to appear (in 2023, at the Lawrence Livermore National Laboratory (USA)), with the integration of Zen 4 architecture CPUs into the MI300. This confirms AMD’s likely progress in GPUs and supercomputers in the near future. [**375**] pointed out the continued small size of the L1 cache — this could be a potential weak point of the MI300.

In relation to HPC, it is worth pointing out AMD’s great focus on this area compared to Nvidia’s greater focus on AI, which is evident in the new supercomputers included in the Top500.

## Conclusion

In this review, the advantages and possible disadvantages of GPUs were discussed above. As for specific performance data, a lot has been said about them above, and it is advisable to combine them with cost indicators (including energy consumption indicators), which are not discussed here. What follows is a mostly general summary of a slightly different, more general and/or briefer nature.

1. There is competition among modern GPU developers in terms of performance and other indicators, including not only from the USA, but possibly also from China; a European accelerator is also expected, which can also be used on exascale supercomputers. Accordingly, current Nvidia GPUs near-monopoly will likely diminish little by little.

2. The general technological direction of building GPUs is the use of multi-chip technologies (chiplets), which were probably first actively used by AMD [376]. This enables performance scalability that can be implemented in a cost-effective manner with currently negligible latency increases, including for interconnects between compute cores.
3. The general trend may be the integration of CPU and GPU (AMD MI300, Intel Falcon Shore, Nvidia GH200), as well as several GPUs within one model (as in AMD MI200 and Intel PVC).
4. The ultra-fast development of GPUs leads to the fact that attempts to standardize any hardware components are still being implemented to a limited extent. Thus, the use of the OAM form factor standard in MI200, PVC and BR100 is combined with Nvidia's use of its own SXM form factors, which looks quite understandable in the light of Nvidia's supply of its own ready-to-run computing systems — from multi-GPU servers to clusters.

   Standardization is even more difficult for GPU interconnects — all of the GPUs reviewed used their own, different selection from the competition. Here, the reason for the orientation not towards standards is rather the competitive struggle for leadership in performance.
5. The general direction is to expand the use of multi-GPU systems for AI and HPC tasks. In the corresponding servers, it is necessary to use two processors containing dozens of processor cores, which include not only Intel Xeon and AMD EPYC, but also ARM processors (the classic option would be Nvidia Grace or GH200). An alternative may be to use one server processor containing over fifty cores (both AMD and Intel offer such CPUs today).

   Traditional for HPC quantum chemistry problems previously used GPUs primarily for calculations in plane wave basis sets or computationally more complex methods taking into account electron correlation, where the bulk of the calculation time was spent on general mathematical methods. But for widespread quantum chemistry problems, performance is limited by the time it takes to calculate two-electron integrals in a Gaussian basis (in the HF and DFT methods). Here, high efficiency of GPU calculations, which gives good performance scalability when working with multiple GPUs, was achieved only very recently (the corresponding data is presented in the review). This makes it possible to more actively use multi-GPU systems also for quantum chemistry problems.

6. As for the BR100, they may be relevant primarily to software developers, including in the scientific field (since there are practically no applications running on the BR100) — and, perhaps, especially in countries where program developers and GPU consumers have significant financial restrictions. However, the absence of the FP64 format in the BR100, traditional in the classical HPC field, will likely lead to the BR100's main focus on AI tasks. But the prospects for the use of BR100 are unclear due to US sanctions.

7. Given certain development delays in Intel's semiconductor technology and delays in the start of shipments of Ponte Vecchio compared to Intel's original plans, and the emerging need to move to work with the DPC++ /oneAPI software model, it can be assumed that much faster progress with GPUs by Intel can be expected after the appearance of the Falcon Shores GPU or a subsequent x86-integrated device (rather after the implementation of the new promising 18A technology).

   Intel's most striking contribution to the development of GPU applications today seems to be the development and support of the DPC++ programming model.

8. There is no doubt that the adoption of Nvidia's H100 GPU will continue to expand rapidly (with faster growth of H100 using for AI tasks). Both the already released and the upcoming Nvidia H100 GPUs (GH200 also actually contain the H100) will be very relevant for AI and HPC tasks. But the most powerful leap forward in performance scalability seems to be the ability to build clusters with the H100 using NVLink networks.

   The advantage of Nvidia hardware primarily for AI tasks is the delivery of AI-ready DGX server systems and DGX SuperPOD clusters, which are at the supercomputing level. Although such systems can be used for HPC tasks, applications ready for them are needed. Another advantage of the H100 is its huge set of SDKs for HPC and AI that interact effectively and complement each other.

9. GPUs from AMD, as an alternative to Nvidia GPUs, seem to be most relevant at present, primarily for calculations on applications that

have already shown high performance on the AMD MI200, or are expected to do so in the very near future, as well as a field for the development of new software, including porting existing applications. In terms of potential application use in the near future, this could make sense for any HPC and AI fields.

The use of these GPUs will obviously expand primarily in science and in the activities of software developers. There is a certain degree of comparability in performance with the Nvidia GPUs reviewed, which means that price points data are of increased importance.

10. Taking into account current trends, it can be assumed that the use of universal software development tools on GPUs will expand, rather than those focused on a specific manufacturer's architecture.

## References

[1] *Top500 the list*, The Green500 ranking, 61st edition, 2023. URL ↑307

[2] W. Tschudi, T. Xu, D. Sartor, J. Stein. *High-performance data centers. A research roadmap*, LBNL-53483, 2004, 53 pp. URL ↑307

[3] T. Maltenberger, I. Ilic, Tolovski I, T. Rab. "Evaluating multi-GPU sorting with modern interconnects", *SIGMOD'22: Proceedings of the 2022 International Conference on Management of Data* (Philadelphia, PA, USA, June 12–17, 2022), ACM, New York, 2022, ISBN 978-1-4503-9249-5, pp. 1795–1809. doi ↑307, 324, 355

[4] *Top500 the list*, The Top500 ranking, 61st edition, 2023. URL ↑308, 309, 421

[5] M. B. Kuzminsky. "Modern server ARM processors for supercomputers: A64FX and others. Initial data of benchmarks", *Program Systems: Theory and Applications*, **13**:1(52) (2022), pp. 131–194. doi URL ↑308, 312, 365

[6] J. Gao, F. Zheng, Qi F, Ding Y, H. Li, H. Lu, W. He, H. Wei, L. Jin, X. Liu, D. Gong, F. Wang, Y. Zheng, H. Sun, Z. Zhou, Y. Liu, H. You. "Sunway supercomputer architecture towards exscale computing: analysis and practice", *Science China Information Sciences*, **64**:4 (2021), id. 141101, 21 pp. doi URL ↑308

[7] J. Selig. *The cerebras software development kit: A technical overview*, Cerebras systems Inc., 2022, 8 pp. URL ↑308

[8] *Andromeda, a 13.5 Million Core AI Supercomputer*, a section on the Cerebras company site, 2024. URL ↑308

[9] *Top500 the list*, List Statistics of TOP500, 61st edition, 2023. URL URL ↑308

[10] T. P. Morgan. *Chip roadmaps unfold, crisscrossing and interconnecting, at AMD*, The Next Platform, Stackhouse Publishing, 2022. URL ↑308, 313

[11] A. Shah. *Intel Reiterates Plans to Merge CPU, GPU High-performance Chip Roadmaps*, Tabor network, HPCwire, 2022. URL ↑308

[12] T. P. Morgan. *The Increasingly Graphic Nature Of Intel Datacenter Compute*, The Next Platform, Stackhouse Publishing, 2022. URL ↑308

[13] J. Evans. "Nvidia Grace", *2022 IEEE Hot Chips 34 Symposium (HCS)* (Cupertino, CA, USA, 21–23 August 2022), IEEE, 2022, pp. 1–20. doi ↑308, 313, 314, 384, 385

[14] A. C. Elster, T. A. Haugdahl. "Nvidia Hopper GPU and Grace CPU Highlights", *Computing in Science and Engineering*, **24**:2 (2022), pp. 95–100. doi ↑308, 314, 386

[15] J. Evans. *Inside Grace*, Featured Playlists, GPU Technology Conference (GTC), Nvidia On-Demand, Nvidia, 2022. URL ↑308, 314

[16] *CUDA C++ Programming Guide*, Release 12.4, Nvidia, 2024, 544 pp. URL ↑309, 315, 316, 317, 347, 349, 350, 352, 359

[17] *Ampere Tuning Guide*, Release 12.4, Nvidia, 2024, 22 pp. URL ↑309, 319

[18] Z. Zhang, S. Jiao, J. Li, W. Wu, L. Wan, X. Qin, W. Hu, J. Yang. "KSSOLV-GPU: An efficient GPU-enabled MATLAB toolbox for solving the Kohn-Sham equations within density functional theory in plane-wave basis set", *Chinese Journal of Chemical Physics*, **34**:5 (2021), pp. 552–564. doi ↑310

[19] P. Giannozzi, O. Baseggio, P. Bonfà, D. Brunato, R. Car, I. Carnimeo, C. Cavazzoni, de Gironcoli S., P. Delugas, Ruffino . F., A. Ferretti, N. Marzari, I. Timrov, A. Urru, S. Baroni. "Quantum ESPRESSO toward the exascale", *The Journal of chemical physics*, **152**:15 (2020), id. 154105. doi ↑310

[20] He Lizhong. *The pace of GPU localization is accelerating, and emerging teams continue to emerge*, Industry Brief Report, Capital Securities, Beijing, 2022 (in Chinese), 15 pp. URL ↑311

[21] J. Bispo, J. Barbosa, P. Silva, C. Morales, M. Myllykoski, P. Ojeda-May, M. Bialczak, M. Uchroński, odarczyk A. Wł, P. Wauligmann, E. Krishnasamy, S. Varrette, S. Lührs. *Best Practice Guide: Modern Accelerators*, ed. Shoukourian H., PRACE, 2021, 111 pp. URL ↑311, 312, 396

[22] J. Finkelstein, J. S. Smith, S. M. Mniszewski, K. Barros, C. F. A. Negre, E. H. Rubensson, A. M. N. Niklasson. "Quantum-based molecular dynamics simulations using tensor cores", *Journal of Chemical Theory and Computation*, **17**:10 (2021), pp. 6180–6192. doi ↑311

[23] S. Posey, J. Luitjens, O. Hennigh, S. Oberlin. "GPU-based HPC and AI developments for CFD" (Maui, Hawaii, USA, July 11-15, 2022), 2022, id. ICCFD11-3803, 5 pp. URL ↑311

[24] R. Schade, T. Kenter, H. Elgabarty, M. Lass, O. Schütt, A. Lazzaro, H. Pabst, S. Mohr, J. Hutter, T. D. Kühne, C. Plessl. "Towards electronic structure-based *ab-initio* molecular dynamics simulations with hundreds of millions of atoms", *Parallel Computing*, **111** (July 2022), id. 102920, 11 pp. doi ↑311

[25] O. Terzo, J. Martinovič (eds.). *HPC, Big Data, and AI Convergence Towards Exascale: Challenge and Vision*, 1st ed., CRC Press, 2022, ISBN 9781003176664, 322 pp. ↑312

[26] M. Nowicki, Górski Ł., P. Bała. "PCJ Java library as a solution to integrate HPC, Big Data and Artificial Intelligence workloads", *Journal of Big Data*, **8**:1 (2021), pp. 1–21, id. 62. ↑312

[27] F. Yin, F. Shi. "A comparative survey of Big Data computing and HPC: from a parallel programming model to a cluster architecture", *International Journal of Parallel Programming*, **50**:1 (2022), pp. 27–64. ↑312

[28] J. Yin, F. Wang, M. Shankar. "Strategies for integrating deep learning surrogate models with HPC simulation applications", *2022 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)* (Lyon, France, 2022), IEEE, 2022, ISBN 978-1-6654-9747-3, pp. 01–10. ↑312, 437

[29] S. R. Sukumar, J. A. Balma, C. D. Rickett, K. J. Maschhoff, J. Landman, C. R. Yates, A. G. Chittiboyina, Y. K. Peterson, A. Vose, K. Byler, J. Baudry, I. A. Khan. "The convergence of HPC, AI and Big Data in rapid-response to the COVID-19 pandemic", *Driving Scientific and Engineering Discoveries Through the Integration of Exeriment, Big Data, and Modeling and Simulation: 21st Smoky Mountains Computational Sciences and Engineering, SMC 2021, Virtual Event, October 18-20, 2021, Revised Selected Papers*, Communications in Computer and Information Science, vol. **1512**, 2022, ISBN 978-3-030-96497-9, pp. 157-172. ↑312

[30] J. Ejarque, R. M. Badia, L. Albertin, f, G. Aloisio, E. Baglione, Y. Becerra, S. Boschert, J. R. Berlin, A. D'Anca, D. Elia, F. Exrtier, S. Fiore, J. Flich, A. Folch, S. J. Gibbons, N. Koldunov, F. Lordan, S. Lorito, Løvholt F., J. Macías, M. Volpe. "Enabling dynamic and intelligent workflows for HPC, data analytics, and AI convergence", *Future Generation Computer Systems*, **134** (September 2022), pp. 414–429. ↑312

[31] N. Ihde, P. Marten, A. Eleliemy, G. Poerwawinata, P. Silva, I. Tolovski, Ciorba . M., T. Rabl. "A survey of Big Data, High Performance Computing, and Machine Learning benchmarks", Technology Conference on Performance Evaluation and Benchmarking, Lecture Notes in Computer Science, vol. **13169**, Springer, Cham, 2021, ISBN 978-3-030-94436-0, pp. 98–118. ↑312

[32] *High-Performance Deep Learning Project (HiDL)*, NOWLAB: Network Based Computing Lab, Ohio state university. URL ↑312

[33] *High-Performance Big Data Project (HiBD)*, NOWLAB: Network Based Computing Lab, Ohio state university. URL ↑312

[34] W. Jeon, G. Ko, J. Lee, H. Lee, D. Ha, W. W. Ro. "Deep learning with GPUs", *Advances in Computers*, **122** (2021), pp. 167–215. ↑312

[35] M. Hong, L. Xu. "Biren BR100 GPGPU: Accelerating Datacenter Scale AI Computing", 2022 IEEE Hot Chips 34 Symposium (HCS) (Cupertino, CA, USA), 2022, pp. 1–22. ↑312, 319, 321, 322, 323, 324, 325, 326

[36] A. Shilov. *Russian Company Taps China's Zhaoxin x86 CPU to Replace AMD, Intel CPUs*, Tom's Hardware, Future US, New York, 2022. URL ↑312

[37] H. Shang, F. Li, Y. Zhang, L. Zhang, Y. Fu, Y. Gao, Y. Wu, X. Duan, R. Lin, X. Liu, Y. Liu, D. Chen. "Exreme-scale *ab initio* quantum Raman spectra simulations on the leadership HPC system in China", *SC'21: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ACM, New York, November 2021, ISBN 978-1-4503-8442-1, id. 6, 13 pp. doi ↑313

[38] D. Schneider. "The Exascale Era is Upon Us: The Frontier supercomputer may be the first to reach 1 000 000 000 000 000 000 operations per second", *IEEE Spectrum*, **59**:1 (January 2022), pp. 34–35. doi ↑313

[39] J. Dongarra, A. Geist. *Report on the Oak Ridge National Laboratory's Frontier System*, Technical Report ICL-UT-22-05, Oak Ridge National Laboratory, 2022, Accessed 15.10.2023. URL ↑313

[40] *Frontier Spec Sheet*, Oak Ridge National Laboratory, UT-Battelle, 2019, 4 pp. URL ↑313

[41] *GPU nodes — LUMI-G*, Hardware documentation, LUMI (Large Unified Modern Infrastructure) consortium. URL ↑313, 400, 403, 404, 406, 407, 408

[42] G. S. Markomanolis, A. Alpay, J. Young, M. Klemm, N. Malaya, A. Esposito, J. Heikonen, S. Bastrakov, A. Debus, T. Kluge, K. Steiniger, J. Stephan, R. Widera, M. Bussmann. "Evaluating GPU programming models for the LUMI supercomputer", *Supercomputing Frontiers*, Lecture Notes in Computer Science (Asian Conference on Supercomputing Frontiers), vol. **13214**, Springer, Cham, 2022, ISBN 978-3-031-10419-0, pp. 79–101. doi ↑313, 366, 373, 396, 408, 409, 410, 414, 416

[43] *Aurora*, Argonne Leadership Computing Facility, Argonne National Laboratory. URL ↑313

[44] O. Peckham. *LRZ announces new phase of SuperMUC-NG Supercomputer with Intels Ponte Vecchio GPU*, Tabor network, HPCwire, 2021. URL ↑313

[45] J. H. Kwack, J. Tramm, C. Bertoni, Y. Ghadar, B. Homerding, E. Rangel, C. Knight, S. Parker. "Evaluation of performance portability of applications and mini-apps across AMD, Intel and Nvidia GPUs", *2021 International Workshop on Performance, Portability and Productivity in HPC (P3HPC)* (14 November 2021, St. Louis, MO, USA), IEEE, 2021, ISBN 978-1-6654-2439-4, pp. 45–56. doi ↑313, 413

[46] *HPE Cray Supercomputing EX*, Hewlett Packard Enterprise Development LP, 2024. URL ↑313, 408

[47] C. Bertoni, S. Parker. *Aurora overvew*, ALCF SDL Workshop (October 6, 2022), 2022, 20 pp. URL ↑313, 337

[48] T. P. Morgan. *The NVSwitch Fabric That Is The Hub Of The DGX H100 SuperPOD*, The Next Platform, Stackhouse Publishing, 2022. URL ↑313

[49] A. Ishii, R. Wells. “The Nvlink-Network switch: Nvidia’s switch chip for high communication-bandwidth superpods”, *2022 IEEE Hot Chips 34 Symposium (HCS)* (21–23 Aug., 2022, Cupertino, CA, USA), IEEE, 2022, ISBN 978-1-6654-6028-6, pp. 1–23. doi ↑313

[50] A. Eassa, A. Ishii, R. Wells. *Upgrading Multi-GPU Interconnectivity with the Third-Generation Nvidia NVSwitch*, Technical blog, Nvidia developer, 2022. URL ↑313

[51] *BR100 series general purpose GPU chip*, Biren Technology, Shanghai, 2023. URL ↑314, 321, 322, 326

[52] M. Andersch, G. Palmer, R. Krashinsky, N. Stam, V. Mehta, G. Brito, S. Ramaswamy. *Nvidia Hopper Architecture In-Depth*, Technical blog, Nvidia developer, 2022. URL ↑314

[53] P. Alcorn. *From Opteron to Milan: Crusher Supercomputer Comes Online With New AMD CPUs and MI250X GPUs*, Tom’s Hardware, Future US, New York, 2022. URL ↑314

[54] *Intel Xeon CPU Max series product overview*, Intel, 2023, Accessed 15.10.2023. URL ↑314

[55] *Accelerator Processor Stream*, European Processor Initiative, 2022. URL ↑314

[56] *EPI EPAC1.0 RISC-V test chip samples delivered*, News, European Processor Initiative, 2021. URL ↑314

[57] M. Kovač, P. Notton, D. Hofman, J. Knezović. “How Europe is preparing its core solution for exascale machines and a global, sovereign, advanced computing platform”, *Mathematical and Computational Applications*, **25**:3 (2020), pp. 46. doi ↑314

[58] *HIP Programming Guide*, Version 5.0, 2023, Accessed 15.10.2023. URL ↑315, 316, 317, 358, 409

[59] *OpenMP Application Programming Interface*, Version 5.2, OpenMP Architecture Review Board, 2021, 669 pp. URL ↑315

[60] *Khronos OpenCL Registry*, Formatted specifications and other related documentation, Khronos Group. URL ↑315

[61] *SYCL 2020 Specification*, rev. 6, Khronos Group, 2022, 585 pp. URL ↑315

[62] *DPC++ Part 1: An Introduction to the New Programming Model*, Intel (Accessed 15.10.2023). URL ↑315

[63] N. N. Bavarsad, H. M. Makrani, H. Sayadi, L. Landis, S. Rafatirad, H. Homayoun. “HosNa: A DPC++ benchmark suite for heterogeneous architectures”, *2021 IEEE 39th International Conference on Computer Design (ICCD)* (24–27 October 2021, Storrs, CT, USA), IEEE, 2021, ISBN 978-1-6654-3219-1, pp. 509–516. doi ↑315

[64] C. Trott, L. Berger-Vergiat, D. Poliakoff, S. Rajamanickam, D. Lebrun-Grandie, J. Madsen, N. Al Awar, M. Gligoric, G. Shipman, G. Womeldorff. “The Kokkos EcoSystem: comprehensive performance portability for high performance computingâ”, *Computing in Science & Engineering*, **23**:5 (2021), pp. 10–18. doi ↑315

[65] C. R. Trott, D. Lebrun-Grandié, D. Arndt, J. Ciesko, V. Dang, N. Ellingwood, R. Gayatri, E. Harvey, D. S. Hollman, D. Ibanez, N. Liber, J. Madsen, J. Miles, D. Poliakoff, A. Powell, S. Rajamanickam, M. Simberg, D. Sunderland, B. Turcksin, J. Wilke. "Kokkos 3: Programming model extensions for the exascale era", *IEEE Transactions on Parallel and Distributed Systems*, **33**:4 (2021), pp. 805–817. doi ↑315

[66] S. Moore. *The state of the LAMMPS KOKKOS package*, SAND2021-9785C, Sandia National Lab, Albuquerque, NM, 2021, Accessed 15.10.2023. URL ↑315

[67] Y. Ghadar, T. Applencourt, B. Homerding, K. Harms, J. Hammond. *SYCL Programming Model for Aurora*, 2020 ECP Annual Meeting, 2020. ↑316, 329, 338

[68] R. Van Oostrum, N. Chalmers, D. Mc Dougall, P. Bauman, N. Curtis, N. Malaya, N. Wolfe. *AMD GPU Hardware Basics*, Frontier Application Readiness Kick-Off Workshop, 2019, 55 pp. URL ↑316

[69] *Intel oneAPI GPU Optimization Guide Release 2022.3*, Intel (Accessed 15.10.2023). URL ↑317, 329, 331, 332, 333, 334, 335, 336, 338

[70] D. Khudia, J. Huang, P. Basu, S. Deng, H. Liu, J. Park, M. Smelyanskiy. *Fbgemm: Enabling high-performance low-precision deep learning inference*, 2021, 5 pp. arXiv 2101.05615 ↑318

[71] R. Carrasco, R. Vega, C. A. Navarro. "Analyzing GPU tensor core potential for fast reductions", *2018 37th International Conference of the Chilean Computer Science Society (SCCC)* (05–09 November 2018, Santiago, Chile), IEEE, 2018, ISBN 9781538692349, pp. 1–6. doi ↑318

[72] G. Gupta. *Using Tensor Cores for Mixed-Precision Scientific Computing*, Technical blog, Nvidia developer, 2019. URL ↑318

[73] *Nvidia A100 Tensor Core GPU Architecture*, V1.0, Nvidia, 2020, 82 pp. URL ↑318, 333, 344, 345, 346, 347, 350, 351, 352, 353, 355

[74] *754-2019 — IEEE Standard for Floating-Point Arithmetic*, IEEE Std 754-2019, Revision of IEEE 754-2008, 2019, 84 pp. doi ↑319

[75] D. Kalamkar, D. Mudigere, N. Mellempudi, D. Das, K. Banerjee, S. Avancha, D. T. Vooturi, N. Jammalamadaka, J. Huang, H. Yuen, J. Yang, J. Park, A. Heinecke, E. Georganas, S. Srinivasan, A. Kundu, M. Smelyanskiy, B. Kaul, P. Dubey. *A study of BFLOAT16 for deep learning training*, 2019, 10 pp. arXiv 1905.12322 ↑319

[76] D. Stosic, P. Micikevicius. *Accelerating AI Training with Nvidia TF32 Tensor Cores*, Technical blog, Nvidia developer, 2021. URL ↑319

[77] P. Micikevicius, D. Stosic, N. Burgess, M. Cornea, P. Dubey, R. Grisenthwaite, S. Ha, A. Heinecke, P. Judd, J. Kamalu, N. Mellempudi, S. Oberman, M. Shoeybi, M. Siu, H. Wu. *Fp8 formats for deep learning*, 2022, 9 pp. arXiv 2209.05433 ↑319

[78] *Nvidia H100 Tensor Core GPU Architecture*, Includes final GPU / memory clocks and final TFLOPS performance specs, V1.04, Nvidia, 2023, 71 pp. URL ↑319, 333, 346, 347, 348, 349, 350, 379, 381, 382, 389, 390, 391

[79] W. Sun, A. Li, T. Geng, S. Stuijk, H. Corporaal. “Dissecting tensor cores via microbenchmarks: latency, throughput and numerical behaviors”, *IEEE Transactions on Parallel and Distributed Systems*, **34**:1 (2022), pp. 246–261. doi ↑319, 365

[80] M. Lehmann, M. J. Krause, G. Amati, M. Sega, J. Harting, S. Gekle. “Accuracy and performance of the lattice Boltzmann method with 64-bit, 32-bit, and customized 16-bit number formats”, *Physical Review E*, **106**:1 (2022), id. 015308. doi ↑320

[81] J. Domke, K. Matsumura, M. Wahib, H. Zhang, K. Yashima, T. Tsuchikawa, Y. Tsuji, A. Podobas, S. Matsuoka. Double-precision FPUs in high-performance computing: an embarrassment of riches? *2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS)* (20–24 May 2019, Rio de Janeiro, Brazil), IEEE, 2019, ISBN 978-1-7281-1246-6, pp. 78–88. doi ↑320

[82] R. Schade, T. Kenter, H. Elgabarty, M. Lass, O. Schütt, A. Lazzaro, H. Pabst, S. Mohr, J. Hutter, T. D. Kühne, C. Plessl. “Towards electronic structure-based *ab-initio* molecular dynamics simulations with hundreds of millions of atoms”, *Parallel Computing*, **111** (2022), id. 102920, 11 pp. doi ↑320

[83] R. Schade, T. Kenter, H. Elgabarty, M. Lass, T. D. Kühne, C. Plessl. *Breaking the exascale barrier for the electronic structure problem in* ab-initio *molecular dynamics*, 2022, 6 pp. arXiv 2205.12182 ↑320

[84] V. W. Yu, M. Govoni. *GPU acceleration of large-scale full-frequency GW calculations*, 2022, 54 pp. arXiv 2203.05623 ↑320

[85] J. J. Eriksen. “Efficient and portable acceleration of quantum chemical many-body methods in mixed floating point precision using OpenACC compiler directives”, *Molecular Physics*, **115**:17–18 (2017), pp. 2086–2101. doi ↑320

[86] D. Ruda, S. Turek, D. Ribbrock, P. Zajac. *Very fast FEM Poisson solvers on lower precision accelerator hardware*, ECCOMAS Congress 2022 (5–9 June 2022, Oslo, Norway), 2022, 24 pp. URL ↑320

[87] H. Ootomo, R. Yokota. “Recovering single precision accuracy from Tensor Cores while surpassing the FP32 theoretical peak performance”, *The International Journal of High Performance Computing Applications*, **36**:4 (2022), pp. 475–491. doi ↑320

[88] A. Jain, N. Sharma. “Accelerated AI inference at CNN-based machine vision in ASICs: A design approach”, *ECS Transactions*, **107**:1 (2022), pp. 5165. doi ↑320

[89] B. Gallet, M. Gowanlock. *Computing double precision Euclidean distances using GPU tensor cores*, 2022, 10 pp. arXiv 2209.11287 ↑320, 350

[90] J. Domke, E. Vatai, A. Drozd, P. T. Chen, Y. Oyama, L. Zhang, S. Salaria, D. Mukunoki, A. Podobas, M. T. Wahib, S. Matsuoka. Matrix engines for high performance computing: A paragon of performance or grasping at straws? *2021 IEEE International Parallel and Distributed Processing Symposium (IPDPS)* (17–21 May 2021, Portland, OR, USA), IEEE, 2021, ISBN 978-1-6654-4066-0, pp. 1056–1065. doi ↑320, 351, 362

[91] H. Tan, R. Yan, L. Yang, L. Huang, L. Xiao, Q. Yang. "Efficient multiple-precision and mixed-precision floating-point fused multiply-accumulate unit for HPC and AI applications", *Algorithms and Architectures for Parallel Processing*, 22nd International Conference ICA3PP 2022 (Copenhagen, Denmark, October 10–12, 2022), Lecture Notes in Computer Science, vol. **13777**, Springer Nature Switzerland, Cham, 2023, ISBN 978-3-031-22676-2, pp. 642–659. doi ↑321

[92] *Exlusive interview with Biren Technology executives: Deconstructing the company's first 7nm GPU*, Semiconductor Industry Observation by MooreElite.com (Hefei) Co., 2022 (in Chinese). URL ↑321, 324, 325, 326

[93] *Nvidia A100 Tensor Core GPU Datasheet*, V1.0, Nvidia, 2020, 3 pp. URL ↑323, 324, 325

[94] J. Choquette, E. Lee, R. Krashinsky, V. Balan, B. Khailany. "3.2 The A100 Datacenter GPU and Ampere Architecture", *2021 IEEE International Solid-State Circuits Conference (ISSCC)* (13–22 February 2021, San Francisco, CA, USA), IEEE, 2021, ISBN 9781728195506, pp. 48–50. doi ↑323, 324, 326, 351, 352, 353, 355

[95] *Nvidia A100 tensor core GPU architecture*, V1.0, Nvidia, 2020, 82 pp. URL ↑324

[96] M. Hassanpour, M. Riera, A. González. "A survey of near-data processing architectures for neural networks", *Machine Learning and Knowledge Extraction*, **4**:1 (2022), pp. 66–102. doi ↑324

[97] J. Gómez-Luna, Y. Guo, S. Brocard, J. Legriel, R. Cimadomo, G. F. Oliveira, G. Singh, O. Mutlu. *An experimental evaluation of machine learning training on a real processing-in-memory system*, 2022, 21 pp. arXiv 2207.07886 ↑324

[98] D. Niu, S. Li, Y. Wang, W. Han, Z. Zhang, Y. Guan, T. Guan, F. Sun, F. Xue, L. Duan, Y. Fang, H. Zheng, X. Jiang, S. Wang, F. Zuo, Y. Wang, B. Yu, Q. Ren, Y. Xie. "184QPS/W 64Mb/mm$^2$3D logic-to-DRAM hybrid bonding with process-near-memory engine for recommendation system", *IEEE International Solid-State Circuits Conference (ISSCC)* (20–26 February 2022, San Francisco, CA, USA), IEEE, 2022, pp. 1–3. doi ↑324

[99] *BiLi 106M*, Product details, Biren Technology, Shanghai, 2020–2023. URL ↑323, 325

[100] *BiLi 106B, 106C*, Product details, Biren Technology, Shanghai, 2020–2023. URL ↑323, 325

[101] R. Blankenship, M. Wagh. *Introducing the CXL 3.1 Specification*, Compute express link consortium, 2022, 27 pp. URL ↑324

[102] T. Coughlin. "Digital storage and memory", *Computer*, **55**:1 (2022), pp. 20–29. doi ↑324

[103] *Nvidia A100 Tensor Core GPU Datasheet*, Nvidia, 2021, 3 pp. URL ↑324

[104] *Ampere Tuning Guide*, Release 12.4, Nvidia, 2024, 22 pp. URL ↑324

[105] *Server/OAI*, Wiki page, Open computers project. URL ↑325

[106] *Nvidia DGX A100*, Datasheet, Nvidia, 2023, 2 pp. URL ↑325

[107] T. P. Morgan. *China launches the inevitable indigenous GPU*, The Next Platform, Stackhouse Publishing, 2022. URL ↑326

[108] *BIRENSUPA software development platform*, Product details, Biren Technology, Shanghai, 2023. URL ↑326

[109] *MLPerf inference: datacenter benchmark suite results*, MLCommons. URL ↑326, 327

[110] V. J. Reddi, C. Cheng, D. Kanter, P. Mattson, G. Schmuelling, J. Carole-Wu, B. Anderson, M. Breughe, M. Charlebois, W. Chou, R. Chukka, C. Coleman, S. Davis, P. Deng, G. Diamos, J. Duke, D. Fick, J. S. Gardner, I. Hubara, S. Idgunji, T. B. Jablin, J. Jiao, T. S. John, P. Kanwar, D. Lee, J. Liao, A. Lokhmotov, F. Massa, P. Meng, P. Micikevicius, C. Osborne, G. Pekhimenko, A. T. R. Rajan, D. Sequeira, A. Sirasao, F. Sun, H. Tang, M. Thomson, F. Wei, E. Wu, L. Xu, K. Yamada, B. Yu, G. Yuan, A. Zhong, P. Zhang, Y. Zhou. "Mlperf inference benchmark", *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)* (30 May 2020–03 June 2020, Valencia, Spain), IEEE, 2020, ISBN 978-1-7281-4661-4, pp. 446–459. doi ↑326

[111] M. H. Saad, S. Hashima, W. Sayed, E. H. El-Shazly, A. H. Madian, M. M. Fouda. "Early diagnosis of COVID-19 images using optimal CNN hyperparameters", *Diagnostics*, **13**:1 (2023), id. 76. doi ↑326

[112] J. Devlin, W. Ming-Chang, K. Lee, K. Toutanova. "BERT: Pre-training of deep bidirectional transformers for language understanding", *Human Language Technology: Conference of the North American Chapter of the Association of Computational Linguistics*. V. 1, NAACL-HLT 2019 (June 2–June 7, 2019, Minneapolis, Minnesota, USA), ACL, 2019, ISBN 978-1-950737-13-0, pp. 4171–4186. URL ↑327

[113] *Nvidia TensorRT, an SDK for high-performance deep learning inference*, Web site, Nvidia developer, Nvidia. URL ↑328

[114] D. Blythe. "The $X^e$ GPU architecture", *2020 IEEE Hot Chips 32 Symposium (HCS)* (16–18 August 2020, Palo Alto, CA, USA), IEEE, 2020, ISBN 978-1-7281-7129-6, pp. 1–27. doi ↑329, 330, 336

[115] D. Blythe. "$X^e$HPC Ponte Vecchio", *2021 IEEE Hot Chips 33 Symposium (HCS)* (22–24 August 2021, Palo Alto, CA, USA), IEEE, 2021, ISBN 978-1-6654-1397-8, pp. 1–34. doi ↑329, 331

[116] *Intel data center GPU Max series product brief*, Intel (Accessed 15.10.2023). URL ↑329, 330, 331, 335, 336, 337, 341

[117] *Intel data center GPU flex series product brief*, Intel (Accessed 15.10.2023). URL ↑329

[118] D. Dhote, C. Virmani, K. G. Krishna, S. Raghav. "The science of ray tracing", *International Journal of Computer Applications*, **176**:42 (2020), pp. 15–20. doi ↑329

[119] *Intel data center GPU Max series*, Intel (Accessed 15.10.2023). URL ↑330, 335

[120] H. Jiang. “Intel’s Ponte Vecchio GPU: architecture, systems and software”, *2022 IEEE Hot Chips 34 Symposium (HCS)* (21–23 August 2022, Cupertino, CA, USA), IEEE, 2022, ISBN 978-1-6654-6028-6, pp. 1–29. doi ↑329, 334, 335, 338, 341

[121] M. Sidorova, L. Gorbushin, N. Koneva. “Analytical review of electronic devices of modern supercomputing systems”, Proceedings of the International Russian Automation Conference, RusAutoCon2021 (September 5-11, 2021, Sochi, Russia), Lecture Notes in Electrical Engineering, vol. **857**, Springer, Cham, 2022, ISBN 978-3-030-94201-4, pp. 25–33. doi ↑329

[122] W. Tian, B. Li, Z. Li, H. Cui, J. Shi, Y. Wang, J. Zhao. “Using chiplet encapsulation technology to achieve processing-in-memory functions”, *Micromachines*, **13**:10 (2022), pp. 1790. doi ↑330, 331

[123] S. K. Moore. “3 paths to 3D processors”, *IEEE Spectrum*, **59**:6 (2022), pp. 24–29. doi ↑330

[124] S. Zhang, Z. Li, H. Zhou, R. Li, S. Wang, W. Kyung-Paik, P. He,. “Recent prospectives and challenges of 3D heterogeneous integration”, *e-Prime-Advances in Electrical Engineering, Electronics and Energy*, 2022, id. 100052. doi ↑330

[125] R. Hadidi, B. Asgari, B. A. Mudassar, S. Mukhopadhyay, S. Yalamanchili, H. Kim. “Demystifying the characteristics of 3D-stacked memories: A case study for Hybrid Memory Cube”, *2017 IEEE international symposium on Workload characterization (IISWC)* (01–03 October 2017, Seattle, WA, USA), IEEE, 2017, pp. 66–75. doi ↑330

[126] X. Ma, Y. Wang, Y. Wang, X. Cai, Y. Han. “Survey on chiplets: interface, interconnect and integration methodology”, *CCF Transactions on High Performance Computing*, 2022, no. 4, pp. 43–52. doi ↑330

[127] *Universal chiplet interconnect express specifications*, Universal Chiplet Interconnect Express, 2023. URL ↑330

[128] W. Gomes, A. Koker, P. Stover, D. Ingerly, S. Siers, S. Venkataraman, C. Pelto, T. Shah, A. Rao, O’ .,Mahony, E. Karl, L. Cheney, I. Rajwani, H. Jain, R. Cortez, A. Chandrasekhar, B. Kanthi, R. Koduri. “Ponte Vecchio: A multi-tile 3D stacked processor for exascale computing”, *2022 IEEE International Solid-State Circuits Conference (ISSCC)* (20–26 February 2022, San Francisco, CA, USA), IEEE, 2022, ISBN 978-1-6654-2800-2, pp. 42–44. doi ↑330, 331

[129] W. Gomes, A. Koker, P. Stover, D. Ingerly, S. Siers, S. Venkataraman, C. Pelto, T. Shah, A. Rao, F. O’Mahony, E. Karl, L. Cheney, I. Rajwani, H. Jain, R. Cortez, A. Chandrasekhar, B. Kanthi, R. Koduri. *Ponte Vecchio: A multi-tile 3D stacked processor for exascale computing*, HPC user forum, Accelerated Computing Systems and Graphics Group, 2021. URL ↑330, 331, 332, 333, 334, 335

[130] *Intel data center GPU Max series technical overview*, Intel, 2023 (Accessed 15.10.2023). URL ↑330, 331, 332, 333, 335, 336, 337

[131] S. K. Moore. *Behind Intel’s HPC chip that will pierce the exascale barrier*, Blog, IEEE Spectrum, IEEE, 2022. URL ↑330

[132] D. B. Ingerly, S. Amin, L. Aryasomayajula, A. Balankutty, D. Borst, A. Chandra, K. Cheemalapati, C. S. Cook, R. Criss, K. Enamul, W. Gomes, D. Jones, K. C. Kolluru, A. Kandas, G.-Kim S., H. Ma, D. Pantuso, C. F. Petersburg, M. Phen-givoni, A. M. Pillai, A. Sairam, P. Shekhar, P. Sinha, P. Stover, A. Telang, Z. Zell. "Foveros: 3D integration and the use of face-to-face chip stacking for logic devices", *2019 IEEE International Electron Devices Meeting (IEDM)* (07–11 December 2019, San Francisco, CA, USA), IEEE, 2019, ISBN 978-1-7281-4033-9, pp. 19.6.1-19.6.4. ↑331

[133] R. Mahajan, R. Sankman, N. Patel, W. Dae-Kim, K. Aygun, Z. Qian, Y. Mekonnen, I. Salama, S. Sharan, D. Iyengar, D. Mallik. "Embedded multi-die interconnect bridge (EMIB)–a high density, high bandwidth packaging interconnect", *2016 IEEE 66th Electronic Components and Technology Conference (ECTC)* (31 May 2016–03 June 2016, Las Vegas, NV, USA), IEEE, 2016, pp. 557–565. ↑331

[134] S. Irani. *Hang SK Intel Ponte Vecchio compute accelerator OAM product and system*, 2021 OCP Global Summit, 2021. ↑331

[135] A. Tekin, A.Durak T., C. Piechurski, D. Kaliszan, F. A. Sungur, F. Robertsén, P. Gschwandtn. *State-of-the-art and trends for computing and interconnect network solutions for HPC and AI*, Partnership for Advanced Computing in Europe, PRACE, 2021, 38 pp. ↑332

[136] W. Sun, A. Li, T. Geng, S. Stuijk, H. Corporaal. "Dissecting tensor cores via microbenchmarks: latency, throughput and numerical behaviors", *IEEE Transactions on Parallel and Distributed Systems*, **34**:1 (2023), pp. 246–261. ↑332

[137] *Intel Products formerly Alchemist*, Intel (Accessed 15.10.2023). ↑333

[138] D. Watts. *Lenovo ThinkSystem and ThinkAgile GPU Summary*, Product Guide, Lenovo press, 2024, 71 pp. ↑333

[139] Zh. Liu. *Intel Axes Data Center GPU Max 1350, Preps New Max 1450 for 'Different Markets'*, Tom's Hardware, Future US, New York, 2023. ↑333

[140] R. Vuduc, A. Chandramowlishwaran, J. Choi, M. (E.) Guney, A. Shringarpure. "On the limits of GPU acceleration", *Proceedings of the 2nd USENIX conference on Hot topics in parallelism*, HotPar'10 (June 14–15, 2010, Berkeley, CA, USA), USENIX Association, Berkeley, 2010, id. 13, 6 pp. ↑334

[141] B. Hanindhito, D. Gourounas, A. Fathi, D. Trenev, A. Gerstlauer, L. K. John. "GAPS: GPU-acceleration of PDE solvers for wave simulation", *ICS '22: Proceedings of the 36th ACM International Conference on Supercomputing* (June 28–30, 2022, Virtual Event), ACM, NeW York, 2022, ISBN 978-1-4503-9281-5, id. 30, 13 pp. ↑334

[142] N. Chalmers, A. Mishra, D. Mc Dougall, T. Warburton. "HipBone: A performance-portable GPU-accelerated C++ version of the NekBone benchmark", *The International Journal of High Performance Computing Applications*, **37**:5 (2023), pp. 560-577. ↑334, 417, 430, 432, 433

[143] J.-L. Philippe. *Intel HW roadmap and architecture specifics*, OneAPI workshop with FocusCoE, 2022, 48 pp. ↑334, 336

[144] M. Min, H. Yu-Lan, P. Fischer, T. Rathnayake, J. Holmen. *Nek5000/RS Performance on Advanced GPU Architectures*, ANL-22/81, Argonne National Lab.(ANL), Argonne, IL, 2022., 30 pp. ↑337, 431, 432

[145] *oneAPI GPU Optimization Guide*, edition 2023.1, Intel, 2023, 411 pp. ↑337

[146] D. Blythe. "XeHPC ponte vecchio", 2021 IEEE hot chips 33 symposium (HCS), 2021, pp. 1–34. ↑336

[147] A. J. van der Steen, "Overview of recent supercomputers": J. J. Dongarra, A. J. Van der Steen, *High-performance computing systems: Status and outlook*, Acta Numerica, vol. **21**, 2012, pp. 379–474 . ↑337

[148] *Intel Xeon CPU Max Series*, Product Brief, Intel., 2023, 3 pp. ↑337

[149] G. M. Shipman, S. Swaminarayan, G. Grider, J. Lujan, R. J. Zerr. *Early performance results on 4th Gen Intel Xeon scalable processors with DDR and Intel Xeon processors, codenamed sapphire rapids with HBM*, 2022, 5 pp. arXiv 2211.05712 ↑337

[150] *SiPearl: collaboration with Intel to accelerate exascale supercomputing deployment in Europe*, Press release, The Silicon Pearl, 2 pp. ↑337

[151] S. Parker. *Future ALCF systems*, 2021 ALCF Computational Performance Workshop, Argonne National Laboratory, 2021, 25 pp. ↑338

[152] Y. Ghadar, T. Williams. *An overview of Aurora, Argonnes upcoming exascale system*, ALCF Developer Session (December 11 2019), 2020, 45 pp. ↑338

[153] *SYCL*, The SYCL main page, Khronos Group. ↑338

[154] Castaño G., Y. aqir-Rhazoui, C. García, M. Prieto-Matías. "Evaluation of Intel's DPC++ compatibility tool in heterogeneous computing", *Journal of Parallel and Distributed Computing*, **165** (2022), pp. 120–129. ↑338, 339

[155] *Intel oneAPI 2023 Release: Preview the Tools*, Intel (Accessed 15.10.2023). ↑338

[156] *Intel oneAPI plug-ins from Codeplay for Nvidia and AMD GPUs*, Intel (Accessed 15.10.2023). ↑338

[157] *oneAPI DPC++ Compiler documentation*, LLVM documentation, Intel, 2024. ↑338

[158] *Benchmarking the performance of oneAPI on heterogeneous computing platforms*, webinar slides, Moasys, Intel Software, 2021, 30 pp. ↑338

[159] *OneAPI Specifications*, Unified Acceleration Foundation, 2024. ↑338

[160] Z. Wang, Y. Plyakhin, C. Sun, Z. Zhang, Z. Jiang, A. Huang, H. Wang. "A source-to-source CUDA to SYCL code migration tool: Intel DPC++ Compatibility Tool", *IWOCL '22: Proceedings of the 10th International Workshop on OpenCL* (May 10–12, 2022, Bristol, United Kingdom), ACM, New York, 2022, ISBN 978-1-4503-9658-5, id. 17, 2 pp. ↑339

[161] A. Fortenberry, S. Tomov. "Extending MAGMA portability with OneAPI", *2022 Workshop on Accelerator Programming Using Directives (WACCPD)* (13–18 November, 2022, Dallas, TX, USA), IEEE, 2022, ISBN 978-1-6654-9019-1, pp. 22–31. ↑339

[162] D. J. Hardy, J. Choi, W. Jiang, E. Tajkhorshid. "Experiences porting NAMD to the Data Parallel C++ programming model", *IWOCL '22: Proceedings of the 10th International Workshop on OpenCL* (May 10–12, 2022, Bristol, United Kingdom), ACM, New York, 2022, ISBN 978-1-4503-9658-5, id. 15, 5 pp. ↑339

[163] A. Alekseenko, S. Páll, E. Lindahl. "Experiences with adding SYCL support to GROMACS", *IWOCL '21: Proceedings of the 9th International Workshop on OpenCL* (April 27–29, 2021, Munich, Germany), ACM, New York, ISBN 978-1-4503-9033-0, pp. 1, id. 17. ↑339

[164] *GROMACS Highlights*, GROMACS development team, 2023. ↑339

[165] A. Alpay, B. Soproni, H. Wünsche, V. Heuveline. "Exploring the possibility of a hipSYCL-based implementation of oneAPI", *IWOCL '22: Proceedings of the 10th International Workshop on OpenCL* (May 10–12, 2022, Bristol, United Kingdom), ACM, New York, 2022, ISBN 978-1-4503-9658-5, id. 10, 12 pp. ↑339, 410

[166] I. Sakiotis, K. Arumugam, M. Paterno, D. Ranjan, B. Terzić, M. Zubair, *High Performance Computing*, 38th International Conference ISC High Performance 2023 (May 21–25, 2023, Hamburg, Germany), Lecture Notes in Computer Science, vol. **13948**, Springer, Cham, 2023, ISBN 978-3-031-32040-8, pp. 339–358. ↑339

[167] I. Z. Reguly, A. M. B. Owenson, A. Powell, S. A. Jarvis, G. R. Mudalige. "Under the hood of SYCL — an initial performance analysis with an unstructured-mesh CFD application", *High Performance Computing*, 36th International Conference ISC High Performance 2021 (June 24–July 2, 2021, Virtual Event), Lecture Notes in Computer Science, vol. **12728**, Springer, Cham, 2021, ISBN 978-3-030-78712-7, pp. 391–410. ↑339

[168] A. C. Walden, M. Zubair, E. J. Nielsen. "Performance and portability of a linear solver across emerging architectures", *Accelerator Programming Using Directives*, 7th International Workshop WACCPD 2020 (November 20, 2020, Virtual Event), Lecture Notes in Computer Science, vol. **12655**, Springer, Cham, 2021, ISBN 978-3-030-74223-2, pp. 61–79. ↑339, 409

[169] M. Zubair, C. Stone, A. Walden, E. Nielsen. *Experiences in Moving CUDA-Optimized Kernels to Intel GPUs using oneAPI*, SC21, 2021, 22 pp. ↑339

[170] *Nvidia HPC Compilers User's Guide*, DU-09862-001-V2023, version 2023, 173 pp. ↑339

[171] M. Karp, D. Massaro, N. Jansson, A. Hart, J. Wahlgren, P. Schlatter, S. Markidis. "Large-scale direct numerical simulations of turbulence using GPUs and modern Fortran", *The International Journal of High Performance Computing Applications*, **37**:5 (2023), pp. 487–502. doi ↑340, 433, 434

[172] *OpenMP Compilers & Tools*, OpenMP, 2023. URL ↑340

[173] T. Cojean, Y. H. M. Tsai, H. Anzt. "Ginkgo—A math library designed for platform portability", *Parallel Computing*, **111** (2022), id. 102902. doi ↑340, 366, 367, 368, 372

[174] J. Fuentes, D. López, S. González. "Teaching heterogeneous computing using DPC++", *2022 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)* (30 May 2022–03 June 2022, Lyon, France), IEEE, 2022, ISBN 978-1-6654-9747-3, pp. 354–360, id. 102902. doi ↑340

[175] Y. Fridman, G. Tamir, G. Oren. "Portability and scalability of OpenMP offloading on state-of-the-art accelerators", *High Performance Computing*, ISC High Performance 2023 International Workshops (May 21–25, 2023, Hamburg, Germany), Lecture Notes in Computer Science, vol. **13999**, Springer, Cham, 2023, ISBN 978-3-031-40842-7, pp. 378–390. doi ↑340

[176] I. Z. Reguly. "Evaluating the performance portability of SYCL across CPUs and GPUs on bandwidth-bound applications", *Proceedings of the SC '23 Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis*, SC-W'23 (November 12–17, 2023, Denver, CO, USA), ACM, New York, 2023, ISBN 979-8-4007-0785-8, pp. 1038–1047. doi ↑340, 341, 344, 410

[177] *SPEC CPU2017 Results*, Standard Performance Evaluation Corporation. URL ↑341

[178] L. Solis-Vasquez, E. Mascarenhas, A. Koch. "Experiences migrating CUDA to SYCL: A molecular docking case study", *Proceedings of the 2023 International Workshop on OpenCL*, IWOCL'23 (April 18–20, 2023, Cambridge, United Kingdom), ACM, New York, 2023, ISBN 979-8-4007-0745-2, pp. 1–11, id. 15. doi ↑341, 342

[179] P. Nguyen, P. Nayak, H. Anzt, *Proceedings of the SC '23 Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis*, SC-W '23 (November 12–17, 2023, Denver, CO, USA), ACM, New York, 2023, ISBN 979-8-4007-0785-8, pp. 1048–1058. arXiv 2308.08417 doi ↑342

[180] T. P. Morgan. *One New Feature For Intel's HPC Compute Engines: Contrition*, The Next Platform, Stackhouse Publishing, 2022. URL ↑343

[181] T. P. Morgan. *Aurora In A Socket: What Intel's Falcon Shores XPU Might Do*, The Next Platform, Stackhouse Publishing, 2022. URL ↑343

[182] *Nvidia Parallel Thread Execution ISA*, Release 8.4, Nvidia, 2024, 598 pp. URL ↑345

[183] *Inline PTX Assembly in CUDA*, vol. **1**, Release 12.4, Nvidia, 2024, 16 pp. URL ↑345

[184] *Nvidia Ampere GA102 GPU Architecture*, Updated with Nvidia RTX A6000 and Nvidia A40 Information, V2.0, Nvidia, 2021, 53 pp. URL ↑345

[185] *GPU Specs Database*, A reference list of most graphics cards released in recent years, TechPowerUp. URL ↑345, 346, 350, 351, 379, 398, 399, 404, 430

[186] M. Osama, D. Merrill, C. Cecka, M. Garland, J. D. Owens, *Proceedings of the 28th ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming*, PPoPP'23 (25 February 2023–1 March 2023, Montreal, QC, Canada), ACM, New York, 2023, ISBN 979-8-4007-0015-6, pp. 429–431. doi arXiv 2301.03598 ↑350

[187] *Nsight Compute*, The User Guide for Nsight Compute, v2024.1.1, Nvidia. URL ↑351

[188] A. Li, S. L. Song, M. Wijtvliet, A. Kumar, H. Corporaal. "SFU-driven transparent approximation acceleration on GPUs", *Proceedings of the 2016 International Conference on Supercomputing*, ICS'16 (June 1–3, 2016, Istanbul, Turkey), ACM, New York, 2016, ISBN 978-1-4503-4361-9, pp. 1–14, id. 15. doi ↑352

[189] Z. Jia, M. Maggioni, B. Staiger, D. P. Scarpazza. *Dissecting the Nvidia volta GPU architecture via microbenchmarking*, 2018, 66 pp. arXiv 1804.06826 ↑352

[190] J. Choquette, W. Gandhi, O. Giroux, N. Stam, R. Krashinsky. "Nvidia A100 tensor core GPU: performance and innovation", *IEEE Micro*, **41**:2 (2021), pp. 29–35. doi ↑352, 355, 373, 376

[191] *Multi-Instance GPU User Guide*, RN-08625-v2.0, Nvidia, 2024, iv+53 pp. URL ↑353

[192] *Nvidia A100 80GB PCIe GPU*, Product Brief, PB-10577-001_v03, Nvidia, 2022. URL ↑353

[193] A. Li, S. L. Song, J. Chen, J. Li, X. Liu, N. R. Tallent, K. J. Barker. "Evaluating modern GPU interconnect: PCIe, NVLink, NV-SLI, NVSwitch and GPUDirect", *IEEE Transactions on Parallel and Distributed Systems*, **31**:1 (2019), pp. 94–110. doi ↑354

[194] C. Lutz, S. Breß, S. Zeuch, T. Rabl, V. Markl. "Pump up the volume: Processing large data on GPUs with fast interconnects", *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, SIGMOD'20 (June 14–19, 2020, Portland, OR, USA), ACM, New York, 2020, ISBN 978-1-4503-6735-6, pp. 1633–1649. doi ↑354

[195] *Nvidia NVSwitch*, Technical Overview, Nvidia, 2018, 8 pp. URL ↑354

[196] M. Špeťko, O. Vysocký, B. Jansík, L. Říha. "DGX-A100 Face to Face DGX-2—performance, power and thermal behavior evaluation", *Energies*, **14**:2 (2021), id. 376, 18 pp. doi ↑355, 356

[197] Y. R. Choi, V. Nikolskiy, V. Stegailov. "Matrix-matrix multiplication using multiple GPUS connected by Nvlink", *2020 Global Smart Industry Conference (GloSIC)* (17–19 November 2020, Chelyabinsk, Russia), IEEE, 2020, ISBN 9781728180755, pp. 354–361. doi ↑356

[198] M. Manathunga, C. Jin, V. W. D. Cruzeiro, Y. Miao, D. Mu, K. Arumugam, K. Keipert, H. M. Aktulga, K. M. Merz jr, A. W. Götz. "Harnessing the power of multi-GPU acceleration into the quantum interaction computational kernel program", *Journal of Chemical Theory and Computation*, **17**:7 (2021), pp. 3955–3966. ↑356, 373, 374

[199] Y. R. Choi, V. Nikolskiy, V. Stegailov. "Matrix-matrix multiplication using multiple GPUS connected by Nvlink", *2020 Global Smart Industry Conference (GloSIC)* (17–19 November 2020, Chelyabinsk, Russia), IEEE, 2020, ISBN 9781728180755, pp. 354–361. ↑356

[200] Y. R. Choi, V. Stegailov. "Multi-GPU GEMM algorithm performance analysis for Nvidia and AMD GPUs connected by NVLink and PCIe", *Mathematical Modeling and Supercomputer Technologies: 22nd International Conference*, Revised Selected Papers, 22nd International Conference MMST 2022 (November 14–17, 2022, Nizhny Novgorod, Russia), Springer, Cham, 2022, ISBN 978-3-031-24144-4, pp. 281–292. ↑356

[201] *Nvidia DGX A100*, Datasheet, Nvidia, 2023, 2 pp. ↑356

[202] *Nvidia DGX A100*, User Guide, DU-09821-001_v01, Nvidia, 2023, 126 pp. ↑356

[203] *Nvidia DGX Platform*, Cloud & Data Center, Nvidia, 2024. ↑356

[204] *Nvidia DGX SuperPOD: Scalable infrastructure for AI leadership*, Reference Architecture, RA-09950-001, Nvidia, 2021, 30 pp. ↑356

[205] *Leonardo HPC system*, Technical info, Leonardo Pre-exascale Supercomputer, 2024. ↑357, 413

[206] *Perlmutter architecture*, NERSC documentation, NERSC. ↑357, 413

[207] *Nvidia HPC SDK*, Overview, Containers, Nvidia. ↑357

[208] *Nvidia HPC SDK Version 24.3 Documentation*, Nvidia, 2024. ↑357, 358

[209] *CUDA LLVM Compiler*, Nvidia Developer, Nvidia. ↑358

[210] S. Potluri, K. Hamidouche, A. Venkatesh, D. Bureddy, K. Panda. "Efficient inter-node MPI communication using GPUDirect RDMA for InfiniBand clusters with Nvidia GPUs", *2013 42nd International Conference on Parallel Processing* (01–04 October 2013, Lyon, France), IEEE, 2013, ISBN 978-0-7695-5117-3, pp. 80–89. ↑358

[211] *Nvidia CUDA Fortran programming guide*, HPC SDK documentation, version 24.3, Nvidia, 2024. ↑360, 361

[212] G. Ruetsch, M. Fatica. *CUDA Fortran for scientists and engineers. Best practices for efficient CUDA Fortran programming*, 1st ed., Morgan Kaufmann, 2013, ISBN 978-0-12-416970-8, 338 pp. ↑361

[213] S. Oyanagi. *HPE Cray MPI Update*, Slides, SC'21 ANL MPICH BOF (November 17, 2021), 6 pp. ↑362

[214] *Developing a Linux Kernel Module using GPUDirect RDMA*, v12.4, Nvidia, 2024, 48 pp. ↑362

[215] *MVAPICH: MPI over InfiniBand, Omni-Path, Ethernet/iWARP, RoCE, and Slingshot*, User guide, Version 2.3.7, ed. Dhabaleswar K., NBCL, 2024. URL ↑362

[216] S. Bhalachandra, B. Austin, S. Williams, N. J. Wright. *Understanding the impact of input entropy on FPU, CPU, and GPU power*, 2022, 6 pp. arXiv 2212.08805 ↑363

[217] *Nvidia Docs. Matrix multiplication background*, User's Guide, Deep learning, DU-09799-001_v001, Nvidia, 2023, 17 pp. URL ↑363, 368

[218] D. Mukunoki, K. Ozaki, T. Ogita, T. Imamura. "DGEMM using tensor cores, and its accurate and reproducible versions", *High Performance Computing*, 35th International Conference, ISC High Performance 2020 (June 22–25, 2020, Frankfurt/Main, Germany), Lecture Notes in Computer Science, vol. **12151**, Springer, Cham, 2020, ISBN 978-3-030-50742-8, pp. 230–248. doi ↑363

[219] M. Fasi, N. J. Higham, M. Mikaitis, S. Pranesh. "Numerical behavior of Nvidia tensor cores", *PeerJ Computer Science*, 2021, id. 7e330. doi ↑363

[220] M. Fasi, N. J. Higham, F. Lopez, T. Mary, M. Mikaitis. "Matrix multiplication in multiword arithmetic: error analysis and application to GPU tensor cores", *SIAM Journal on Scientific Computing*, **45**:1 (2023), pp. C1–C19. doi ↑364

[221] S. Li, K. Osawa, T. Hoefler. *Efficient quantized sparse matrix operations on tensor cores*, 2022, 13 pp. arXiv 2209.06979 ↑364

[222] D. Tao, J. Tiuan. *Performance of Sample CUDA Benchmarks on Nvidia Ampere A100 vs Tesla V100*, GitHub Inc., 2021. URL ↑364, 368

[223] *CUDA Samples*, Reference Manual, TRM-06704-001_v11.2, 142 pp. URL ↑364

[224] *All ACCEL Results Published by SPEC*, Standard Performance Evaluation Corporation, 2024. URL ↑364

[225] H. Brunst, S. Chandrasekaran, F. Ciorba, N. Hagerty, R. Henschel, G. Juckeland, J. Li, V. G. M. Vergara, S. Wienke, M. Zavala, *2022 22nd International Symposium on Cluster, Cloud and Internet Computing (CCGrid)* (16-19 May 2022, Taormina, Italy), 2022, ISBN 978-1-6654-9956-9, pp. 675–684. doi arXiv 2203.06751 % ↑364, 414

[226] *All HPC2021 Results Published by SPEC*, SPEC, 2024. URL ↑364, 391

[227] M. Svedin, S. W. D. Chien, G. Chikafa, N. Jansson, A. Podobas. "Benchmarking the Nvidia GPU lineage: From early K80 to modern A100 with asynchronous memory transfers", *Proceedings of the 11th International Symposium on Highly Efficient Accelerators and Reconfigurable Technologies* (June 21–23, 2021, Online Germany), ACM, New York, 2021, ISBN 978-1-4503-8549-7, id. 9, 6 pp. doi arXiv 2106.04979 ↑364

[228] L. Zhang, M. Wahib, P. Chen, J. Meng, X. Wang, T. Endo, S. Matsuoka. *Persistent kernels for iterative memory-bound GPU applications*, 2022. arXiv 2204.02064 ↑365

[229] M. Špeťko, O. Vysocký, B. Jansík, L. Říha. “DGX-A100 face to face DGX-2—performance, power and thermal behavior evaluation”, *Energies*, **14**:2 (2021), pp. 376. doi ↑365

[230] B. Jansik. *Mandelbrot benchmark*, Accessed 15.10.2023. URL ↑365

[231] D. Mudigere, Y. Hao, J. Huang, Z. Jia, A. Tulloch, S. Sridharan, X. Liu, M. Ozdal, J. Nie, J. Park, L. Luo, J. A. Yang, L. Gao, D. Ivchenko, A. Basant, Y. Hu, J. Yang, E. K. Ardestani, X. Wang, R. Komuravelli, H. Ching-Chu, S. Yilmaz, H. Li, J. Qian, Z. Feng, Y. Ma, J. Yang, E. Wen, H. Li, L. Yang, C. Sun, W. Zhao, D. Melts, K. Dhulipala, R. KKishore, T. Graf, A. Eisenman, K. K. Matam, A. Gangidi, G. J. Chen, M. Krishnan, A. Nayak, K. Nair, B. Muthiah, M. Khorashadi, P. Bhattacharya, P. Lapukhov, M. Naumov, A. Mathews, L. Qiao, M. Smelyanskiy, B. Jia, V. Rao. “Software-hardware co-design for fast and scalable training of deep learning recommendation models”, *Proceedings of the 49th Annual International Symposium on Computer Architecture*, ISCA ’22 (June 18–22, 2022, New York, USA), ACM, New York, 2022, ISBN 978-1-4503-8610-4, pp. 993–1011. doi arXiv 2104.05158 ↑365, 367, 368, 369, 379

[232] T. Deakin, J. Price, M. Martineau, S. McIntosh-Smith. “Evaluating attainable memory bandwidth of parallel programming models via BabelStream”, *International Journal of Computational Science and Engineering*, **17**:3 (2018), pp. 247–262. doi URL ↑365

[233] J. D. McCalpin. *Memory bandwidth and machine balance in current high performance computers*, IEEE computer society technical committee on computer architecture (TCCA) newsletter, 1995, 7 pp. URL ↑365

[234] Y. M. Tsai, T. Cojean, H. Anzt. “Evaluating the performance of Nvidia’s A100 Ampere GPU for sparse and batched computations”, *2020 IEEE/ACM Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS)* (12 November 2020, GA, USA), IEEE, 2020, ISBN 978-0-7381-1048-6, pp. 26–38. doi arXiv 2008.08478 ↑366, 367, 368, 370, 371, 372

[235] Y. M. Tsai, T. Cojean, H. Anzt. *Evaluating the performance of Nvidia’s A100 Ampere GPU for sparse linear algebra computations*, 2020, 9 pp. arXiv 2008.08478 ↑366, 367, 368, 372

[236] J. R. Hammond, T. Deakin, J. Cownie, S. McIntosh-Smith. “Benchmarking Fortran DO CONCURRENT on CPUs and GPUs using BabelStream”, *2022 IEEE/ACM International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS)* (13-18 November 2022, Dallas, TX, USA), IEEE, 2022, ISBN 978-1-6654-5185-7, pp. 82–99. doi ↑366, 367, 414

[237] C. J. Balos. “Reproduced computational results report for “Ginkgo”: a modern linear operator algebra framework for high performance computing"”, *ACM Transactions on Mathematical Software (TOMS)*, **48**:1 (2022), id. 3, 7 pp. doi ↑367, 368

[238] T. Grützmacher, H. Anzt, E. S. Quintana-Ortí. "Using Ginkgo's memory accessor for improving the accuracy of memorybound low precision BLAS", *Software: Practice and Experience*, **53**:1, Special Issue: New Trends in High-Performance Computing: Software Systems and Applications (2021), pp. 81–98. doi ↑367, 368, 371, 372

[239] *Mixbench*, A benchmark suite for GPUs on mixed operational intensity kernels, Openbenchmarking.org, Phoronix Media, 2024. URL ↑367

[240] T. Dong, A. Haidar, P. Luszczek, S. Tomov, A. Abdelfattah, J. Dongarra. "Magma batched: A batched blas approach for small matrix factorizations and applications on gpus", *Journal of Latex class files*, **14**:8 (2015), Accessed 15.10.2023. URL ↑367

[241] A. Abdelfattah, S. Tomov, J. Dongarra. "Batch QR factorization on GPUs: design, optimization, and tuning", Computational Science – ICCS 2022 (June 21–23, 2022, London, UK), Lecture Notes in Computer Science, vol. **13350**, Springer, Cham, 2022, ISBN 978-3-031-08750-9, pp. 60–74. doi ↑368, 371, 414, 415

[242] A. Abdelfattah, V. Barra, N. Beams, R. Bleile, J. Brown, S. Jean-Camier, R. Carson, N. Chalmers, V. Dobrev, Y. Dudouit, P. Fischer, A. Karakus, S. Kerkemeier, T. Kolev, H. Yu-Lan, E. Merzari, M. Min, M. Phillips, T. Rathnayake, R. Rieben, T. Stitt, A. Tomboulides, S. Tomov, V. Tomov, A. Vargas, T. Warburton, K. Weiss. "GPU algorithms for efficient exascale discretizations", *Parallel Computing*, **108** (2021), id. 102841, 10 pp. doi ↑368, 369, 416

[243] T. Dong, V. Dobrev, T. Kolev, R. Rieben, S. Tomov, J. Dongarra. "A step towards energy efficient computing: Redesigning a hydrodynamic application on CPU-GPU", *2014 IEEE 28th International Parallel and Distributed Processing Symposium* (19–23 May 2014, Phoenix, AZ, USA), IEEE, 2014, ISBN 978-1-4799-3800-1, pp. 972–981. doi ↑369

[244] A. Abdelfattah, M. Baboulin, V. Dobrev, J. Dongarra, C. Earl, J. Falcou, A. Haidar, I. Karlin, T. Kolev, I. Masliah, S. Tomov. "High-performance tensor contractions for GPUs", *Procedia Computer Science*, **80** (2016), pp. 108–118. doi ↑369

[245] A. Heinecke, G. Henry, M. Hutchinson, H. Pabst. "LIBXSMM: accelerating small matrix multiplications by runtime code generation", *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC'16 (13–18 November 2016, Salt Lake City, UT), IEEE, 2016, ISBN 978-1-4673-8815-3, pp. 981–991. doi ↑369

[246] I. Bethune, F. Reid, A. Lazzaro. *CP2K Performance from Cray XT3 to XC30*, Cray User Group (CUG), 2014, 11 pp. URL ↑369

[247] A. Sedova, A. Tharrington, B. Messer. *Portability in scientific computing: The molecular dynamics non-bonded forces calculation as a case study*, 2018, 25 pp. URL ↑369

[248] H. Waugh, S. McIntosh-Smith. "On the use of BLAS libraries in modern scientific codes at scale", *Driving Scientific and Engineering Discoveries Through the Convergence of HPC, Big Data and AI*, 17th Smoky Mountains Computational Sciences and Engineering Conference SMC 2020 (August 26–28, 2020, Oak Ridge, TN, USA), Communications in Computer and Information Science, vol. **1315**, Springer, Cham, 2020, ISBN 978-3-030-63392-9, pp. 67–79. doi ↑369

[249] N. Mijić, D. Davidović. "Batched matrix operations on distributed GPU's with application in theoretical physics", *2022 45th Jubilee International Convention on Information, Communication and Electronic Technology (MIPRO)* (23–27 May 2022, Opatija, Croatia), IEEE, 2022, ISBN 978-953-233-103-5, pp. 293–299. doi arXiv 2203.09353 ↑370

[250] C. Stylianou, M. Weiland. *Optimizing sparse linear algebra through automatic format selection and machine learning*, 2023, 10 pp. arXiv 2303.05098 ↑372

[251] V. R. Pascuzzi, M. Goli. "Benchmarking a proof-of-concept performance portable SYCL-based fast Fourier transformation Library", *Proceedings of the 10th International Workshop on OpenCL*, International Workshop on OpenCL IWOCL'22 (May 10–12, 2022, Bristol, United Kingdom), ACM, New York, 2022, ISBN 978-1-4503-9658-5, id. 20, 9 pp. doi arXiv 2203.09384 ↑372, 414

[252] D. Tolmachev. "VkFFT-a performant, cross-platform and open-source GPU FFT library", *IEEE Access*, **11** (2023), pp. 12039–12058. doi ↑372

[253] B. Li, S. Cheng, J. Lin. *tcFFT: Accelerating half-precision FFT through tensor cores*, 2021, 10 pp. arXiv 2104.11471 ↑372, 373

[254] N. Hagerty, Vergara V. Melesse, A. Tharrington. "Studying performance portability of LAMMPS across diverse GPU-based platforms", S2 World 2020. CUG 2021 & 2022. PN_HCP. HeteroPar 2022, *Concurency and computation. Practice and Experience*, **35**:28 (2023), id. e7895. doi ↑373, 396, 424

[255] A. Poenaru, W.-C. Lin, S. McIntosh-Smith. "A performance analysis of modern parallel programming models using a compute-bound application", *ISC High Performance 2021: High Performance Computing*, Lecture Notes in Computer Science, vol. **12728**, Springer, Cham, 2021, ISBN 978-3-030-78712-7, pp. 332–350. doi ↑373

[256] L. Solis-Vasquez, A. F. Tillack, D. Santos,-Martins, A. Koch, S. Le,Grand, S. Forli. "Benchmarking the performance of irregular computations in AutoDock-GPU molecular docking", *Parallel Computing*, **109** (2022), id. 102861, 12 pp. doi ↑373

[257] M. Manathunga, H. M. Aktulga, A. W. Götz, K. M. Merz. "Quantum mechanics/molecular mechanics simulations on Nvidia and AMD graphics processing units", *J. Chem. Inf. Model.*, **63**:3 (2023), pp. 711-717. doi ↑374

[258] V. W. D. Cruzeiro, M. Manathunga, K. M. Merz, A. W. Götz. "Open-source multi-GPU-accelerated QM/MM simulations with AMBER and QUICK", *J. Chem. Inf. Model.*, **61**:5 (2021), pp. 2109–2115. doi ↑374

[259] A. Shajan, M. Manathunga, A. W. Götz, K. M. Jr. Merz. “Geometry optimization: A comparison of different open-source geometry optimizers”, *J. Chem. Theory Comput.*, **19**:21 (2023), pp. 7533–7541. doi ↑374

[260] D. B. Williams-Young, A. Asadchev, D. T. Popovici, D. Clark, J. Waldrop, T. Windus, E. F. Valeev, W. A. de Jong. “Distributed memory, GPU accelerated Fock construction for hybrid, Gaussian basis density functional theory”, *The Journal of Chemical Physics*, **158**:23 (2023), id. 234104. doi ↑374

[261] I. Kim, D. Jeong, J. Won-Son, J. Hyung-Kim, Y. M. Rhee, Y. Jung, H. Choi, J. Yim, I. Jang, D. S. Kim. “Kohn-Sham time-dependent density functional theory on the massively parallel GPUs”, *npj Computational Materials*, **9** (2023), id. 81, 12 pp. doi ↑374, 375

[262] A. Siegel, E. Draeger, J. Deslippe, T. M. Evans, M. Francois, T. Germann, D. Martin, W. Hart. *Application Results on Early Exascale Hardware*, No. ORNL/TM-2022/2437, Oak Ridge National Lab.(ORNL), Oak Ridge, TN (US), 2022. URL ↑375, 404, 412, 413, 415, 417, 418, 424, 425

[263] G. Dang, S. Liu, T. Guo, J. Dang, X. Li. “Direct numerical simulation of compressible turbulence accelerated by graphics processing unit: An open-source high accuracy accelerated computational fluid dynamic software”, *Physics of Fluids*, **34**:12 (2022), id. 126106. doi ↑375

[264] M. Min, A. Tombouldies. *Simulating Atmospheric Boundary Layer Turbulence with Nek5000/RS*, No. ANL-22/79, Argonne National Lab.(ANL), Argonne, IL, 2022, 34 pp. ↑375

[265] P. Fischer, S. Kerkemeier, M. Min, H. Yu-Lan, M. Phillips, T. Rathnayake, E. Merzari, A. Tomboulides, A. Karakus, N. Chalmers, T. Warburton. “NekRS, a GPU-accelerated spectral element Navier–Stokes solver”, *Parallel Computing*, **114** (2022), id. 102982, 13 pp. doi ↑376, 416

[266] *FUN3D is a Computational Fluid Dynamics (CFD) suite of tools actively developed at NASA that benefits Aeronautics, Space Technology, and Exloration by modeling fluid flow*, 14.0.2-16d1333, NASA Official: David P. Lockard. URL ↑376

[267] G. Nastac, A. Walden, E. J. Nielsen, K. Frendi. “Implicit thermochemical nonequilibrium flow simulations on unstructured grids using GPUs”, AIAA Scitech 2021 Forum (11–15, 19-21 January 2021, virtual event), 2021, id. AIAA 2021-0159. doi ↑376

[268] G. Nastac, A. Walden, L. Wang, E. J. Nielsen, Y. Liu, M. Opgenorth, J. Orender, M. Zubair. “A multi-architecture approach for implicit computational fluid dynamics on unstructured grids”, AIAA Scitech 2023 Forum (23–27 January 2023, National Harbor, MD & Online), 2023, id. AIAA 2023-1226. doi ↑376, 420

[269] V. Pasquariello, Y. Bunk, S. Eberhardt, H. Pei-Huang, J. Matheis, M. Ugolotti, S. Hickel. “GPU-accelerated simulations for eVTOL aerodynamic analysis”, AIAA Scitech 2023 Forum (23–27 January 2023, National Harbor, MD & Online), 2023, id. AIAA 2023-2107. doi ↑376

[270] T. Regev, J. Nestmann, A. Garzuzi, D. Greenblatt, S. Frankel. “GPU-accelerated high-fidelity implicit large eddy simulations of coanda cylinder flow instabilities”, AIAA Scitech 2023 Forum (23–27 January 2023, National Harbor, MD & Online), 2023, id. AIAA 2023-0272. ↑376

[271] H. C. V. Kakumani, A. S. Chamarthi, N. Hoffmann, S. H. Frankel. “GPU-accelerated numerical study of temperature effects in choked under-expanded supersonic jets”, AIAA SCITECH 2023 Forum (23–27 January 2023, National Harbor, MD & Online), 2023, id. AIAA 2023-0976. ↑376

[272] J. Sitaraman, D. Jude. “Development of GPGPU capable multi-solver overset methods”, AIAA SCITECH 2023 Forum (23–27 January 2023, National Harbor, MD & Online), 2023, id. AIAA 2023-0042. ↑376

[273] M. Mortazawy, M. Rao, J. Jilesen, D. Work, R. Shock. *Early Stage Vehicle Aerodynamics Development using a GPU Based LBM CFD Solver*, SAE Technical Paper No2023-01-0560, 2003, 7 pp. ↑377

[274] A. Kummerländer, M. Dorn, M. Frank, M. J. Krause. “Implicit propagation of directly addressed grids in lattice Boltzmann methods”, *Concurrency and Computation: Practice and Experience*, **35**:8 (2023), id. e7509. ↑377

[275] De Vanna F., F. Avanzi, M. Cogo, S. Sandrin, M. Bettencourt, F. Picano, E. Benini. “URANOS: A GPU accelerated Navier-Stokes solver for compressible wall-bounded flows”, *Computer Physics Communications*, 2023, id. 108717, 18 pp. ↑377

[276] H. Chandravamsi, A. S. Chamarthi, N. Hoffmann, S. H. Frankel. “On the application of gradient based reconstruction for flow simulations on generalized curvilinear and dynamic mesh domains”, *Computers & Fluids*, 2023, id. 105859, 28 pp. ↑377

[277] P. Mattson, C. Cheng, C. Coleman, G. Diamos, P. Micikevicius, D. Patterson, H. Tang, Y. Gu-Wei, P. Bailis, V. Bittorf, D. Brooks, D. Chen, D. Dutta, U. Gupta, K. Hazelwood, A. Hock, X. Huang, A. Ike, B. Jia, D. Kang, D. Kanter, N. Kumar, J. Liao, G. Ma, D. Narayanan, T. Oguntebi, G. Pekhimenko, L. Pentecost, V. J. Reddi, T. Robie, T. S. John, T. Tabaru, J. Carole-Wu, L. Xu, M. Yamazaki, C. Young, M. Zaharia. “MLPerf training benchmark”, *Proceedings of Machine Learning and Systems*. V. 2, MLSys 2020, eds. I. Dhillon, D. Papailiopoulos, V. Sze, 2020, pp. 336–349. arXiv 1910.01500 ↑377

[278] *MLPerf Training v.2.1 Results*, ML Commons. ↑377, 378, 392

[279] *MLPerf Training HPC v2.0 results*, ML Commons. ↑378

[280] *Nvidia NVLink and NVLink Switch*, Cloud & Data Center, Nvidia. ↑381

[281] *PRE-EOS 128 NODE DGX SuperPOD — Nvidia DGX H100, Xeon Platinum 8480C 56C 2GHZ, Nvidia H100 Tensor core GPUs, Nvidia ConnectX-7 NDR 400G Infiniband*, Top500.org, 2023. ↑383

[282] *Kestrel System Configuration*, National Renewable Energy Laboratory Computing Systems, Alliance for Sustainable Energy, 2023. ↑383

[283] *G242-P36 (rev. 100)*, Products, GIGA-BYTE Technology, 2024. URL ↑383

[284] *Nvidia Grace CPU Superchip Whitepaper*, V1.1, Nvidia, 2024, 20 pp. URL ↑384, 385, 386

[285] *Nvidia Grace CPU Superchip*, Datasheet, Nvidia, 2024, 3 pp. URL URL ↑384

[286] *Arm Neoverse V2 Core Technical Reference Manual*, Accessed 15.10.2023. URL ↑384

[287] *Nvidia GH200 Grace Hopper Siperchip Architecture*, V1.01, Nvidia, 2024, 39 pp. URL ↑386, 387, 388, 389

[288] *H263-V11*, Products, rev. LAW1, Giga-Byte Technology. URL ↑388

[289] H. Petty, I. Goldwasser, P. Desale. *One Giant Superchip for LLMs, Recommenders, and GNNs: Introducing Nvidia GH200 NVL32*, Technical blog, Nvidia developer, 2023. URL ↑389

[290] *Nvidia BlueField-3 networking platform*, Datasheet, Nvidia, 2023, 2 pp. URL ↑389

[291] *CUDA Python Manual*, v. 12.4.0, Nvidia, 2024. URL ↑389

[292] *Nvidia NVVM IR Specification*, v. 12.4, Nvidia, 2024, 80 pp. URL ↑390

[293] J. Choquette. "Nvidia Hopper GPU: scaling performance", *2022 IEEE Hot Chips 34 Symposium (HCS)* (21–23 August 2022, Cupertino, CA, USA), IEEE, 2022, ISBN 978-1-6654-6028-6, pp. 1–46. doi ↑391

[294] *Amber22: pmemd.cuda performance information*, The Amber project, ed. Kollman P., 2023. URL ↑392

[295] *MLPerf Training v.3.0 Results*, MLCommons, 2024. URL ↑392, 394, 395

[296] *AMD Instinct$^{TM}$ MI100 Accelerators*, Overview, AMD, 2020. URL ↑396

[297] *AMD Instinct$^{TM}$ Accelerators*, Products, AMD. URL ↑396

[298] *MLPerf Inference Datacenter v.3.1 Results*, MLCommons. URL ↑392

[299] A. Smith, N. James. "AMD Instinct$^{TM}$ MI200 series accelerator and node architectures", *2022 IEEE Hot Chips 34 Symposium (HCS)* (21–23 August 2022, Cupertino, CA, USA), IEEE, 2022, ISBN 978-1-6654-6028-6, 23 pp. doi ↑396, 400, 402, 403, 404, 406, 407, 408

[300] *AMD Instinct$^{TM}$ MI210 Accelerators*, AMD, 2022. URL ↑399, 405

[301] *AMD Instinct$^{TM}$ MI250 drivers & support*, Accessed 15.10.2023. URL ↑399, 405

[302] *AMD Instinct$^{TM}$ MI250X drivers & support*, Accessed 15.10.2023. URL ↑399, 405

[303] *Frontier user guide*, Oak Ridge National Laboratory, 2024. URL ↑399, 406, 408, 413

[304] *Introducing AMD CDNA architecture*, AMD, 2020, 11 pp. URL ↑399, 402

[305] *Introducing AMD CDNA 2 Architecture*, White paper, AMD Instinct MI200, Advanced Micro Device, 2021, 17 pp. URL ↑399, 400, 401, 402, 406, 407, 408

[306] *"AMD Instinct MI200" instruction set architecture*, Reference Guide, AMD Instinct MI200, Advanced Micro Devices, 2021, 275 pp. URL ↑402

[307] C. Sitaraman, N. Chalmers, N. Malaya, D. McDougal, O. O'Reilly, R. van Oostrum. *AMD matrix cores*, GPU open, AMD Labs notes, ed. Greathouse R., Advanced Micro Devices, 2023. URL ↑402

[308] C. Pearson. *Interconnect bandwidth heterogeneity on AMD MI250x and Infinity fabric*, 2023. arXiv 2302.14827 ↑407

[309] M. Gates, A. YarKhan, D. Sukkari, K. Akbudak, S. Cayrols, D. Bielich, A. Abdelfattah, M. A. Farhan, J. Dongarra. "Portable and efficient dense linear algebra in the beginning of the exascale era", *2022 IEEE/ACM International Workshop on Performance, Portability and Productivity in HPC (P3HPC)* (13–18 November 2022, Dallas, TX, USA), IEEE, 2022, ISBN 978-1-6654-6021-7, pp. 36–46. doi ↑406

[310] V. G. Melesse Vergara, R. D. Budiardja, M. J. Davis, M. A. Ezell, J. A. Hanley, C. J. Zimmer, M. J. Brim, W. R. Elwasif, D. T. Dietz. "Approaching the final Frontier: lessons learned from the deployment of HPE/Cray EX Spock and Crusher supercomputers", *Cray User Group 2022 Proceedings*, CUG (May 2, 2022 – May 5, 2022), Oak Ridge National Lab. (ORNL), 2022. ↑408

[311] *AMD Instinct Accelerator Qualified Servers Q4 2022*, Reference Guide, AMD Instinct MI200, Advanced Micro Devices, 2022, 3 pp. URL ↑408

[312] *Welcome to AMD ROCm Platform*, Revision e2b73a17, Advanced Micro Devices, 2021. URL ↑408, 409

[313] *AMD ROCm documentation*, v. 6.1.1, 2024. URL ↑409, 410, 411

[314] N. Kondratyuk, V. Nikolskiy, D. Pavlov, V. Stegailov. "GPU-accelerated molecular dynamics: State-of-art software performance and porting from Nvidia CUDA to AMD HIP", *The International Journal of High Performance Computing Applications*, **35**:4 (2021), pp. 312–324. doi ↑409, 423

[315] D. Charrier et al.. *GPUFORT: S2S translation tool for CUDA Fortran and Fortran+X in the spirit of hipify*, AMD ROCm Software, GitHub Inc.. URL ↑410

[316] *AMD ROCm documentation*, AMD, 2024. URL ↑410

[317] K. S. Khorassani, C. Chen-Chen, B. Ramesh, A. Shafi, H. Subramoni, D. K. Panda. "High Performance MPI over the Slingshot Interconnect", *Journal of Computer Science and Technology*, **38**:1 (2023), pp. 128–145. doi ↑411

[318] *Welcome to the LUMI supercomputer user guide*, LUMI (Large Unified Modern Infrastructure) consortium. URL ↑413

[319] *Crusher Quick-Start Guide*, Oak Ridge National Laboratory, 2024. URL ↑413

[320] *Spock Quick-Start Guide*, Oak Ridge National Laboratory, 2023. URL ↑413

[321] *ThetaGPU Machine Overview*, Argonne National Laboratory, Accessed 15.10.2023. URL ↑413

[322] *Polaris Machine Overview*, Argonne National Laboratory, 2023. URL ↑413

[323] *Summit User Guide*, Alpine, Oak Ridge National Laboratory, 2023. URL ↑413

[324] M. A. Herou et al.x. *ECP software technology capability assessment report*, No. ORNL/TM-2022/2651, V3.0, Oak Ridge National Lab, 2022, 237 pp. URL ↑413, 422

[325] S. Sathyanarayana, M. Bernardini, D. Modesti, S. Pirozzoli, F. Salvadore. *High-speed turbulent flows towards the exascale: STREAmS-2 porting and performance*, 2023, 32 pp. arXiv 2304.05494 ↑414, 429, 430, 431

[326] A. Müller, B. Schmidt, R. Membarth, R. Leißa, S. Hack. *AnySeq/GPU: A novel approach for faster sequence alignment on GPUs*, 2022, 11 pp. arXiv 2205.07610 ↑415

[327] M. Manathunga, H. M. Aktulga, A. W. Götz, K. M. Merz jr. "Quantum mechanics/molecular mechanics simulations on Nvidia and AMD Graphics Processing Units", *Journal of Chemical Information and Modeling*, **63**:3 (2023), pp. 711–717. doi ↑416, 417

[328] T. Kolev, P. Fischer, A. P. Austin, A. T. Barker, N. Beams, J. Brown, S. Jean-Camier, N. Chalmers, V. Dobrev, Y. Dudouit, L. Ghaffary, S. Kerkemeier, H. Yu-Lan, E. Merzari, M. Min, W. Pazner, T. Rathnayake, M. S. Shephard, M. H. Siboni, C. W. Smith, J. L. Thompson, S. Tomov, T. Warburton. *High-order algorithmic developments and optimizations for large-scale GPU-accelerated simulations*, ECP Milestone Report: WBS 2.2.6.06, Milestone CEED-MS36, 2021, 51 pp. URL ↑417, 437

[329] M. Thavappiragasam, W. Elwasif, A. Sedova. *Portability for GPU-accelerated molecular docking applications for cloud and HPC: can portable compiler directives provide performance across all platforms?*, 2022, 10 pp. arXiv 2203.02096 ↑418

[330] C. P. Stone, A. Walden, M. Zubair, J. Nielsen. "Accelerating unstructured-grid CFD algorithms on Nvidia and AMD GPUs", *2021 IEEE/ACM 11th Workshop on Irregular Applications: Architectures and Algorithms (IA3)* (15 November 2021, St. Louis, MO, USA), 2021, ISBN 978-1-6654-1126-4, pp. 19–26. doi ↑418

[331] S. Pumma, A. Vishnu. "Semantic-aware lossless data compression for Deep Learning Recommendation Model (DLRM)", *2021 IEEE/ACM Workshop on Machine Learning in High Performance Computing Environments (MLHPC)* (15 November 2021, St. Louis, MO, USA), IEEE, 2021, ISBN 978-1-6654-1124-0, pp. 1–8. doi ↑418

[332] F. Han, N. Kumar. *HPC Application Performance on Dell PowerEdge R750xa Servers with the AMD Instinct MI210 Accelerator*, Dell, 2022. URL ↑419

[333] *AMD Instinct MI200 Series Accelerators Benchmarks*, AMD, 2024. URL ↑419, 423, 425, 426, 429

[334] Y. Yu, C. Cai, J. Wang, Z. Bo, Z. Zhu, H. Zheng. "Uni-dock: Gpu-accelerated docking enables ultralarge virtual screening", *Journal of Chemical Theory and Computation*, **19**:11 (2023), pp. 3336–3345. doi ↑419

[335] Y. Hao, X. Zhao, B. Bao, D. Berard, W. Constable, A. Aziz, X. Liu. *TorchBench: benchmarking PyTorch with High API surface coverage*, 2023, 13 pp. arXiv 2304.14226 ↑420

[336] K. Punniyamurthy, B. M. Beckmann, K. Hamidouche. *Optimizing distributed ML communication with fused computation-collective operations*, 2023, 12 pp. arXiv 2305.06942 ↑420

[337] Y. Guo, L. Lu, S. Zhu. "Novel accelerated methods for convolution neural network with matrix core", *The Journal of Supercomputing*, **79**:17 (2023), pp. 19547–19573. ↑420

[338] A. Eassa, C. Porter. *Fueling high-performance computing with full-stack innovation*, Technical blog, Nvidia developer, 2022. ↑420

[339] *Driving the Industry into the Exascale Era with AMD Instinct Accelerators*, AMD Community, AMD, 2022. ↑420

[340] R. D. Budiardja, M. Berrill, M. Eisenbach, G. R. Jansen, W. Joubert, S. Nichols, D. M. Rogers, A. Tharrington, O. E. B. Messer,. "Ready for the Frontier: preparing applications for the world's first exascale system", *High Performance Computing*, ISC High Performance 2023, LNCS, 13948, Springer, Cham, 2023, ISBN 978-3-031-32040-8, pp. 182–201. ↑421

[341] F. Wittwer, N. K. Sauter, D. Mendez, B. K. Poon, A. S. Brewster, J. M. Holton, M. E. Wall, W. E. Hart, D. J. Bard, J. P. Blaschke. *Accelerating X-ray tracing for exascale systems using Kokkos*, 2022, 6 pp. arXiv 2205.07976 ↑421

[342] T. Papatheodore. *Frontier/Crusher node performance*, Frontier Training Workshop (February 16, 2023), Oak Ridge National Laboratory, 13 pp. ↑422, 423

[343] *MVAPICH: MPI over InfiniBand, Omni-Path, Ethernet/iWARP, RoCE, and Slingshot*, NOWLAB: Network Based Computing Lab, NBCL. ↑422

[344] J. Kurzak, N. Malaya, M. Klemm, E. Hiew. *Matrix Multiply Stress Test*, GitHub inc., 2023. ↑422

[345] T. Papatheodore. *GPU XGEMM*, Benchmark, 2023. ↑422

[346] *Enhancing LAMMPS simulations with AMD instinct accelerators: unleashing performance and scalability*, Solution Brief, High Performance Computing, AMD, 2023, 4 pp. ↑423, 424

[347] *HPC Comes to Life with AMD Instinct GPUs and NAMD*, Solution Brief, High Performance Computing, AMD, 2022, 4 pp. ↑425

[348] S. Pall, A. Alekseenko. "GROMACS 2023: Readiness on the AMD GPU Heterogeneous Platform", *PDC Newsletters*, 2023, no. 1. ↑425

[349] *Molecular Dynamics. Nvidia GPU Benchmarks AMBER 22*, Blog, Exxact, 2023. ↑425, 426

[350] J. Zeng, D. Zhang, D. Lu, P. Mo, Z. Li, Y. Chen, M. Rynik, L. Huang, Z. Li, S. Shi, Y. Wang,; Ye H., P. Tuo, J. Yang, Y. Ding, Y. Li, D. Tisi, Q. Zeng, H. Bao, Y. Xia, J. Huang, K. Muraoka, Y. Wang, J. Chang, F. Yuan, S. L. Bore, C. Cai, Y. Lin, B. Wang, J. Xu, J.-X. Zhu, C. Luo, Y. Zhang, R. E. A. Goodall, W. Liang, A. K. Singh, S. Yao, J. Zhang, R. Wentzcovitch, J. Han, J. Liu, W. Jia, D. M. York, W. E, R. Car, L. Zhang, H. Wang. "DeePMD-kit v2: A software package for deep potential models", *The Journal of chemical physics*, **159**:5 (2023), id. 054801. doi ↑426

[351] A. Prokopenko, P. Sao, D. Lebrun-Grandie. "A single-tree algorithm to compute the Euclidean minimum spanning tree on GPUs", *ICPP '22: Proceedings of the 51st International Conference on Parallel Processing* (29 August 2022–1 September 2022, Bordeaux, France), ACM, New York, 2022, ISBN 978-1-4503-9733-9, id. 14, 10 pp. doi ↑426

[352] A. Bagusetty, A. Panyala, G. Brown, J. Kirk. "Towards cross-platform portability of coupled-cluster methods with perturbative triples using SYCL", *2022 IEEE/ACM International Workshop on Performance, Portability and Productivity in HPC (P3HPC)* (13–18 November 2022, Dallas, TX, USA), IEEE, 2022, ISBN 978-1-6654-6021-7, pp. 81–88. doi ↑427

[353] A. Bussy, O. Schütt, J. Hutter. "Sparse tensor based nuclear gradients for periodic Hartree–Fock and low-scaling correlated wave function methods in the CP2K software package: A massively parallel and GPU accelerated implementation", *The Journal of Chemical Physics*, **158**:16 (Apr 28 2023), id. 164109. doi ↑428

[354] L. Mazur, D. Bollweg, D. A. Clarke, L. Altenkort, O. Kaczmarek, R. Larsen, T. Hai-Shu, J. Goswami, P. Scior, H. Sandmeyer, M. Neumann, H. Dick, S. Ali, J. Kim, C. Schmidt, P. Petreczky, S. Mukherjee. "SIMULATeQCD: A simple multi-GPU lattice code for QCD calculations", *Computer Physics Communications*, **300** (July 2024), id. 109164. doi ↑428

[355] S. Gottlieb, H. Jeong, A. Strelchenko. *Two-link staggered quark smearing in QUDA*, 2023, 10 pp. arXiv 2301.05518 ↑429

[356] P. Mullowney, S. Thomas, A. K. Carr, K. Swirydowicz, M. Day, L. Esclapez. *Novel solver algorithms for nearly singular linear systems arising in combustion modelling*, NREL/PR-2C00-81907, 2022 SIAM Conference on Parallel Processing for Scientific Computing (February 23, 2022), National Renewable Energy Lab. (NREL), 2022. URL ↑429

[357] R. Halver, C. Junghans, G. Sutmann. "Using heterogeneous GPU nodes with a Cabana-based implementation of MPCD", *Parallel Computing*, **117** (September 2023), id. 103033. doi ↑429

[358] M. Min, M. Brazell, A. Tomboulides, M. Churchfield, P. Fischer, M. Sprague. *Towards exascale for wind energy simulations*, 2022, 16 pp. arXiv 2210.00904 ↑432, 433

[359] T. Kolev, P. Fischer, A. Abdelfattah, N. Beams, J. Brown, S. Jean-Camier, R. Carson, N. Chalmers, V. Dobrev, Y. Dudouit, L. Ghaffari, A. Y. Joshi, S. Kerkemeier, Y.-H. Lan, D. Mc Dougall, D. Medina, M. Min, A. Mishra, W. Pazner, M. Phillips, T. Ratnayaka, M. S. Shephard, M. H. Siboni, C. W. Smith, J. L. Thompson, A. Tomboulides, S. Tomov, V. Tomov, T. Warburton. *High-order algorithmic developments and optimizations for more robust exascale applications*, ECP Milestone Report. WBS 2.2.6.06, Milestone CEED-MS38, 2022, 76 pp. URL ↑432, 433, 434

[360] G. R. J. Lesur, S. Baghdadi, G. Wafflard-Ernandez, J. Mauxion, C. M. T. Robert, M. Van den Bossche. "IDEFIX: a versatile performance-portable Godunov code for astrophysical flows", *Astronomy and Asrtrophysics*, **677** (September 2023), id. A9, 17 pp. doi ↑434, 435

[361] C. J. White, P. D. Mullen, F. Yan-Jiang, S. W. Davis, J. M. Stone, V. Morozova, L. Zhang, *The Astrophysical Journal*, **949**:2 (2023), id. 103, 29 pp. doi ↑435

[362] P. Grete, J. C. Dolence, J. M. Miller, J. Brown, B. Ryan, A. Gaspar, F. Glines, S. Swaminarayan, J. Lippuner, C. J. Solomon, G. Shipman, C. Junghans, D. Holladay, J. M. Stone, L. F. Roberts. "Parthenon—a performance portable block-structured adaptive mesh refinement framework", *The International Journal of High Performance Computing Applications*, **37**:5 (2023), pp. 465–486. doi ↑435

[363] N. Schild, M. Räth, S. Eibl, K. Hallatschek, K. Kormann. "A performance portable implementation of the semi-Lagrangian algorithm in six dimensions", *Computer Physics Communications*, **295** (February 2024), id. 108973. doi ↑435

[364] I. Sfiligoi, E. A. Belli, J. Candy, R. D. Budiardja. "Optimization and Portability of a Fusion OpenACC-based Fortran HPC code from Nvidia to AMD GPUs", *PEARC '23: Practice and Experience in Advanced Research Computing* (Portland, OR, USA, July 23–27, 2023), ACM, New York, July 2023, ISBN 978-1-4503-9985-2, pp. 246–250. doi ↑435, 436

[365] S. Diederichs, C. Benedetti, A. Huebl, R. Lehe, A. Myers, A. Sinn, J.-Vay L., W. Zhang, M. Thévenet. "HiPACE++: a portable, 3D quasi-static particle-in-cell code", *Computer Physics Communications*, **278** (September 2022), id. 108421. doi ↑436

[366] *Breaking Barriers in Plasma Physics with PIConGPU and AMD Instinc MI250 GPU*, Solution Brief, High Performance Computing, AMD, 2023, 4 pp. URL ↑436

[367] L. Fedeli, A. Huebl, F. Boillod-Cerneux, T. Clark, K. Gott, C. Hillairet, S. Jaure, A. Leblanc, R. Lehe, A. Myers, C. Piechurski, M. Sato, N. Zaim, W. Zhang, L. Jean-Vay, H. Vincenti. "Pushing the Frontier in the design of laser-based electron accelerators with groundbreaking mesh-refined particle-in-cell simulations on exascale-class supercomputers", *2022 SC22: International Conference for High Performance Computing, Networking, Storage and Analysis (SC)* (Dallas, Texas, USA, November 13–18, 2022), IEEE, 2022, id. 3, 12 pp. doi ↑436

[368] A. Huebl, R. Lehe, E. Zoni, O. Shapoval, R. T. Sandberg, M. Garten, A. Formenti, R. Jambunathan, P. Kumar, K. Gott, A. Myers, W. Zhang, A. Almgren, C. E. Mitchell, J. Qiang, D. Grote, A. Sinn, S. Diederichs, M. Thevenet, L. Fedeli, T. Clark, N. Zaim, H. Vincenti, L. Jean-Vay,. *From compact plasma particle sources to advanced accelerators with modeling at exascale*, 2023, 4 pp. arXiv 2303.12873 ↑436

[369] M. F. Adams, P. Wang, J. Merson, K. Huck, M. G. Knepley. *A performance portable, fully implicit Landau collision operator with batched linear solvers*, 2024, 20 pp. arXiv 2209.03228 ↑436

[370] O. Thawakar, R. M. Anwer, J. Laaksonen, O. Reiner, M. Shah, F. S. Khan. *3D mitochondria instance segmentation with spatio-temporal transformers*, 2023, 10 pp. arXiv 2303.12073 ↑437

[371] D. Samuel, A. Kutuzov, S. Touileb, E. Velldal, Øvrelid L., Rønningstad E., E. Sigdel, A. Palatkina. *NorBench–A benchmark for Norwegian language models*, 2023, 16 pp. arXiv 2305.03880 ↑437

[372] L. Yankovskaya, M. Tars, A. Tättar, M. Fishel. "Machine translation for low-resource Finno-Ugric languages", *Proceedings of the 24th Nordic Conference on Computational Linguistics (NoDaLiDa)* (Tórshavn, Faroe Islands, May 22–24, 2023), University of Tartu Library, 2023, ISBN 978-99-1621-999-7, pp. 762–771. URL ↑437

[373] L. Charpentier, S. Wold, D. Samuel, E. Rønningstad. "BRENT: Bidirectional retrieval enhanced Norwegian transformer", *Proceedings of the 24th Nordic Conference on Computational Linguistics (NoDaLiDa)* (Tórshavn, Faroe Islands, May 22–24, 2023), University of Tartu Library, 2023, ISBN 978-99-1621-999-7, pp. 202–214. URL ↑437

[374] D. Samuel, L. Øvrelid. "Tokenization with factorized subword encoding", *Findings of the Association for Computational Linguistics: ACL 2023* (Toronto, Canada, July 9–14, 2023), ACL, 2023, ISBN 9781959429623, pp. 14143–14161. doi ↑437

[375] *AMD Radeon Instinct MI300*, GPU Specs Database, TechPowerUp. URL ↑438

[376] S. Naffziger, N. Beck, T. Burd, K. Lepak, G. H. Loh, M. Subramony, S. White. "Pioneering chiplet technology and design for the AMD EPYC$^{\text{TM}}$ and Ryzen$^{\text{TM}}$ processor families: Industrial product", *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)* (Valencia, Spain, 14–18 June 2021), IEEE, 2021, ISBN 978-1-4503-9086-6, pp. 57–70. doi ↑439

## Appendix. Abbreviations used in sections of the review (for GPUs from different manufacturers)

### *Abbreviations for hardware*

| | | |
|---|---|---|
| ALU | Arithmetic Logic Unit | *1, 5.1* |
| ASIC | Application-Specific Integrated Circuit | *1* |
| CXL | Compute Exress Link | *Introduction, 2, 3.1* |
| CPU | Central Processing Unit | *Introduction, 1, 2, 3.1, 3.2, 3.3, 4.1, 4.2, 5, 5.1, 5.2, 5.3, 5.4* |
| ECC | Error-correcting Code | *4.1, 4.2* |
| ECP | Exascale Compute Project | *3.1, 4.1, 5.3* |
| EPI | European Processor Initiative | |
| GPU | Graphics Processing Unit | *Introduction, 1, 2, 3, 3.1, 3.2, 3.3, 4, 4.1, 4.2, 4.3, 5, 5.1, 5.2, 5.3, 5.4* |
| GPGPU | General-purpose GPU | *Introduction, 3.1, 4.1, 4.2* |
| HBM | High Bandwidth Memory | *3.1, 4.1, 4.2, 5, 5.3* |
| ISA | Instruction Set Architecture | *1, 3.1, 4.1, 4.2, 5.1* |
| NUMA | Non-Uniform Memory Access | *2, 4.2, 5.1* |
| OoO | Out-of-order | *4.2* |
| PIM | Processing-in-memory | *2, 3.1, 4.1* |
| SIMT | Single Instruction, Multiple Threads | *1, 4.1* |
| SoC | System-on-Chip | *3.1, 4.2* |
| SVE2 | Scalable Vector Extension 2 (ARM) | *4.2* |
| TCO | Total Cost of Ownership | *Introduction, 5.3, 5.4* |
| TDP | Thermal Design Power | *2, 3.1, 3.3, 4.1, 4.2, 5, 5.1* |

### *Abbreviations for mathematical and programming fields*

| | | |
|---|---|---|
| API | Application Programming Interface | *1, 4.1, 5.1, 5.2* |
| BLAS | Basic Linear Algebra Subprograms | *1, 4.1, 5.3* |
| FFT | Fast Fourier Transform | *4.1, 5.3* |
| HPC | High Performance Calculations | *Introduction, 1, 3.1, 3.2, 3.3, 4, 4.1, 4.2, 4.3, 5.1, 5.2, 5.3, 5.4* |

| | | |
|---|---|---|
| HPCG | High Performance Conjugate Gradients | *5.3* |
| HPL | High Performance Linpack | *Introduction, 3.2, 5.3* |
| LLVM | Low Level Virtual Machine (early abbreviation; now — a famous set of compilers and tools) | *4.1, 4.2, 5.2* |
| PGAS | Partitioned Global Address Space | *4.1* |
| SDK | Software Development Kit | *Introduction, 1, 3, 4.1, 5.2, 5.3, 5.4* |

*Abbreviations for areas of use of GPUs and applications that use them*

| | | |
|---|---|---|
| CFD | Computational Fluid Dynamics | *3.2, 4.1, 4.2, 5.2, 5.3* |
| DFT | Density Functional Theory | *Introduction, 4.1, 5.4* |
| DLRM | Deep Learning Recommendation Models | *4.1, 4.2, 5.3* |
| BERT | Bidirectional Encoder Representations from Transformers | *2, 4.2, 5.3* |
| NLP | Natural Language Processing | *2, 4.2* |

*Abbreviations common to GPUs from different companies*

| | | |
|---|---|---|
| SU | Special Function Units | *4.1* |
| OAM | OpenCompute Accelerator Module | *Introduction, 2, 3.1, 5.1, 5.3, 5.4* |

*Abbreviations for Nvidia GPUs*

| | | |
|---|---|---|
| CUDA | Compute Unified Device Architecture | *Introduction, 1, 2, 3.2, 3.3, 4.1, 4.2, 5, 5.1, 5.2, 5.3, 5.4* |
| GPC | GPU (*or* Graphics) Processing Clusters | *2, 4.1, 4.2* |
| GTC | GPU Technology Conference | |
| MIG | Multi-Instance GPU | *2, 4.1, 4.2* |
| NCCL | Nvidia Collective Communications Library | *4.1, 4.2, 5.2* |
| PTX | Parallel Threads execution | *4.1, 4.2, 5.3* |
| SM | Streaming Multiprocessor | *1, 3.1, 4, 4.1, 4.2, 5, 5.1, 5.2, 5.3* |
| SXM | Server PCI Exress Module | *2, 3.1, 3.3, 4.1, 4.2, 5, 5.1, 5.3, 5.4* |
| TPC | Texture Processing Clusters | *4.1, 4.2* |

## *Abbreviations for Intel* GPUs

| | | |
|---|---|---|
| AIC | Add-in Card | *3.1* |
| DPC++ | Data Parallel C++ | *Introduction, 1, 3.2, 3.3, 4.1, 5.2, 5.3, 5.4* |
| DPCT | DPC++ Compatibility Tool | *3.2, 5.2* |
| PVC | Ponte Vecchio | *3, 3.1, 3.2, 3.3, 5.1, 5.4* |

## *Abbreviations for AMD* GPUs

| | | |
|---|---|---|
| CU | Compute Unit | *1, 2, 5, 5.1, 5.2, 5.3* |
| GCD | Graphics Compute Die | *3.1, 5.1, 5.3, 5.4* |
| HIP | Heterogeneous Computing Interface for Portability | *Introduction, 1, 3.2, 3.3, 4.1, 5.1, 5.2, 5.3* |

## *Abbreviations for* GPU *BR100*

| | | |
|---|---|---|
| CU | | *1, 2, 5, 5.1, 5.2, 5.3* |
| SVI | Secure Virtual Instance | *2* |

*Received* 16.10.2023;
*approved after reviewing* 24.01.2024;
*accepted for publication* 01.03.2024;
*published online* 28.06.2024.

Recommended by *prof. Sergey M. Abramov*

## Information about the author:

Mikhail Borisovich Kuzminsky

Senior Researcher, Laboratory of Computer Software for Chemical Research, Candidate of Chemical Sciences, Institute of Organic Chemistry, Russian Academy of Sciences. The scientific interests are high-performance computing, computer hardware, computational chemistry.

iD 0000-0002-3944-8203

**e-mail:** ***kus@free.net***

*The author declare no conflicts of interests.*