ISSN 2079-3316 **PROGRAM SYSTEMS: THEORY AND APPLICATIONS** **vol. 13**, No 1(52), pp. 131–194
Review Article HARDWARE, SOFTWARE AND DISTRIBUTED SUPERCOMPUTER SYSTEMS

UDC 004.272+004.382.2+004.42+004.4'2
10.25209/2079-3316-2022-13-1-131-194

Modern server ARM processors for supercomputers: A64FX and others
Initial data of benchmarks

Mikhail Borisovich **Kuzminsky**✉
Zelinsky Institute of Organic Chemistry of RAS, Moscow, Russia,
kus@free.net✉ (*learn more about the author on p. 194*)

Abstract. A comparative analysis of the performance of ARM server processors used on supercomputers or also aimed at high-performance computing (HPC) is given. Fujitsu A64FX, Marvell ThunderX2 and Huawei Kunpeng 920 were selected for the initial performance analysis. The HPC performance review focuses primarily on benchmarks and applications for the A64FX, which supports longer vectors than other ARM processors and has higher peak performance. The performance of the A64FX is compared against corresponding data for Intel Xeon Skylake and Cascade Lake, and AMD EPYC with Zen 2 and 3 (Roma and Milan), as well as Nvidia V100 and A100 GPUs. A short set of potential pros and cons of the A64FX microarchitecture has been formulated. Comparison of performance data obtained using different compilers for A64FX. Features have been formed when A64FX usually gives advantages in performance over x86-64, and when it concedes to x86-64.

It is clear that the use of A64FX in supercomputers can grow further. There is an assumption that x86-64 hegemony in HPC will decrease, in particular, due to the increased use of server ARM processors. But the analysis of A64FX and new AArch64 processors expected in the near future showed that A64FX will not necessarily lead in this process.

Key words and phrases: ARM, A64FX, x86-64, high performance computing, supercomputers, benchmarks

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

For citation: Kuzminsky M. B. *Modern server ARM processors for supercomputers: A64FX and others. Initial data of benchmarks* // Program Systems: Theory and Applications, 2022, **13**:1(52), pp. 131–194. `http://psta.psiras.ru/read/psta2022_1_131-194.pdf`

Эта статья по-русски: *http://psta.psiras.ru/read/psta2022_1_63-129.pdf*

Introduction

The direction of work indicated in the title of this article becomes very actual. It is associated not only with the growth in the market share of HPC systems occupied by ARM processors [1], but with the recent cardinal changes in the market of corresponding hardware in general, much more radical than, for example, the growth in the market share of server x86-64 processors occupied by AMD [1] with certain delays in the development of semiconductor technology at Intel [2] (although there are statements that 10 nm from Intel is practically equivalent in size to 7 nm from TSMC [1]).

Publications claim that Moore's law is slowing down (see, for example, [1, 3–7]) , and HPC began to pay special attention to power supply (see, for example, [4–7]). The actuality of energy efficiency (performance per Watt) has been taken into account in various HPC applications for a long time. For example, in quantum chemistry — with special attention to ARM processors [8], including a comparison of 64-bit ARM and x86-64 [9]. For supercomputers, energy efficiency is especially important [1].

The area of HPC and supercomputers naturally became the leaders of the expected changes. Accordingly, the relevance of reviews on all computer technology used in HPC also increases. Due to a certain lag in Europe in this area, new initiatives have appeared for the development and production of new ARM processors *EPI (European Processor Initiative)* and a new PRACE infrastructure *(Pan-European High Performance Computing Infrastructure and Services)* has been created, where, among other things, various latest HPC hardware and performance tests data [1, 2, 10]. There is also a similar Russian review, which also considers Russian hardware [11].

In general, server ARM processors or computing systems based on them began to be developed and produced in different regions of the world, including China [12], and the corresponding supercomputers are offered by Cray/HPE and Fujitsu [13].

However, the rate of development of the corresponding computer facilities is very high, and new information on the performance of the software used there quickly appears, which is very relevant for HPC and the optimal hardware selection, including the building of new supercomputers. And the HPC area itself now often includes AI, where it is relevant to work with data types other than the traditional HPC type (double precision, DP) [13]. And information about the performance of the new AArch64 server processors, which began to push x86-64 from their leading position

for HPC, and, in particular, supercomputers, seems to be very relevant, including due to the high energy efficiency of the corresponding ARM processors [11, 14]. Considering also the active promotion of RISC-V [15], the high peak performance of IBM Power10 (which claims to be the leader in DP performance and was expected to be released in 2021) [16], the use of the SW26010 RISC processor in the still top ten TOP500 of the Chinese supercomputer Sunway TaihuLight (see, for example, [17, 18]) we can generally talk about some revival of RISC.

Therefore, the review of performance data in HPC (and, in particular, on supercomputers) of server ARM processors (primarily the most high-performance Fujitsu A64FX) carried out in this work seems to be very relevant.

1. Hardware features (important for HPC) of AArch64 processors

1.1. AArch64 server processors with support of 128-bit vectorization

The first widely known AArch64 processor which began to be used in supercomputers from the TOP500, became, apparently, Marvell ThunderX2 [19], which supports work with 128-bit vectors (Advanced SIMD, NEON). Although other ARM processors were used earlier in supercomputers [20]. Various ThunderX2 performance data is already well known. It combines not very high cost with competitiveness with mass server x86-64 processors in terms of performance for HPC [21]. Even *unconfirmed data*[URL] appeared about higher ThunderX2 performance indicators, for example, relative to Xeon Skylake 6148 — including SPECrate2017_int_peak. However, Xeon Skylake is usually ahead in the performance of ThunderX2, including on the *SPEC OMP2012*[URL] benchmarks or, for example, in various applications of computational chemistry [21], or *UoB-HPC/benchmarks*[URL]. According to [7], ThunderX2 can outperform Xeon Skylake on memory-intensive HPC applications, while giving greater power efficiency, and more computationally intensive applications can read faster on skylake. Computational fluid dynamics (CFD) is a classic area of HPC applications where memory bandwidth is so important [20], while computational chemistry includes, for example, molecular dynamics, which is a computationally intensive area of HPC.

Table 1 compares the main specifications of the ThunderX2 processor that affect performance with other more modern AArch64 processors. With the exception of Fujitsu A64FX, which supports SVE [22], the other of the above in Table 1 ARM processors support the same 128-bit vectorization as

ThunderX2. The number of floating point operations per clock per core of Xeon Scalable of all generations is higher than all these processors; accordingly (in proportion to the clock frequency), the Xeon cores also have higher peak performance (FLOPS). And other AArch64 processors have higher peak performance compared to ThunderX2 (by default, we are talking about working with DP, which is standard for traditional HPC applications). New publications about the performance of ThunderX2 continue to appear (see, for example, [**23**]). The corresponding results are more important for using ThunderX2-based computing systems, since the performance of the newer available AArch64 processors is usually higher, and they are more relevant to purchase. Therefore, given the many published articles on ThunderX2 performance, it is considered in this review only when compared with other newer ARM server processors.

The subsequent ThunderX3 [**24**] and [**25**] processor models that were supposed to be released could surpass ThunderX2 in all the most important parameters, including the number and frequency of cores (see above Table 1). Although ThunderX3 was not supposed to use SVE, because the developers were afraid of its still insufficient maturity for software tools, the number of NEON vector devices was doubled relative to ThunderX2, and the sharply increased peak performance would have surpassed the A64FX. And in ThunderX4 it was planned to switch to work with SVE. However, in mid-2020, Marvell made a statement that could be interpreted as abandoning the release of ThunderX3 and moving to work in other directions than HPC.

Another example of an available 7nm ARM server processor would be the 64-core Huawei Kunpeng 920 [**12**], which was originally intended to be used for HPC, and is now referred to as such processors [**2**, **10**, **11**]. It uses a more modern ARM v8.2-A architecture than ThunderX2 (same as in A64FX, but without the SVE extension). It supports NEON SIMD blocks, just like ThunderX2. The Kunpeng 920 outperforms it in core count and other metrics (see above), including peak performance. Of the hardware features of the Kunpeng 920, we should point out an unusual memory hierarchy compared to Xeon processors, including NUMA in two superclusters of cores (SCCL) inside the Kunpeng chip (see, for example, [**14**, **26**]), as well as the ability to support ccNUMA on a scale chiplets based on them [**12**]. Kunpeng's core grouping is somewhat similar to the A64FX and Zen, and this is probably due to the significant growth in the number of cores common to server processors. However, there are still few publications on the HPC benchmarks of Kunpeng 920.

Table 1. Main spedcifications of ARM processors for HPC

Specifications	A64FX	Kunpeng 920-6426	Thunder X2CN9980	Thunder X3	Altra (Q80)	Graviton2
Peak performance(DP), TFLOPS	2,8-3,4	1,33	$\leqslant 0,64$	$\leqslant 4,75$	2,12	
Technology	TSMC 7 nm	TSMC 7 nm	TSMC 14 nm	TSMC 7 nm	TSMC 7 nm	TSMC 7 nm
Number of cores	48+4	64	32	$\leqslant 96$	80	64
SMP	1(ccNUMA)	$\leqslant 4$ (NUMA)	2/4(NUMA)	1–2 (NUMA)	1–2 (NUMA)	1
Clock, Ghz	1,8–2,2 GGz[1]	2,6 GGz	2,1–2,5 GGz[1]	$\leqslant 3,1$ GGz	2,6–3,3 GGz	2,5 GGz
Simultaneous multithreading	no	no	SMT0/2/4	SMT4	no	no
Vectorization, bit	SVE,512	NEON,128	NEON,128 (2xFMA)	NEON,128 (4xFMA)	NEON,128 (2 devices)	NEON,128
Architecture	ARM v8.2-A +SVE	ARM v8.2-A (Neoverse N1)	ARM v8.1 Vulcan	ARM v8.3+ Triton	ARM v8.2+ (Neoverse N1)	Neoverse N1
L1 cache I+D	3 MB (64 KBx48)	64KB+64KB per core	32+32 KB per core	64+32 KB per core	64+64 KB per core	4+4 MB
L2 cache	32 MB (4x8MB)	32 MB (512 KB per core)	256 KB per core	512 KB per core	1 MB per core	64 MB
L3 cache	no	64 MB (1 MB per core)	32 MB (1MB per core)	90 MB	$\leqslant 32$ MB	32 MB
Memory interface	HBM2	8xDDR4-2933	8xDDR4-2666	8xDDR4-3200	8xDDR4-3200	8xDDR4-3200
bandwidth	1 TB/c	188 GB/c	171 GB/c	205 GB/c	205 GB/c	205 GB/c
Memory size	32 GB	$\leqslant 2$ TB	$\leqslant 2$ TB x2[2]	no data	$\leqslant 1TB$	$\leqslant 1$ TB
Interface I/O	PCIe-v3 x16 +TofuD	PCIe-v4 40x (up to x16)	PCIe-v3 48x, 56x	16 x PCIe-v4 x4	PCIe-v4 128×	PCIe-v4 64×
TDP, W	No data	180	180/2,2 GGz	100-240	250[3]	80-110

[1]maximum in boost mode; [2]for dual processor server; [3]for 3.3 Ghz.

Table data taken from manufacturer's websites.

TABLE 2. SPECrate 2017 base benchmark data for servers with Kunpeng 920 and x86-64 processors

	Integer	Floating point
dual processor		
Kunpeng 920-7260	318	263
Xeon Skylake 8180	297	276
Xeon Cascade Lake 8280	303	308
EPYC Rome, 7742	701	524
EPYC Milan, 7763	839	651
Four-processor		
Kunpeng 920-7260	628	516
Xeon Skylake 8180	564	523
Xeon Cascade Lake 8280	670	566

The data from the SPECcpu 2017 benchmarks [**27**] should now be considered the most massive. The relevant data and comparison with servers based on modern x86-64 processors are given in Table 2. According to these data, it can be seen that the 64-core Kunpeng 920 7260 with a frequency of 2.6 GHz, winning a little on integer performance, is inferior to the major Xeon Skylake models in floating point calculations.

The maximum data as of 09/18/2021 with the indicated processors is given. More recent Xeon Ice Lake data is not shown here as it is even faster than the Kunpeng 920.

Since floating point is important for traditional HPC applications, we will talk about it by default in the following. The more modern Cascade Lake extends the performance lead over the Kunpeng 920, and the third generation Xeon Scalable is even faster.

SPECcpu 2017 is not often used for performance evaluations for HPC, especially for supercomputers. For the Kunpeng 920, memory bandwidth benchmarks (stream) [**10**] are available in Table 3. These metrics are important for HPC applications where memory bandwidth limits the supported performance (for example, for CFD [**20**]). Kunpeng 920 is ahead of ThunderX2 and Xeon Skylake here, and behind AMD EPYC Roma — which corresponds to the memory supported by these processors (see Table 1 and Table 4).

In a RoCE (100 Gb/s) cluster of 4 dual-processor servers with Kunpeng 920, HPCG benchmarks with MPI parallelization were carried out; data for

Table 3. Comparison of stream benchmark data (GB/s) in dual-socket nodes with Kunpeng 920, ThunderX2 and x86-64 processors1. All bandwidths are in GB/s

Processor	Copy	Scale	Add	Triad
Kunpeng 920-6426	322	322	324	324
ThunderX2/2,5 GGz	∼ 225	∼ 225	∼ 240	∼ 240
EPYC Rome 7742	338	338	340	340
Xeon 8160/2,1 GGz	∼ 180	∼ 180	∼ 200	∼ 200
Xeon 8174/2,4 GGz	∼ 180	∼ 180	∼ 200	∼ 200

[1]The highest results achieved in [10] are given.

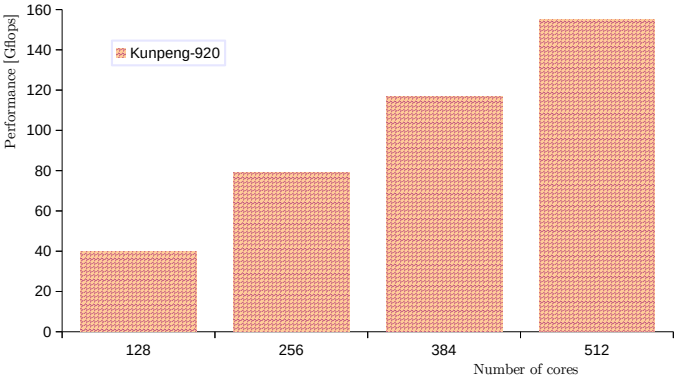


Figure 1. Kunpeng 920 performance in HPCG

1–4 nodes are shown in Figure 1 from [10, Figure 7]. A single-socket server with ThunderX2/2.5 GHz in this test received noticeably less than 20 GFLOPS, and a two-socket server with Kunpeng — about 40 GFLOPS [10], which indicates its higher performance than ThunderX2. A two-socket server with Kunpeng 920 in the HPCG test can be compared with two-socket servers based on 24-core Xeon Skylake: with Xeon Platinum 8160/2.7 GHz, performance is noticeably below 40 GFLOPS, and with Xeon Platinim 8174/2.7 GHz — about 40 GFLOPS [10]. Thus, two-socket servers with Kunpeng 920 proved to be competitive in performance in HPCG with two-socket servers with Xeon Skylake, which has AVX-512 support. This is due to the fact that working with vectors/matrices is not as important in HPCG as it is in HPL.

Table 4. Main features of A64FX and modern x86-64 processors

Features	A64FX	Xeon Platinum 8180	Xeon Platinum 8280	Xeon Platinum 8380	AMD EPYC 7763 (milan)	AMD EPYC 7662 (ROME)
Architecture	ARM-v8.2-A + SVE	Skylake	Cascade Lake	Ice Lake (Sunny Cove)	Zen 3	Zen 2
Technology	TSMC 7 nm	14 nm	14 nm	10 nm	TSMC 7 nm	TSMC 7 nm
Cores number	48+4	28	28 (32[1])	40	64	64
Clock GGz	1,8-2,2	2,5/3,8[2]	2,7/4,0[2]	2,3/3,4[2]	2,45/3,5[2]	2,0/3,3[2]
SMP sockets	1	via UPI up to 8	via UPI up to 8[1]	$\leqslant 8$	1/2	1/2
Multithreading	no	SMT2	SMT2	SMT2	SMT2	SMT2
Vectorization, bit	SVE,512	AVX-512	AVX-512	AVX-512	AVX-256 (AVX2)	AVX-256 (AVX2)
FLOPS/cicle (DP, per 1 core)	32	32	32	32	16	16
L1 cache (I+D)	64KB+64KB per core 4-way	32KB+32KB per core 8-way	32KB+32KB per core 8-way	32KB+48KB per core 8/12-way	32KB+32KB per core 8-way	32KB+32KB per core 8-way
L2 cache	32MB (8MB×4) 16-way	1 MB/core 16-way	1 MB/core 16-way	512 KB/core 8-way	32 MB 8-way	32 MB 8-way
L3 cache	no	38,5 MB 11-way	38,5 MB 11-way	60 MB 16-way	256 MB 16-way	256 MB 16-way
Memory	HBM2	6xDDR4-2666	6xDDR4-2933	8xDDR4-3200	8xDDR4-3200	8xDDR4-3200
	32 GB	$\leqslant$ 768 GB	$\leqslant$ 1 TB	$\leqslant$ 6 TB	$\leqslant$ 4 TB	$\leqslant$ 4 TB
Peak bandwidth[3]	1 TB/c (4x256 GB/c)	128 GB/c	141 GB/c	205 GB/c	205 GB/c	205 GB/c
I/O	PCIe-v3×16 + TofuD	PCIe-v3 48x	PCIe-v3 48x	PCIe-v4 64×	PCIe-v4 128×	PCIe-v4 128×
TDP, W	no data	205	205	270	280	225
Price, $	no data	10010	10010	8100	7890	7000

[1]For Cascade Lake AP 92XX; [2]High values — maximum turbo frequency; [3]peak value

Data for *Intel* and *AMD* processors taken from the sites 5.09.2021. European data *(PRACE)* also used.

It is well known that working with matrices, primarily matrix multiplication (GEMM), often limits the calculation time. Examples include, for example, HPL [28] or quantum chemical calculations using methods that take into account electronic correlations [6]. There are many mathematical libraries supporting BLAS [29]. However, they are developing very quickly, and new libraries also appear, including those for optimization problems on new computer architectures. A comparative analysis of various libraries with BLAS for DGEMM calculations in a dual-processor server with 48-core Kunpeng 920/2.6 GHz showed maximum performance in OpenBLAS, where DGEMM was then optimized by explicitly taking into account the peculiarities of working with NUMA, which gave a gain in operation with large matrices. Maximum performance was achieved using pthreads for parallelization in OpenBLAS rather than OpenMP. The increase in performance achieved due to the update of DGEMM is about 20% [14].

Another notable thing about HPC is the sparse matrix vector multiplication, SpMV. It is actual, for example, in the numerical solution of partial differential equations [30]. In quantum chemistry, it is used, for example, for calculations by the configuration interaction method [31]. In [32] a study was done on the Kunpeng 920 of the performance achieved when using NEON SIMD instructions with an analysis of the dependence on the data storage format in the matrix. It should also be noted that for the performance of SpMV, it was found important to explicitly take into account the presence of the NUMA architecture [30].

In [33] , a detailed comparison of the performance of Kunpeng 920-6426 and 18-core Xeon Gold 6140 was made. In cases where performance was limited by working with cache or memory, Kunpeng turned out to be ahead. But in more computationally intensive benchmarks using vectorization, the Xeon outperformed. *A study at the Research Computing Center of Moscow State University within the framework of a joint project with Huawei*[URL] covers the comparison of Kunpeng 920-6426 not only with Xeon 6140, but also with EPYC 7763 (Milan, with Zen 3). The Kunpeng 920 outperformed the Xeon 6140 in computationally intensive not vectorizable benchmarks and was slower in sparse matrices. And EPYC 7763 usually outperformed both Xeon 6140 and Kunpeng 920. At the same time, some features of EPYC 7763 are close to Kunpeng 920: 64 cores with comparable frequencies, both support only 128-bit vectors, 8 memory channels each

(although EPYC is supported by DDR4-3200 vs DDR4-2933 in Kunpeng). However, this data is somewhat preliminary.

In general, it can be considered clear that Kunpeng 920 is slow in performance compared to leading models of modern server processors Intel Xeon Scalable 2nd or 3rd generation, and AMD EPYC with Zen 2 or 3 architecture. Newer models of Kunpeng server processors have also appeared, including the Kunpeng 920 7265, in which the cores operate at a higher frequency of 3.0 Ghz (the list of models *is available in cite*). The Kunpeng 920 performance data is not analyzed in more detail here, among other reasons, due to the small number of relevant publications. And in 2021, the Kunpeng 930 (5 nm) with SVE support is expected to be released, and it is likely to become used more in supercomputers. However, the implementation of this is now possibly frozen by sanctions against Huawei, including restrictions on the supply of modern chips from TSMC.

Another modern AArch64 processor that might be of interest to HPC is the *80-core Ampere Altra Q80* [**1**] (see Table 1 above) and the newer *128-core Altra Max*. They have more cores than the 64-core Kunpeng and higher peak performance. Ampere states that their 80-core processor has a performance comparable to AMD Roma EPYC 7742 and 2 times faster than Intel Cascade Lake SP Xeon Platinum 8280 [**1**] (see Table 4 for general information about processors of this type). Thus, in a *molecular dynamics performance test using NAMD*, the Altra Q80/3.3 GHz processor turned out to be faster than 1-–2 processor servers with Xeon 8280, and a single processor server with EPYC 7742, but lost to a two-processor server with EPYC 7742. However, this test does not use AVX-512, which can reduce Xeon performance by, say, 2 times. And information about parallelization was not given at all. *The Ampere Altra Review* also shows the data obtained for SPEC benchmarks (including SPECrate2017_int and _fp). Data on other more reliable benchmarks topical to HPC is not available for these processors. The manufacturer is generally focused not on the traditional HPC market, but rather on AI and other areas of application, and Altra is not considered in this review.

The 64-core Amazon AWS Graviton2 *is inferior to Ampere Altra in all parameters (important for performance)* given in this review (see Table 1), as in the above SPECcpu2017 benchmarks, and is also focused not on traditional HPC, but on the use of cloud computing technologies, and is not considered in the review.

Considering all that has been said above about AArch64 processors supporting NEON vectorization, this review does not analyze their performance in more detail.

1.2. Processors with SVE support: Fujitsu A64FX and computing systems based on it

Other than the A64FX, processors that support SVE are yet to be released in the near future. These are Huawei Kunpeng 930, which was also planned to support multithreading (SMT), and the European SiPearl Rhea [1]. It assumes support for SVE-256 bits (not only with DP, but also with bfloat16), but also additional DDR5 memory to HBM2E. Perhaps this reflects the desire of developers for slightly more mainstream applications for HPC, since the relevance of today's hardware support for GEMM is disputed [4].

1.2.1. *A64FX important for performance hardware features*

The ARM processors discussed above at the time of their release were characterized by increased memory bandwidth (compared to Xeon processors), which became more relevant for HPC during the transition to work with multi-core processors, since the bandwidth increased not so fast [20]. As noted above, this is very important, for example, for CFD calculations.

The A64FX became famous in HPC not only due to the record peak performance for today, but also due to the stable leadership in the TOP500 of the Japanese supercomputer Fugaku, built on these processors without the use of a GPU at all [13]. It differs from the most modern x86-64 processors not only in the use of radically faster HBM2 memory (see Table 4), but also in the construction of the entire memory hierarchy (although EPYC Zen 2 and 3 have analogies in cache construction) and an advanced SVE system of work with vectors. For energy management, the A64FX also has more sophisticated tools than Xeon, allowing you to affect energy efficiency. To optimize performance and energy efficiency, all this must be taken into account in compilers, libraries for HPC, and sometimes even in applications themselves.

The most important features for A64FX performance are shown above (Table 4), where they are compared with various models of modern Intel and AMD server processors, and Figure 2 based on Figure 2 from [35]

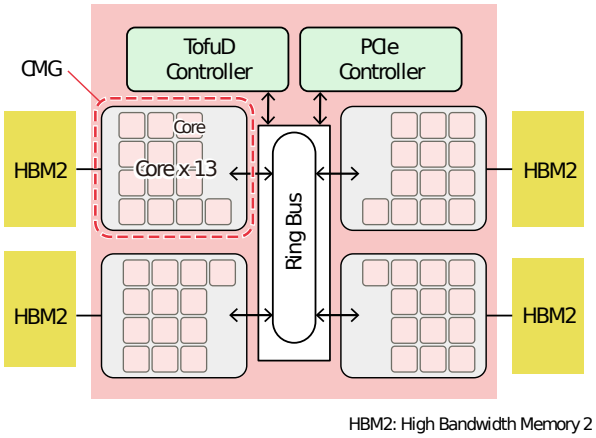


FIGURE 2. Block diagram of A64FX CPU

shows the general design of the A64FX, illustrating the memory design of the A64FX [34, 35]. This processor consists of 4 groups of cores and memory (CMG, core cluster) — so that each CMG has 12 cores (plus one auxiliary), an HBM2 memory controller and shares the L2 cache (and each core has its own L1 cache). All CMGs in the processor are connected by a single ring bus, so that the L2 cache and the entire HBM2 are common to the entire A64FX, and the processor is a ccNUMA system [35].

This memory hierarchy is different from all modern x86-64 server processors, in which the L3 cache is shared, and the L1 and L2 caches belong to the cores (see above, Table 4). In the A64FX, the size of the L1 cache area in the chip is more than 10% of each core, and adding another cache level in the core would significantly increase the die size, which was found to be economically unreasonable [34]. At the same time, the HBM2 used in the A64FX has a dramatically higher bandwidth than DDR4 in x86-64 processors — 256 GB/s for each CMG (see Table 4 above). Each CMG contains, in addition to 12 computational cores, another auxiliary one — for servicing the operating system. However, Fujitsu PRIMEHPC FX700 supercomputer servers *use A64FX models without such auxiliary cores*.

There is no SMT support in A64FX. This may be of an economic nature (considering that, because of this, the A64FX abandoned both the additional cache level on the cores and the addition of DDR4 memory to HBM2 [34]). Perhaps, it was done because SMT is far from is always useful for HPC (see, for example, [10]), and the implementation of such a complex

thing requires not only additional area, but also energy. Performance increases due to the use of SMT are not always observed in HPCs, since the performance of most applications is limited more often either by different threads using the same execution units, or by memory bandwidth [10]. In [10], there are cases where disabling SMT in the AArch64 processor gives a performance increase, although there are also applications where SMT is important.

SVE (Scalable Vector Extension) is the most important addition of A64FX relative to ARM v8.2-A. The A64FX now supports 128, 256 and 512 bit vectors. In modern processors with vector operations, the length of vectors is indicated in bits, including since ISA assumes vector operations on different data types, not only DP and single precision (SP), but also half precision, including bfloat16. SVE assumes the ability to work with vectors of different lengths from 128 to 2048 bits with a step of 128 [36], and the program for working with hardware support for vectors of other lengths does not need to be retranslated (the length is already determined at runtime). This is one of the advantages of SVE over AVX-512. Another advantage of SVE is the use of predicate registers there (for example, for fully vector processing of operations on arrays of length not a multiple of the hardware length of the vector) [36]. One bit of the predicate register corresponds to one byte of the SVE register, and this allows you to work with fragments of vectors.

Another important plus is the support for vector gather/scatter operations, illustrated by the following two operators

$$\texttt{gather} : \texttt{a}[\texttt{i}] = \texttt{b}[\texttt{index}[\texttt{i}]],$$
$$\texttt{scatter} : \texttt{a}[\texttt{index}[\texttt{i}]] = \texttt{b}[\texttt{i}].$$

As a result, SVE contributes to improved autovectorization by compilers.

Due to the fact that SVE in A64FX can work with 512-bit vectors, you can get 32 results (DP) per core per clock, as in Xeon, starting with Skylake. However, the A64FX has many more cores, giving it higher peak performance. SVE provides not only an increase in performance, but also a reduction in power consumption [37].

Another unique feature of the A64FX, especially for supercomputers, is its integration of highly scalable and fault resilient TofuD RDMA interconnect management. This network has the 6-dimensional grid/torus topology previously used in the K supercomputer, Fugaku's predecessor. But the bandwidth of TofuD, due to its integration into the processor,

now depends on its frequency. For the accelerated (boost, see below) mode with a frequency of 2.2 GHz, it is 6.8 GB/s for each of the 6 communication channels (Tofu Network Interfaces, TNI) [10, 34, 35]. This is less than in Infiniband HDR 4x and 16x, but for large computing systems with a complex communication topology between nodes, the total node (injection) bandwidth of all simultaneously operating channels (TNIs) is important — 40.8 GB/s, which is higher than the bandwidth of a single HDR 4x card, but lower than HDR 12x.

But the reason for using in small clusters (for example, with FX700) not TofuD, but Infiniband EDR is due to the continuous allocation of nodes in TofuD, which can reduce the availability of such clusters due to fragmentation (Yuichiro Ajima, Global Fujitsu Distinguished Engineer, private communication). The latency of RDMA communication (Put 8 bytes) in TofuD is 0.49 (for near CMGs) and 0.54 µs(for far CMGs) [38], while in *Infiniband HDR (Nvidia/Mellanox ConnectX-6)* it is 0.6 µs.

TofuD shares a controller with PCIe-v3 in the A64FX, so the entire A64FX can be classified as a system-on-a-chip (SoC).

The A64FX has special power control. There are 2 basic frequency selection modes — normal and boost; in the FX1000, this corresponds to frequencies of 2.0 and 2.2 GHz. In each of these modes, it is possible to enable another eco-mode, in which only one of the two floating-point ALU pipelines remains functioning [34, 35]. Accordingly, 4 power management modes are obtained, which can be selected to achieve maximum performance or select energy efficiency (this will be discussed below). GPUs are considered to be more energy efficient than CPUs. However, in [35] the energy efficiency of the A64FX is noted as comparable to GPU-based computing systems.

We do not consider the available data on the A64FX microarchitecture in the review; more detailed information about OoO, branch prediction mechanism, TLB construction and other things can be found in *the manual*. But the A64FX has another key feature for HPC. It was developed specifically for HPC tasks [34], and first of all for the creation of the Fugaku supercomputer [35]. In [34], the use of co-design for the A64FX is indicated, when the performance-determining parameters of the future A64FX were selected together with software developers for HPC, and taking into account specific applications. For this, a processor emulator was used. For example, to select the optimal number of cores in CMG, HPL performance was tested. The well-known Japanese complex of quantum

chemical programs NTChem was also one of the "participating in the selection" applications. As a result, [34] clearly demonstrates why Fujitsu chose these parameters. Undoubtedly, this approach is very attractive for HPC-oriented processors.

1.2.2. *Potential deficiences of A64FX for HPC performance*

The A64FX also has a number of weaknesses, usually considered in relation to x86-64 server processors (see above, Table 4), which are often caused by the desire not to increase the cost (for example, not as large a fixed amount of HBM2 memory, to which DDR4 is not added) [34]. There is also data from 2017 that in the NSF supercomputer resources for HPC, 86% of tasks did not require more than 32 GB of memory in the nodes [39]. And the latencies of HBM2 are greater than those of DDR4 [35].

It should be noted that the memory latency values measured in the benchmarks include several components, and the latency in the data transfer bus may not be the main contributor to the total latency. Although the significantly large HBM latencies relative to DRAM have been known for a long time, and this is indicated in a number of publications, there are not so many quantitative estimates, and the resulting latencies depend on the chips working with the memory. As an example, we indicate the latencies obtained in the Shuhai benchmark when hitting the page (that is, when it is already open) when working with the FPGA chip on the Xilinx Alveo U280 board — 107 ns for HBM and 73 ns for DDR4 [40].

Another weakness is the absence of L3 cache, the reason for which is explained above.

The A64FX's out-of-order (OoO) design also used simulation of A64FX and using small parts of applications, but deciding on the right amount of resources for OoO at that time was hampered by the lack of efficiency of the new compiler. At the same time, limiting resources for OoO contributes to lower power consumption, and progress in the work of the compiler can generally make weaker problems with OoO [34].

In A64FX, the capacity of the reorder buffer (ROB) is 128 entries, which is significantly less than in modern x86-64 processors: there are 224 of them in Xeon Skylake, 352 in Ice Lake, and 512 are planned in the expected Alder Lake; in EPYC Zen 2 — 224, in Zen 3 — 256. The number of physical registers (except for predicate registers missing in Xeon) in A64FX is also less than in Xeon Skylake [41]. A large latency in instruction execution times can lead to a weakness of OoO [34]. And

in a number of works [42–45] , large latencies in the execution of various A64FX commands were noted. So performance limitations of the A64FX due to OoO weaknesses are quite likely.

The PCIe used in the A64FX is not the latest version used (see Table 4 above); SSDs can work with PCIe-v4, and the expected Power10 will already have PCIe-v5 [16]. At the same time, SSDs are also used for HPC [46]. But for the A64FX, the integration of the controller with TofuD in the processor is more important, and the problem of a high-speed connection to the GPU did not arise, since they are not supposed to be used in the supercomputer on the A64FX. But the peak performance of the Nvidia V100 and A100 GPUs is several times higher than that of the A64FX: 7.8 and 9.7 TFLOPS, respectively [47], and the new AMD Instinct MI100 has even more (11.5 TFLOPS) against 3.4 TFLOPS in A64FX, and the use of the GPU is now relevant, at least for HPC applications that already work effectively with the GPU. Due to PCIe-v3 limitations in the A64FX, the new Japanese supercomputer Wisteria/BDEC-01 is planned to use a heterogeneous fabric with some nodes on the A64FX and others on the Xeon Ice Lake with A100.

Finally, keep in mind that there are also HPC benchmarks/applications where using SMT (not supported on the A64FX as noted above) can improve performance.

The following section 3 of this review also shows the impact of the A64FX indicators noted in 1.2.2 on performance.

1.2.3. *About computing systems based on A64FX*

The first thing that is unusual for HPC clusters that have been using dual-processor servers for a long time is that servers with A64FX are single-processor, and A64FX performance is often compared with two x86-64 processors. Moreover, two Xeon processors often have a number of cores close to A64FX. Fujitsu with A64FX offers two PRIMEHPC supercomputer platforms — FX700 and FX1000. The latter is used in the Fugaku supercomputer, and has the maximum facilities — up to 384 nodes per water-cooled rack, connected via TofuD. A64FX nodes have a frequency of 2.2 GHz and 48 cores plus 4 assistant ones (see Figure 2). FX700 is based on 2U devices; up to 8 nodes are placed on the chassis, and they work with air cooling. A64FX in them operates at frequencies of 1.8 or 2 GHz, and there are no assistant cores. The nodes are connected via *Infiniband EDR*.

You should pay attention to the difference between A64FX in FX700 and FX1000. The A64FX from the FX700 does not support TofuD, clusters work with Infiniband EDR or HDR100 — this uses a PCIe-v3 x16 slot. But then there is also PCIe-v3 x4, to which the SSD is connected. Other computing systems with A64FX offered by HPE/Cray are Apollo 80 (previously announced as Cray CS500). The nodes here are connected via Infiniband EDR or HDR100. A *42U rack* can contain up to 168 servers with 48-core A64FX clocked at 1.8 or 2 GHz — similar to those used in the FX700.

In general, you should pay attention to the fact that not only the clock frequencies differ in different A64FX models, but there may or may not be attachment processors, and TofuD may not be supported and will work with different bandwidth depending on the processor frequency. And all of the just listed computing systems with A64FX are not traditional servers, but are initially oriented towards working in clusters for HPC.

2. A64FX: Operating systems and software development tools for HPC

To work with the A64FX, you can use the classic Linux distributions for HPC — RHEL 8/CentOS, and SLES 15/OpenSuSE. Here you need to pay attention to the need for their real installation on all SSDs of all computing nodes of the cluster — to store the root file system there. The option of storing in these nodes only root filesystem images resident in memory on the Ookami supercomputer (on special nodes for compiling and debugging SSDs were originally there) required about 14 GB of memory for this in each node — due to work with pages with a capacity of 64 KB, and so much "takes away" even a super-small file [**39**]. And since Fugaku can already be attributed to the EFLOPS-level supercomputer (when working with half precision), to implement HPC with such an ultra-high level of parallelization, a complicated memory hierarchy and other complexities, in addition to Linux (RHEL), the McKernel microkernel [**48**] is used, which fulfills massive system calls for HPC, while more complex ones are forwarded to Linux.

Five C/C++/Fortran compilers capable of creating code with SVE and a number of math libraries are available for the A64FX. There is a statement that the performance of most HPC applications today is limited by memory bandwidth [**49**], which gives the A64FX a clear advantage among available processors. The A64FX, however, has a different performance-to-memory

bandwidth ratio, and real HPC application data and benchmarks show that the A64FX isn't always a performance leader, and the quality of software development tools is critical. In this review, compilers and libraries are analyzed mainly in terms of the performance received by the consumer — and not to analyze the possible causes of their performance flaws and ways to eliminate these. Also relevant for choosing a compiler, the availability of modern versions of C/C++/Fortran, including OpenMP tools, is not taken into account here.

There are already quite a few publications demonstrating the level of efficiency of compilers. But since A64FX uses a number of original microarchitectural solutions, and became available quite recently, compilers are considered as not yet "mature" enough [50], and although all modern versions of them can already implement autovectorization in SVE, in [39,51] it is made essentially the same conclusion. We list the available compiler versions used in a number of publications: Fujitsu 4.5.0 (there are two of its modes — traditional (FJtrad) and based on LLVM (including FJclang), where the back-end is based on LLVM 7); LLVM v.12 (there you can use vectorization in polly loop optimization tools); GNU v.10.2.0, as well as ARM and HPE/Cray compilers — not available on Fugaku due to lack of licenses (at least at the time of sending the publications used in the review). As "immature" compilers for A64FX, we will refer, for example, to data on GNU compiler errors, flang errors in LLVM [50], run-time errors in ARM and Cray compilers [39,52].

Used in [50,53,54] LLVM 12 is a modern stable version, version 13.0.0 appeared only in October 2021. The continuous rapid development of other compilers for the A64FX also clearly manifested itself in 2021. For example, new versions of the ARM compiler (including 21.1) and Cray (12.0) were released.

The Cray and Fujitsu compilers are commercial, as are the ARM compilers included in Allinea Studio's HPC-focused suite, including C/C++/Fortran compilers and the *Arm Performance Library, armPL* (although a free trial version is also available), and the freely available compilers are GNU and LLVM. The main options for all these compilers to enable vectorization, OpenMP and other optimizations are given in [55].

ARM compilers for HPC use cland/flang as front-end, and back-end is based on LLVM (in 21.1 — on Clang 11), where 2 vectorization units are used: one natural for loops, and the other gives parallelism at the superword level (SLP), when vector instructions are generated instead

of a set of similar independent instructions [55]. It should also be noted that the C extension implemented for the first time in the ARM compiler, ACLE (Arm C Language Extensions), which was also made for SVE, and helped autovectorization in the initial period of work with SVE, is now outdated and is planned to be removed in future versions of the compiler.

Fujitsu's compilers have the advantage of supporting some performance-enhancing A64FX hardware features that weren't mentioned above — since they are more sophisticated, they currently could only be used in these compilers, and their impact has been little studied. However, [43] explored the impact of FJtrad 4.4.0a on the use of these facilities through options, environment variables, or pragma directives. This is zero fill (allows better memory handling, and [43] shows performance acceleration in the stream triad test, but other processors have similar mechanisms); the use of hardware barriers for accelerated synchronization of threads and the splitting of the cache into sectors of different sizes. The latter allows for better management of the cache space allocated to each data structure in the code and can help avoid "polluting" the entire cache. In [43], an increase due to this performance was shown when multiplying a dense matrix by a vector. And the use of hardware barriers accelerated the synchronization of threads in FJtrad (fcc), including in comparison with its software implementation in gcc 10.2.

One of the largest compiler studies [50] analyzed not only the choice of compiler options, but also the effectiveness of Fujitsu's recommended choice of the number of OpenMP threads and MPI ranks in hybrid parallelization on the A64FX.

Fujitsu and RIKEN (the Japanese Institute of Physics and Chemistry, where the Fugaku supercomputer is installed) recommend using 12 OpenMP threads on each CMG, and on top — parallelization with MPI rank 4 (according to the number of CMGs). In [50] , other options for choosing combinations of OpenMP/MPI were also used.

For microkernels from RIKEN-priority applications, FJtrad almost always gave the best performance; GNU generated faster code in a few cases, but bugs in some. In a group of mini-applications from RIKEN, for example, for the quantum chemical program NTChem, GNU lags behind other compilers. In the stream tests, LLVM and GNU gave a performance increase of up to 51% relative to FJtrad. In the SPECspeed 2017 benchmarks with multi-threaded floating point workloads and in SPEComp, GNU generally performed the worst. For C/C++ SPEC tests, LLVM-based

compilers (including FJclang) and in some cases, GNU-based compilers may provide a runtime advantage over FJtrad, but for Fortran, LLVM does not provide a significant performance benefit. In the *Polyhedral Benchmark (PolyBench)*, which include 30 single-threaded microkernels, LLVM with polly, according to [50] , very often dramatically outperformed other compilers, but sometimes it was significantly inferior. But we must keep in mind that PolyBench is used for *performance testing* in the development of polly tools. And in [50] these tools are generally classified as of little use anywhere outside the PolyBench benchmarks itself.

In [56], within the BP3 benchmark from CEED (Center for Efficient Exascale Discretizations), gcc 10.2.0 gave better performance than Fujitsu (fcc) 4.4.0a.

In [57] on A64FX/1.8 GHz, performance testing (bandwidth) of DAXPY both in a single-threaded version and on the entire processor with parallelization in OpenMP or pthreads showed that gcc 11 was ahead of ARMclang 21 in performance. As a weakness, gcc 11 only the absence of elementary function vectorization in glibc was pointed out (see below for more on this).

However, in [53] on the application of quantum chromodynamics QPACE 4, and on benchmarks stream and matrix multiplication in a cluster on an FX700 with A64FX/1.8 Ghz, gcc 10.1 and 10.2 was optimal in terms of performance, while gcc 11.1 and LLVM 12 lagged behind. It is also not clear to what extent these results were affected by the use of ACLE. In [54], on the same computer system, with the same benchmarks and application, a comparison with other compilers was added, including with ARMclang 21.0 and FJtrad, FJclang 4.3.1 — but all compilers (including gcc 11.1), except for gcc 10.1/10.2, gave a bad performance.

In [50] it was concluded that the compiler system for HPC with A64FX is not yet as developed as for x86, and it is impossible to point to one basically the best compiler. There may be different recommendations for different types of benchmarks/applications. It seems that the version of GNU 10.2 studied in [50, 53, 54] should still not be recommended for use due to a considerable number of errors; should now initially target the more up-to-date stable version of GNU, and 10.2 is not supported. Support for A64FX and improvements in working with SVE are declared in the currently supported version 10.3, and in the stable version 11.2 a number of bugs are fixed, partial support is already provided for OpenMP 5.0, and autovectorization when working with complex numbers — these GNU

versions seem to be recommended now for extensive work with A64FX. Version 12, which is under development, may be of interest mainly due to the development of OpenMP 5.0 and OpenACC implementations in it.

Other publications on A64FX performance sometimes analyze a wider number of compilers, other versions of the above compilers (and other options), including Cray 10.0.1/10.0.2 and ARM 20.3 (i.e. not only 21.0) [39, 57–59].

Another detailed comparison of compilers for A64FX is [52], where a wide range of compilers was compared in work on 3 different applications based on classical mechanics (including mini-applications, including minimod for seismic modeling implemented by the finite difference method [60]; minimod is used as a platform for comparing the performance of new compilers in HPC). And the SWIM (weather prediction) application used for performance tests in [52] is included in the modern version of the SPEC CPU, and is also used to evaluate the performance achieved on a supercomputer.

In this article, performance investigations were carried out on 2 different A64FX models: with a frequency of 2.2 GHz — on Fugaku and 1.8 GHz — on the Ookami supercomputer (there were no auxiliary cores in addition to 48 computing ones in the processors). In this data, the orientation for applications should be on the total execution time, and not on the acceleration achieved depending on the number of OpenMP threads. However, the reason for the reduced performance of the application on a certain number of A64FX cores may also be insufficient parallelization on OpenMP by the compiler. The number of OpenMP threads used was a multiple of 4, plus calculations with 1 and 2 threads; the total number of these threads was evenly distributed among the CMGs.

On Fugaku, the performance tests used versions of Fujitsu compilers older than 4.5.0, but there were no Fujitsu compilers on Ookami. On Ookami, Cray 10.0.1 gave all applications maximum performance with any number of threads up to 48 — except for minimod, where the Cray compiler gave an error. gcc 10.2.0 and 9.3 were sometimes ahead and sometimes behind ARM 20.3; the Cray compiler on Fugaku was unavailable.

FJtrad on Fugaku in the *PENNANT* mini-application (which uses some algorithms from the well-known FLAG fluid dynamics application of the Los Alamos National Laboratory in the USA, which gives patterns of typical FLAG memory access) generated less performant code than FJllvm, and that was ahead of gcc 10.3 with less than 12 OpenMP threads

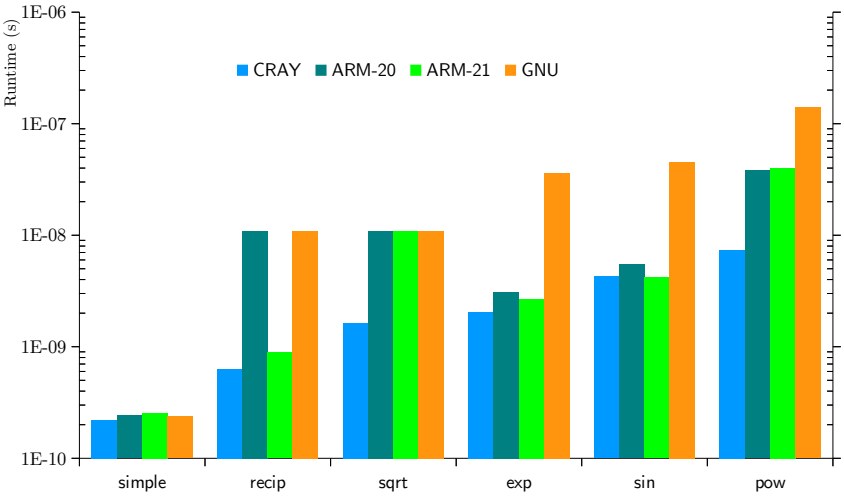


FIGURE 3. Vectorization of elementary operations and functions

used — here the poor performance scaling of version 4.3.0 Fujitsu compilers with an increase in the number of threads was clearly expressed. SWIM translated by Fujitsu 4.4.0a was always faster than GNU 10.2. And on minimod, FJtrad turned out to be the best, then FJllvm, and the slowest — gcc 8.3.1.

This data, unfortunately, provides less comparative information due to different compiler versions used. Probably, Cray 10.0.1 provided better performance, and as a conclusion, [52] points to less efficient OpenMP parallelization in Fujitsu compilers. Unfortunately, the applications tested in [52] did not use the OpenMP SIMD directives.

In [39] on the same Ookami, the same compilers are used for a much wider range of benchmarks. The authors of [39] referred to insufficient experience with the Fujitsu compiler, and believe that for C/C++ the choice of the optimal compiler for A64FX is difficult, including due to possible limitations in supporting new C++ standards, and for Fortran, the Cray compiler was considered the best choice. In [57, 59] the vectorization of elementary functions (square root, exponent, sine, etc.) was investigated, and here Cray 10.0.1 was also faster than GNU 10.2 and ARM 20 and 21 (see Figure 3 from [57, p. 25]). Here "pow" means exponentiation, "recip" is the calculation of the reciprocal, and "simple" probably refers to the

primitive loop from [59] containing the operator

$$y[i] = 2 * x[i] + 3 * x[i] * x[i].$$

The vectorization of such functions allows us to vectorize using cycles which were not previously vectorized at all. This can be used, for example, to calculate basis functions in quantum chemistry. Modern Fujitsu, Cray, and ARM compilers use this vectorization, and GNU 11.1.0 did not vectorize sine, exponent, and exponentiation. To vectorize elementary functions, some compilers can use the Sleef library (see below) [59]. And the best execution times are due to the vectorization of such functions in [59]. And the best execution times due to the vectorization of such functions in [59] were given by the Fujitsu compiler, significantly ahead of Cray.

The GNU 11.1.0 used in [59] is close to the last stable version of GNU 11.2 mentioned above, and this paper assumes that this situation with non-vectorization of elementary functions in the GNU glibc library will continue for a significant time.

In [59] on Ookami, the Cray compiler gave slightly lower performance than the Fujitsu compiler in a number of simple, typical SVE loops (more significantly, in vectorized elementary functions), and almost always in the *NAS Parallel Benchmarks (NPB)* tests for single-core execution. And in NPB, when parallelizing on all 48 A64FX cores, in some NPB benchmarks the Cray compiler was faster than Fujitsu, in others it was inferior, but more often, GNU 11.1.0 became the leader here. LLVM 11 is characterized in [59] as having limited support for SVE, and the weakest point of GNU 11.1 is the absence of vectorization of a number of elementary functions. The Cray compiler often yielded up GNU in the NPB tests on 48 cores, but outperformed GNU in the exascale-oriented explosive fluid dynamics proxy application *LULESH 1.0*. As a conclusion, [59] indicates the highest performance achieved with tools from Fujitsu and HPE/Cray.

In [43], in performance tests, including SpMV, conducted on the FX1000 with A64FX/2.2 GHz, fcc 4.4.0a and gcc 10.2.0 are compared — but there is no clear conclusion about the performance advantage given by fcc or gcc. In [41], for genomics tasks, the FJclang compiler gave better performance than gcc 11 and ARM HPC — but the exact versions of the compilers are not indicated.

Based on the above data on the performance achieved during compilation, we can say that now the only one claiming to be the leader more often, giving the maximum compiler performance for the A64FX, could still be HPE/Cray, although there is absolutely not enough data for this. Another commercial compiler from Fujitsu is not yet ready for this role. It wins mainly when the features of A64FX implemented in it are extremely important, and HPC-oriented LLVM cannot yet be considered advantageous relative to GNU. Yes, we can talk about the lack of maturity of A64FX compilers, but such a clear leadership as Intel's compilers for x86-64 may not develop for A64FX compilers at all. And if in performance tests using these compilers, A64FX is ahead of x86-64 server processors, then it can be argued that this will continue with future compilers.

For ARM processors, a large number of *mathematical libraries* are available for HPC, and for A64FX (including a lot of them for linear algebra problems) — including Fujitsu SSL2 (Scientific Subroutine Library II), HPE/Cray LibSci, Arm Performance Libraries (ArmPL). According to [**59**], by mid-2021, SVE was already used in ArmPL, Cray LibSci, Fujitsu BLAS and FFTW. Libraries freely available in source code (such as OpenBLAS) can be translated to create codes containing SVE. For a complete list of available compilers and libraries, as well as applications on Fugaku, see [**62**].

In [**39**], ArmPL, LibSci, and OpenBLAS are compared on several Ookami benchmarks. On DGEMM and HPL in single-threaded versions, ArmPL turned out to be the best here, LibSci lagged behind (in DGEMM, 57.6% and 34.6% of peak performance were obtained, respectively), and OpenBLAS even stronger. FFTW in the Fujitsu implementation gave 0.93%; LibSci — 0.79%. ArmPL (20.3) did not give SVE codes and only 0.01% of peak performance was obtained in FFTW. In the DGEMM calculation on the whole node, LibSci gave 88% of peak performance, and Fujitsu declared 94% reachability [**39**]. In [**63**] in the Sandia Laboratories, known in the world of supercomputers, it was shown that on the same Fugaku node for DGEMM and ZGEMM, the Fujitsu SSL2 library gives better performance than ArmPL, which corresponds to the data [**59**], where in DGEMM ArmPL, LibSci and OpenBLAS lagged behind in performance on single A64FX core from Fujitsu BLAS.

In [**45**], the performance of DGEMM was studied when working with small matrices that fit in the cache L1 when working with different linear

algebra libraries. Here, tools were used, including those of own design — loopy, and the PSpaMM library for sparse matrix multiplications (it can also work with dense matrices), which generate optimized intermediate code, and then they can be translated, for example, by gcc (used gcc 11). But of all the libraries analyzed, SVE was used only in loopy, and the achieved single-core performance on A64FX/1.8 GHz was only 32% of the peak value. Thus [45] demonstrates the high importance of the software development tools used.

For comparison, here it can be pointed to [57], where for the product of small matrices (non-square, the product of the matrix by the transposed) with the same A64FX/1.8 GHz processors using the MADNESS library with optimization for small non-square matrices gave 92% of the peak performance in the same single-threaded execution. SciLab 10.0.1 gave less than MADNESS performance, often close to ArmPL 20.3, although sometimes significantly ahead of ArmPL.

It is possible that the simple use of SSL2 in [45] would have given significantly higher performance. SSL2 also includes programs for sparse matrix vector multiplication (SpMV) with different sparse matrix storage formats. However, it was found in [43] that careful optimization of SpMV with the choice of storage format can significantly outperform the performance in SSL2 for A64FX.

Other linear algebra libraries have also been ported to the A64FX. For example, in [56], the A64FX used the MAGMA library, known for its focus on working with GPUs, and in [64] — BLIS [65]. The latest library, which features an extended set of functions for operations with dense matrices compared to BLAS, is available in source code and builds modules that are optimal for a certain set of architectures, including A64FX. According to *the documentation*, BLIS usually outperforms SSL2, ArmPL, and Eigen in single-thread and multi-thread versions, although sometimes is slower to SSL2.

The EigenExa library (for solving an eigenvalue problem by reducing a matrix to a tridiagonal one by the Householder method, and then applying the Divide-and-Conquer (DC) method [66] previously worked on a K-computer, and is now ported to Fugaku with the modernization of the DC-part, based on DC from ELPA, and its scaling with MPI-parallelization was tested up to 32768 nodes [66]. Among other things, the SLEEF library can be noted, where vectorization is used to calculate elementary functions; SVE has been supported there since 2018 [67].

But in general, the influence of libraries on the performance obtained in the test should not be overestimated. So in HPL, where work with DGEMM limits performance, the use of LLVM gave 5% more performance than codes translated by Fujitsu compilers, although SSL2 was used in all these calculations [50].

Parallelization facilities are other important ones for working with the A64FX. It is understood that different implementations of OpenMP are available for A64FX simply due to the availability of a number of different compilers. An analysis of the levels of parallelization achieved in OpenMP was carried out in [39,52]. As noted above, for the development of applications, it is more natural to focus on the achieved performance, and not on the level of OpenMP parallelization itself.

On the A64FX, you can work with a whole set of different MPI implementations, including Fujitsu MPI (based on OpenMPI), RIKEN-MPICH, HPE/Cray MPI (based on MPICH), and of course MVAPICH2 and OpenMPI. As rightly noted in [68], good performance testing should include a combination of a compiler and an MPI implementation. In [68] at Ookami on the world-famous FLASH supercomputer application (using hydrodynamics for astrophysics), Cray 10.0.1 with SVE support turned out to be the best in terms of execution time; gcc 10 was inferior to it, Fujitsu was still behind, and ARM 21 was already several times behind. The Cray and gcc variants worked with the corresponding compilers translated MVAPICH 2.3.5 and OpenMPI 4.0.5; the latter was slightly slower than MVAPICH. In [59], it is assumed that there are weaknesses of Fujitsu MPI when working not with TofuD, but with Infiniband HDR, which resulted in poorer scaling (with the number of nodes) in HPL and FFTW on their Fujitsu implementation compared to ArmPL.

In [39], various combinations of compilers and MPI implementations (MVAPICH 2.3.4 and OpenMPI 4.0.5) were successfully tested on Ookami with an Infiniband HDR200 interconnect. In OSU MPI tests between two nodes (point-to-point) using OpenMPI, the bidirectional throughput achieved was 19.4 GB/s (about 78% of the peak), unidirectional — 12.3 GB/s (transmitted at 8 MB); PUT/GET delays (transfers of 8 bytes) were 3.9/5.2 s. In [56], MPI point-to-point communication on the Summit supercomputer (second place in the TOP500 after Fugaku) with different message sizes gave lower latency than on Fugaku.

Since the A64FX was originally planned for use in supercomputers, the popular PGAS parallelization tool, OpenSHMEM, can already be

used here. The PUT/GET latencies obtained in [39] were 4.5/4.1 μs here. Unfortunately, there is no data on the possibility of using the *new Nvidia/Mellanox HPC-X tools* on the A64FX when working with Infiniband, which, in addition to supporting MPI and OpenSHMEM, also has other original developments. In [69], it was when working with HPC-X that the highest performance was obtained in a number of tests on HDR200. Here, the two-socket servers used PCIe-v3 x16 just like the A64FX, but with one shared HDR card per server. GET latency achieved in OSU benchmarks for all used MPI implementations (including for MVAPICH2-2.3.1), and on MVAPICH2 — unidirectional GET/PUT bandwidthes, as well as bidirectional PUT bandwidthes when working with servers on different Xeon processors Haswell and Cascade Lake were significantly better here than when running on servers with a single A64FX in [39].

For difficult HPC tasks that can run on different computing systems and use parallelization, there is also the problem of porting from one hardware environment to another. To solve this problem, Kokkos tools using C++ are used, which create their own high level of abstraction for the hierarchy of computations and memory [70]. Kokkos is planned as a possible use in an EFLOPS supercomputer; naturally, these tools also work with the A64FX. Starting with Kokkos version 3.3, the Fujitsu compiler can be used there; SVE support has been added to the Kokkos SIMD library. However, in [63] it was found that although work with SVE is faster than with NEON, Kokkos works faster on Xeon Skylake than on A64FX.

3. A64FX performance for HPC tasks

3.1. A64FX performance at the level of a large number of supercomputer nodes

At the beginning — a brief information about the performance at the level of the entire Fugaku supercomputer, which allows us to demonstrate the achievable level of performance scaling, and the possibility of building or using a supercomputer without accelerators. The following data is taken from the respective sites for TOP500/GREEN500 and GRAPH500 (June 2021 versions). Fugaku's original performance indicators in the TOP500 [13] have been slightly increased in 2020, which is obviously

a consequence of progress in the software. Fugaku has 158976 nodes, of which 152064 were used in calculations for the TOP500 benchmarks. With a peak performance of 537 PFLOPS (A64FX in Fugaku operating at a frequency of 2.2 GHz), HPL received 442 PFLOPS (this is 86% of the peak performance of all nodes participating in the calculations). The HPL-AI benchmark, which uses mixed precision (including half precision), achieved a level of 2.0 EFLOPS. Coming in second place in both of these benchmarks, the Summit supercomputer (with Power9 and V100 in nodes) has 149 PFLOPS in HPL and 1.15 EFLOPS in HPL-AI (the last result was obtained only in 2021). In HPCG, Fugaku outperforms the second-placed Summit much more — 16 versus 3 PFLOPS, which is 2.8% and 1.3% of peak performance, respectively [**13**].

Previously, Fugaku running on A64FX/2.0 GHz in the HPL-AI test achieved a performance of 1.4 EFLOPS, which was the first achievement of the EFLOPS level, albeit with reduced accuracy [**71**, **72**]. And it's still higher than that received at Summit.

In terms of energy efficiency in the HPL Summit, it is slightly behind Fugaku: 14.72 versus Fugaku's 14.78 GFLOPS/W, but the TOP500 now has supercomputers with much better performance. The GREEN500 is now also headed by the Japanese supercomputer MN-3, which includes Japanese accelerators in servers with Xeon 8260M, but in the TOP500 it is in 336th place. From second to ninth places in the GREEN500 are supercomputers with servers based on AMD EPYC Rome or Milan of various models with NVIDIA A100 GPUs. Of these, we note 6th place in the GREEN500 Perlmutter supercomputer, which also ranks fifth in the TOP500. Fugaku, which led the GREEN500 at the end of 2019, is now in 20th place. This demonstrates not only AMD's recent success with the A100, but also the much more frequent x86-64 server processor improvements compared to the Fujitsu processor. But its possible upgrade may be distinguished by a stronger increase in productivity (see about its potential upgrade in the Conclusion).

GRAPH500 [**73**] demonstrates performance in processing big data, and implements the task of searching in a graph. RIKEN computing systems have been leading in GRAPH500 since 2014, and the earlier leader was a K-computer, and now Fugaku is the leader in BFS search [**73**] Similar to the GREEN500, the GRAPH500 now measures energy efficiency; Fugaku is not represented here, and in the top ten there are two computing systems

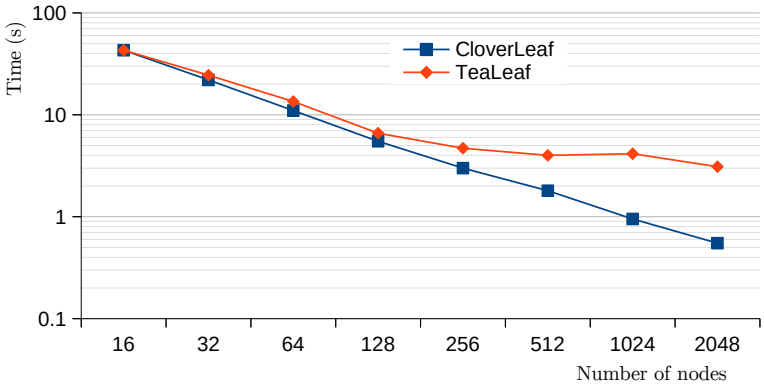


Figure 4. CloverLeaf and TeaLeaf performance

from Russia. For more details on working with GRAPH500 using the BFS core on Fugaku, see [74], and about optimizing power consumption in this case, see subsection 3.2 below.

We also present performance data on FUGAKU in a performance test that is topical for quantum chromodynamics (QCD) problems. QCD calculations were performed using mixed accuracy. On 147456 nodes, a performance of 102 PFLOPS (about 10% of the peak SP performance) was achieved with a power consumption of 20 MW [75, 76]. See section 3.3 for more information on these test calculations.

Let us also point out the initial data [56] on comparing the performance of the Summit and Fugaku supercomputers on the well-known CFD application *NEK5000*, which is part of the exascale-oriented *CEED software suite*. Summit gave the best results here, while Fugaku had I/O problems on large tasks. Performance scalability on Fugaku with A64FX/2.0 GHz up to 2048 nodes in other applications is illustrated in [38], see Figure 4 based on [38, p. 15]. Applications on A64FX and their performance data are provided below in overview subsection 3.3.

Here, one should also point out the NICAM atmospheric model calculations performed using 131072 Fugaku nodes (82% of all nodes), where a performance of 79 PFLOPS was achieved using single precision [77].

As another classical task, which is parallelized between Fugaku nodes connected by TofuD, one should also indicate the eigenvalue problem for dense matrices, which can be applied in various HPC areas, including CFD

and quantum chemistry. In quantum chemistry, it is often considered necessary to diagonalize matrices of large dimensions N, for example, for large molecules, although in some modern methods of quantum chemistry, including the most popular DFT, for large molecular systems, diagonalization requiring $O(N^3)$ was abandoned by developing approximate linearly scalable methods. Both stages used in testing for diagonalization using the EigenExa library — reduction to a tridiagonal matrix and its subsequent diagonalization by the DC method require significant computing resources. But the Householder method is poorly parallelized and requires a large memory bandwidth — and DC is well parallelized, but often gives unbalancing the load of different parallel processes [**66**]. For matrices with $N > 250000$, when parallelizing OpenMP + MPI, an acceptable performance scaling was obtained using up to 16384 nodes [**66**].

3.2. A64FX benchmark data on computing systems with no more than a few nodes

Further, we are only talking about performance from a single A64FX core to multiple compute nodes — a good opportunity for scaling at the level of the big Fugaku supercomputer is almost obvious. As starting values for evaluating these performance tests per node (they were often performed on Fugaku and Ookami), we indicate the estimates given in [**13**] for the stream triad — about 830-840 GB/s, and for DGEMM 2.4 TFLOPS (Ookami, Cray SciLab) and over 2.5 TFLOPS (on Fugaku).

The A64FX performance data below contains mostly published maximum values only; possible selection of optimal algorithms, compilers/options/libraries and so on is not considered here. The performance data here refers to a range from a single core to multiple servers in a cluster (and therefore multiple A64FX processors). When available, performance is compared to x86-64 processors: Xeon Skylake and Cascade Lake; AMD EPYC Zen 2 and 3, and when running on V100 and A100 GPUs. Performance comparisons with older processors or accelerators are not seen as much less interesting (and data on the MI100 is not yet available). Comparative data is sometimes given with the first well known in the supercomputer world ARM processor, ThunderX2, but the performance advantages of the A64FX relative to it, on average, are obvious.

A review of these A64FX benchmarks begins with DGEMM (and other dense matrix multiplications), as this largely determines the performance

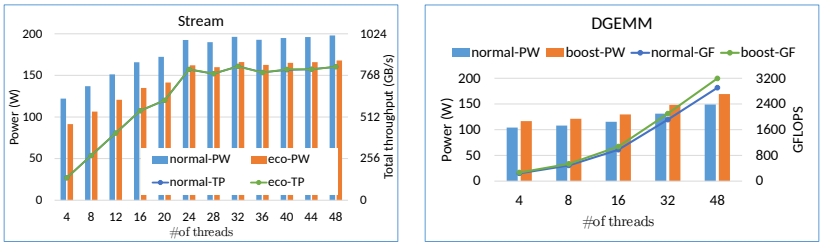


Figure 5. Power and performance in stream and DGEMM on A64FX

in the HPL benchmark used in the TOP500. But DGEMM is topical for a variety of applications. For example, in computational chemistry — in molecular dynamics or in various non-empirical methods of quantum chemistry, taking into account electron correlation. According to [34], on A64FX/1.8 GHz in DGEMM you can get more than 2.5 TFLOPS (90% of peak performance). At the same time, reducing the frequency from 2.0 to 1.6 GHz reduces performance by 20%, and power consumption by 18%; turning on the eco-mode reduces performance by 50%, and power consumption by 16% [34]. In [38], for A64FX/2.2 GHz, the performance is 3.2 TFLOPS at 200 W. Figure 5 from [38, p. 23] shows graphs of achieved performance (in TFLOPS) and power consumption (W) depending on the number of threads used in DGEMM [38]. The data obtained in [5] on the power consumption levels of A64FX components when working with DGEMM show that the main share of power consumption falls on processor cores, the L2 cache consumes many times less, and work with HBM2 takes very little energy.

Since the A64FX is designed to work with HPCs without connecting accelerators, it is useful to compare the performance of DGEMM on the A64FX and on the V100 and A100 GPUs. According to [4], on the V100, 7.2 TFLOPS is achieved, and on the A100, the performance is clearly higher; more information on A100 benchmarks is presented at [47].

The maximum DGEMM performance per A64FX core is achieved using SSL2, so the data for comparison with x86-64 processors is given here for A64FX with SSL2. The A64FX/1.8 GHz achieved 40.9 GFLOPS (71% of peak performance) — which is less than the Xeon Skylake 8160 core (43.4 GFLOPS and 97%), but higher than the EPYC 7742 (25.3 GFLOPS and 70%) [59], which only supports 256-bit vectors. In [39], a calculation

on the same node with A64FX gave lower performance, probably due to not using SSL2.

There is also data on the performance of the A64FX in other matrix products other than the traditional (for HPC) DGEMM. For example, [63] provides data on the performance of ZGEMM on A64FX from Fugaku. Special performance studies were also carried out on the A64FX for multiplications of small matrices, including those that fit into the cache L2 in the A64FX, using various linear algebra libraries [45]. In [45], results close enough to the peak performance of A64FX/1.8 GHz were not achieved — only 1.7 TFLOPS, but newer batch versions of BLAS/DGEMM [78], focused on working with small matrices, or a well-optimized SSL2 library not used here. In [57], A64FX/1.8 GHz performance data for small matrix multiplication by transpose using a specialized algorithm was obtained, which for a single core gave 53.2 GFLOPS (about 92% of peak performance), outperforming the Xeon Skylake/3.7 GHz core with MKL. And in [53], the performance of matrix multiplication, special for problems of quantum chromodynamics, where DGEMM is not used, was studied.

HPL benchmark data at the Fugaku level (including modern versions of the TOP500) is given above in section 3.1. For Ookami with A64FX/1.8 GHz, HPL performance per node was 1129 GFLOPS (40.8% of peak), less than in a two-socket server with Xeon 8160 (1156 GFLOPS and 53.7%, respectively), containing the same 48 cores as at A64FX. A dual-processor server with EPYC 7742 with 128 cores significantly outperformed both of them (1911 GFLOPS and 41.5%, respectively). As the number of nodes increased to 4–8, the performance advantages of clusters based on x86-64 remained [59]. And in terms of energy efficiency in HPL, at the end of 2019, supercomputers based on A64FX/2.0 GHz prototypes, with not very large dimensions, took first place in the TOP500, overtaking systems with accelerators [79]. In [39], performance in HPL on the same node with A64FX was lower, but SSL2 was not used.

In the above tests, A64FX often lagged behind modern x86-64 processors in performance, and now we should pay attention to benchmarks that give data on memory performance (its bandwidth), where A64FX should have clear advantages. *Stream*, which includes 4 performance tests in the form of loops for working with one-dimensional arrays and can be considered a practical standard for this — copying them ("copy"), multiplying by a scalar (scaling, "scale"), summing two arrays into a third

one ("add") and test "triad", the most commonly used in stream benchmark:

$$A[i] = B[i] + s * C[i]$$

where s is a scalar. This benchmark has a number of modernized variants, including BabelStream [80], which is primarily focused on working with the GPU (in this case, the transfer time, for example, over PCIe, is not taken into account) or on devices with a very large number of cores, in which including added another test for the scalar product of vectors. The set of microbenchmarks lmbench [81] also includes a stream with initial data from this set. In the event that the throughput achieved per stream for the A64FX and in alternative hardware is little dependent on the specific stream option, the refinement of that option in the text here can be omitted. But about the use of DAXPY, where the result in the above stream triad loop statement is written to the same array B, and not to another array A, the review clearly indicates.

The above starting estimates [13] for stream triad in A64FX on Fugaku are close to the BabelStream benchmark data for V100, 800-840 GB/s, and on A100 the bandwidth in BabelStream is much higher: 1.3-1.4 TB/s [47]. The bandwidth of such an A64FX processor (about 840 GB/s) in [43] was much higher than that of a dual-processor 96-core server with Cascade Lake (Xeon 9242), where 420 GB/s was obtained, and is close to the bandwidth of V100. In [43], the bandwidth of a stream triad was also studied depending on the number of cores used within one CMG.

In [79], about 80% of the peak bandwidth was received on the Fugaku node in the stream triad. In , for a server from Fugaku, the bandwidth in the stream triad was obtained close to the data in [43], and it was shown that turning on the eco-mode practically does not reduce bandwidth, but reduces power consumption by 18%.

In [41], bandwidth in the Fugaku node on lmbench/stream was compared with a two-socket server with a Xeon 8160 with a total number of processor cores as in A64FX, while also examining the impact of memory localization. A64FX outperformed the Skylake server by a factor of two or more. In [82], the stream copy bandwidth of A64FX/2.2 GHz and ThunderX2 is compared depending on the number of cores; A64FX results were many times greater. In [5], it was found that in the stream benchmark on the A64FX, the main power consumption was related to the processor cores; HBM2 consumed noticeably less, and cache L2 — many times less.

TABLE 5. Bandwidth comparison (BW[1]) in DAXPY: A64FX/1,8 GHz and Xeon Skylake SP[2]

Cores number	A64FX		Skylake	
	BW	bytes/FLOP	BW	bytes/FLOP
1 core	53,0	0,92	21,1	0,17
1 socket	840	0,30	145	$\sim 0,06$

[1]in GB/s; [2]Xeon Gold 6136/3 Ghz — 24 cores

In [39] stream triad gave 830 GB/s on Ookami for A64FX/1.8 GHz, while DAXPY managed to get 840 GB/s. It was several times faster than a two-processor server with Xeon 8160. On one A64FX core, DAXPY gave 53 GB/s — 2.5 more than on a server with Xeon 8160. In [53, 54] for A64FX/2.2 and 1.8 GHz received data for all four stream tests depending on the number of A64FX cores involved; the maximum achieved bandwidth was 835 GB/s, and in [43] the maximum for the Fugaku node was 841 GB/s.

In [57], the memory bandwidth achieved by A64FX in Ookami in DAXPY was studied in both single-threaded and multi-threaded versions (with parallelization in OpenMP), see Table 5. Already the single-thread result in A64FX is much higher than in the Xeon Skylake/3 GHz core. Found that about 6 threads per CMG can saturate bandwidth.

All in all, the A64FX memory bandwidth test data confirms the large hardware advantage of HBM2 over DDR4 memory in Xeon and ThunderX2 processors. But there is also a special, parallelized in OpenMP + MPI, test for memory bandwidth, where the result largely depends on the memory latency — *mega-sweep*[URL]. And here A64FX loses such total bandwidth to a two-processor (probably with Xeon 6126) server with DDR4 (SKL in Figure 6), which has 2 times fewer cores due to higher memory latencies of HBM2 [83]. But a dual-processor server with ThunderX2 (56 cores/2.6 GHz — TX2 in Figure 6), which has the same DDR4 memory, is significantly slower than the A64FX — since its vectors are 2 times shorter than in the A64FX, and vectorization is important in this test.

Matrix-vector multiplication is also important for various applications, such as CFD. For dense-matrix-vector multiplication, memory bandwidth can be the performance limiter, so A64FX can take advantage over x86-64. In [84], where this was important for the operation of the application, the performance of the BLAS program SGEMV was also analyzed, and it was

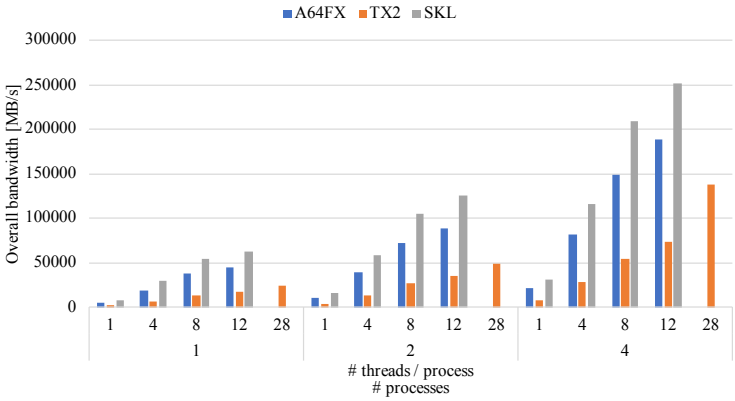
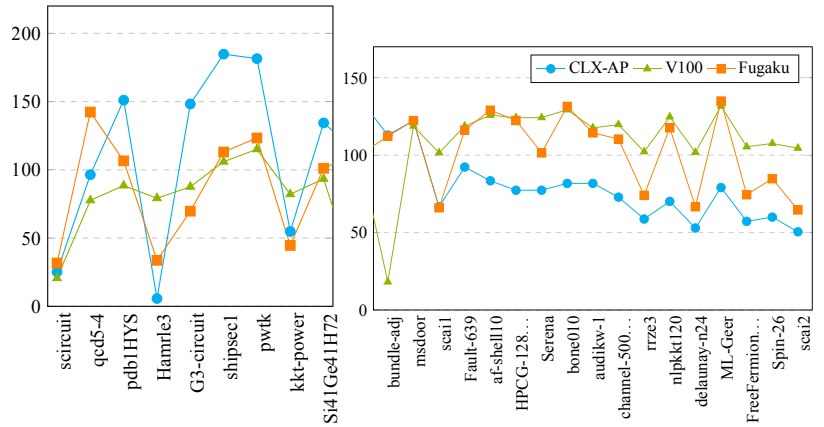


Figure 6. mega-sweep: Overall bandwidth (following Figure 7 in [83])

found that A64FX/2.2 GHz (from FX1000) is several times slower than the GPU A100 and the Japanese computer system NEC TSUBASA based on SX-Aurora processors supporting work with long vectors [85]. However, the A64FX outperformed the dual processor Xeon 6248/2.5 GHz servers (40 cores in total) and AMD EPYC Rome 7702/2.2 GHz servers (128 cores in total). In [43] on the A64FX in the FX1000, one of the specific implementations of dense matrix-vector multiplication was studied, and it was found that the use of a sector cache makes it possible to suppress performance degradation at increasing vector length.

Sparse matrix-vector multiplication (SpMV) can be even more important for various applications, for example, for quantum chemistry (in the method of configuration interaction) or quantum chromodynamics. Possibly, the solution of the problem of multiplication of a sparse matrix by a vector on the A64FX was first undertaken with the use of A64FX processor emulation, in [86]; however, only calculations on a real processor are considered here. As the highest performance achieved in SpMV, one can probably point to about 220 GFLOPS, which was obtained on the A100 [47].

Figure 7 [43, 87] shows the achievable performance in SpMV for large and small matrices with different input data choices indicated on the horizontal axis — for A64FX/2.2 GHz, V100 and a dual-processor server with Xeon 9242 (96 cores in total; in the figure — CLX-AP). In



(*a*) small matrices (*b*) large matrices

FIGURE 7. Comparison of SpMV performance on A64FX with V100 and Xeon 9242 (CLX-AP) based on Figure 6 in [87])

comparative calculations, the SELL-C-σ format was used, in which the highest performance was achieved on the A64FX [43]. The performance of the V100 and A64FX turned out to be quite close (certainly less than 150 GFLOPS), and the CLX-AP often led in performance on small matrices, but was inferior on large matrices. The limit between large and small matrices was chosen as the total capacity of the L2/L3 caches in the CLX-AP.

In [43,87], an analysis of power consumption modes on the performance of A64FX in SpMV was also carried out. Frequency reduction from 2.2 GHz to 2.0 GHz reduces performance by an average of 2.7%, but reduces power consumption by about 13%. Enabling the eco-mode also reduces power consumption by another 21% (31% in total).

In [88], the achieved performance of SpMV on A64FX/1.8 GHz (in FX700) was studied. When scaling to 12 cores (in the CMG), 31 GFLOPS was achieved, and the maximum performance on the entire node was 131 GFLOPS.

HPCG is another benchmark for HPC, where performance also depends heavily on memory bandwidth [89]. Below (Table 6) is the performance data from [89] in Fugaku — up to 8 nodes with A64FX/2.2 GHz, connected via TofuD, compared to clustered systems of two-socket servers: based

Table 6. Comparison of performance of computing systems on ARM and x86-64 server processors in HPCG and OpenSBLI benchmarks

Computing system	1 node	2 nodes	4 nodes	8 nodes	% from peak performance
HPCG benchmark (GFLOPS)					
A64FX/2,2 ГГц[1]	38,3	78,9	157,5	313,5	1,1
EPCC NGIO[2]	37,6	73,9	147,9	292,6	2,0
Fullhame[3]	33,8	67,7	133,3	261,3	3,0
OpenSBLI benchmark (calculation time in seconds)					
A64FX/2,2 GGz	3,4	1,9	1,0	0,7	
EPCC NGIO	1,2	0,8	0,5	0,3	
Fullhame	1,2	0,7	0,6	0,3	

[1]with TofuD; [2]with FDR Infiniband; 2× Xeon 8260M; [3]with Aries; 2× ThunderX2/2,2 GGz

on Cascade Lake (linked via Intel OmniPath) and based on 32-core ThunderX2/2.2 GHz (linked via Infiniband EDR). At the same time, HPCG source codes were optimized for Xeon and ThunderX2, which gave a significant performance boost — but this was not done on the A64FX.

Therefore, the HPCG performance advantages of the A64FX shown in this table over ThunderX2 and Cascade Lake may be even greater. At the same time, the number of cores in the node with A64FX is the same as the number of cores in the node with Cascade Lake, and less than in the node with ThunderX2. For A64FX, the performance leadership vs Xeon becomes more pronounced on several nodes, which may also be associated with efficient work with TofuD [**89**].

There are two sets of cycles created at Livermore Laboratories USA that are used in HPC benchmarks. The classic long-used set [**90**] in Fortran contains typical cycles from some HPC programs, which are used primarily for performance evaluations in vectorization. This set was used in [**34**] to study performance on a single A64FX/2.2 GHz core.

Another set, *RAJA Performance Suite*, which includes C++ looping kernels (including dot product) for HPC with OpenMP parallelization, was used in [**83**] to compare the performance of A64FX/2.0 GHz from FX1000 with two-socket servers with ThunderX2/2.0 GHz (56 cores in total) and Xeon Skylake/2.6 GHz (24 cores in total — probably Xeon 6126) with

different number of threads. In three of the four kernels used from RAJA, A64FX was several times ahead of Skylake, and ThunderX2 from A64FX was always far behind.

Benchmarks generally used in HPC sometimes include separate components that are themselves other well-known benchmarks. For example, HPCC [91] includes HPL, DGEMM, stream and other benchmarks. In [39, 59], performance data are considered on several benchmarks from HPCC, including DGEMM and HPL (they were discussed above), as well as in the FFT (in FFTW). The highest performance on a node with A64FX/1.8 GHz (26 GFLOPS — 0.9% of peak performance) was achieved using FFTW3 from Fujitsu [39]; in [59] it is slightly lower. At the same time, dual-processor nodes with x86-64 processors showed higher performance: 46 GFLOPS (1.5% of the peak) on the Xeon 8160 — also with 48 cores; 72 GFLOPS (1.6% of peak) on EPYC 7742 — with 128 cores per server. When parallelized to several nodes (up to 8), the clear performance advantages of x86-64 processors were preserved [59].

Another famous benchmark that was also run on the A64FX is NPB, which also includes a number of computational components, most of which are based on CFD codes. NPB also has a variant with OpenMP parallelization [92], which was used in the A64FX/1.8 GHz tests in [59], where testing included 6 components from NPB. On one core, the A64FX lost a lot in performance to the Xeon 8160 core; on all cores, the A64FX outperformed Skylake on two NPB components, and narrowed the gap on the other NPB components. The level of parallelization on A64FX was higher than on Skylake. A64FX performed better on NPB components where memory bandwidth is important, while Skylake won in more computationally intensive calculations.

SPEC cpu2017 should probably be considered the most massive benchmark. Authors of [93] measured performance in SPECrate for A64FX, but this was based on using the gem5-based A64FX simulator, and these results are not discussed here. In [50], data was obtained for A64FX SPECspeed — both integer and floating point, and SPEC OMP2012, but the values given in [50] are relative and show only conditional performance obtained by one compiler in relation to another. It was found in [94] that the A64FX in Fugaku in SPEC cpu2017 lagged behind a dual-processor server with Xeon 8168 (48 cores per server) in performance, and in SPEC OMP tests it also lagged behind a 28-core Xeon (the reason for this was

specified support for SMT mode in Xeon). But in some floating point SPEC cpu2017 benchmarks, and SPEC OMP, due to higher memory bandwidth, A64FX had better performance than Xeon.

The main results of the GRAPH500 benchmarks on Fugaku were discussed above in subsection 3.1. These benchmarks are discussed in detail in [**74**, **95**, **96**], where the dependence of BFS performance on the number of nodes is investigated, and the peculiarities of using the TofuD topology are indicated. The dependence of performance on the choice of power consumption mode has also been studied. Since BFS does not work with floating point numbers, turning on the eco-mode was expected to be an effective solution. Respectively, the optimal combination of high performance and power consumption was given by the boost mode + eco-mode.

Another benchmark performed on Fugaku's A64FX is *HIMENO*, which focuses on CFD (Incompressible Fluid Analysis) and solves Poisson's equation by the iterative Jacobi method. This benchmark has become quite common in HPC; it runs on SX-Aurora TSUBASA, there is also an option for a cluster with GPUs in the nodes. According to [**13**], 346 GFLOPS were obtained on the A64FX, 305 GFLOPS on the V100, 286 GFLOPS on the SX-Aurora TSUBASA, and only 85 GFLOPS on a dual-processor server with a Xeon 8168.

In [**97**] on the A64FX/1.8 GHz, performance was investigated on an improved sorting algorithm using vectorization with work on large and small arrays. The speedup achieved by using this algorithm was lower than that obtained on the Xeon 8170 with AVX-512.

Above, in the section about development tools for A64FX programs, some shortcomings in compilers were pointed for A64FX in the vectorization of elementary functions. In [**59**], there is comparative data on the performance of A64FX/1.8 GHz and Xeon 6140 (see Figure 8 based on Figure 2 in [**59**]). This figure uses the same notation as Figure 3, and CPE stands for Cray Programming Environment. It can be seen that Skylake here has performance advantages.

In general, these benchmark data show that A64FX is often inferior in performance to x86-64 processors, and the newer Xeon Ice Lake can be even better in this regard. The performance benefits of the A64FX for HPCs are more often applied when performance is limited by memory bandwidth.

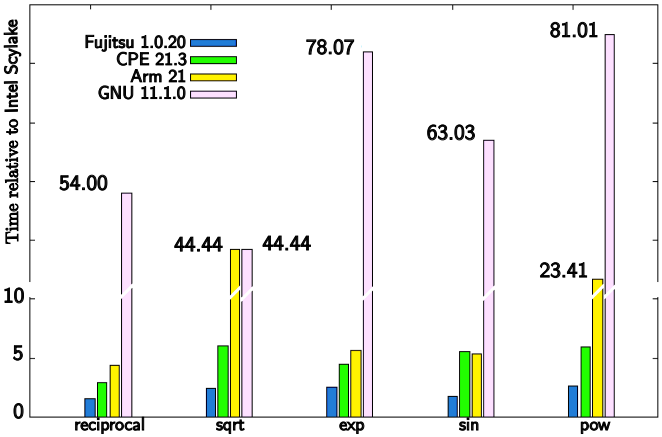


FIGURE 8. Execution time on A64FX (relative to Skylake) elementary functions

3.3. Performance data of A64FX on applications

At the beginning of this section, we will illustrate a figure available in a number of publications showing the increased performance of the A64FX/2.2 GHz compared to Xeon Cascade Lake with the same number of cores (relative to a two-processor server with Xeon 8268/24 cores 2.9 GHz) at almost twice as much low energy consumption (Figure 9 is based on [**38**, p. 24]). The A64FX outperformed Cascade Lake in all applications (except application for weather prediction, MPAS), although Intel has more powerful processors, including the newer Xeon Ice Lake family, and the cost metrics should also be compared.

The data presented in this figure refers to the various applications available in the source code. To understand what this Figure 9 gives, it is important what exactly was computed — this may determine what will be limit calculation time, and in applications of quantum chemistry, methods that are radically different in terms of performance can be used in general. This figure shows different calculations for different applications, where A64FX can be more efficient than Cascade Lake, for example, due to higher memory bandwidth. For HPC applications, more detailed information may be needed, and brief additional information on the tests presented in this figure can be found in [**34**].

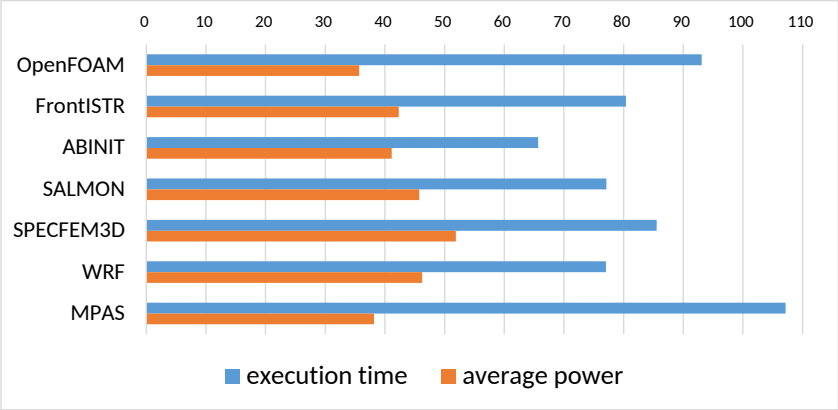


Figure 9. A64FX relative performance data for applications

In addition to the data from [38], related to A64FX from FX1000, a mapping to the same Cascade Lake server is available for A64FX from FX700 [98], where it includes 4 applications out of 7 used in [38]. On two computational chemistry applications (LAMMPS for molecular dynamics and Quantum Espresso for quantum chemistry), the performance of the A64FX was almost identical to that of the Xeon server, while on the other two applications it was significantly better.

This part of the review further discusses the applications and their corresponding mini-applications, although the latter's performance in complex HPC areas may not be a good estimate of the performance of the entire application. But to begin with, it is useful to illustrate the general situation with applications available for the A63FX using the example of computational chemistry applications (see Table 7 below). First, a number of applications mentioned as being available to run on A64FX are simply ported to AArch64, and may not have SVE support. And, for example, the developers of the ABINIT quantum chemical program were still working on testing it on the A64FX, and it is likely that Figure 9 refers to the Japanese ABINIT-MP program running on the A64FX (see also [13]), which implements calculations only for very large molecules specialized approximate FMO method, to which in this figure related data.

TABLE 7. Computational chemistry programs that can run on the A64FX

Package, version	Issuing year w/ARM	URL[1]	Availability of performance data
Quantum chemistry			
Gaussian 16, Rev. C01	2021	https://gaussian.com/relnotes/	—
Gamess-US, R1	2021	https://www.msg.chem.iastate.edu/gamess/versions.html	—
ABINIT, 9[2]	2021		[38]
CASTEP, 18.1.0	2020		+
VASP[2]	2019		—
NWChem, от 6.8[2]	2018	https://gitlab.com/arm-hpc/packages/-/wikis/packages/nwchem [4]	—
CP2K, 8.2[2]	2021		
Quantum Espresso[5]		https://github.com/fujitsu/oss-patches-for-a64fx	
SALMON, v.2.0.1	2020	http://salmon-tddft.jp/download/Manual_SALMON-v.2.0.1_simple_20210129.pdf	[38]
NTChem	2019	https://gitlab.com/arm-/hpc/packages/-/wikis/packages/ntchem	+[6]
Molecular dynamics			
GROMACS	2021	https://manual.gromacs.org/2021/release-notes/2021/major/highlights.html	+
LAMMPS		https://github.com/fujitsu/oss-patches-for-a64fx	+
NAMD[2]	2019		—

[1]only URLs shown for versions that support A64FX; [2]for AArch64; [3]in [38];
[4]For A64FX: https://static.linaro.org/ . . ./TheFirstSVEEnabledArmProcessor_A64FXandBuildingupArmHPCEcosystem5.pdf
[5]made by Fujitsu; [6]NTChem-MINI, https://github.com/fiber-miniapp/ntchem-mini;

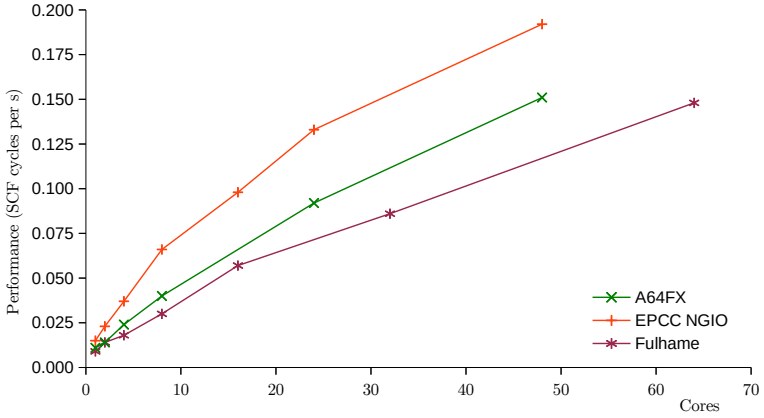


Figure 10. Comparative performance of A64FX in CASTEP

Other performance data that can be attributed to the area of quantum chemistry is the calculation of TiN with CASTEP 18.1.0 (see Figure 10 based on [89, p. 18]), where performance on A64FX/2.2 GHz is compared with two-processor servers: with Xeon 8260M/2.4 GHz (total — also 48 cores) and with ThunderX2/2.2 GHz (total — 64 cores). The data on the vertical axis reflects performance, and the A64FX clearly outperforms ThunderX2 and lags behind Cascade Lake. Table 8 shows the performance of these two-socket servers in relation to the A64FX. But it should be kept in mind that CASTEP is focused on solids research and work on the basis of plane waves, where calculations are limited by the FFT, and in quantum chemistry usually work with the basis of Gaussian orbitals.

Another quantum chemical application ported to the A64FX is SALMON (the TDDFT implementation), which, like ABINIT above,

Table 8. Relative performance of two-socket servers with other processors in relation to a server with A64FX

Server with:	minikab[1]	nekbone	CASTEP
A64FX	1,0	1,0	1,0
Xeon 8260M	$\sim 0,9$	0,3	1,3
ThunderX2	$\sim 0,5$	0,4	0,97

[1]On one processor core.

outperforms a two-processor server with Xeon 8268 (also having 48 cores, 2.9 GHz). However, TDDFT in chemistry is still not widely used, like DFT itself. And problems with memory bandwidth when working with TDDFT were discussed even before the advent of A64FX, in [99], when the authors were working on a K-computer.

A more natural application for quantum chemists tested on the A64FX is perhaps NTChem [50, 100]; more precisely, the mini-application *NTChem-mini* was tested, which is part of the NTChem for calculations by the RI-MP2 method, taking into account electron correlation [?130], which uses parallelization in MPI and XcalableMP, which is now moving in the PGAS direction [101]. Comparison of performance on NTChem-mini (on Fugaku) with a K-computer depending on the number of nodes is available in [101], while MPI gave almost the same performance as parallelization on XcalableMP. It is also useful to note that for calculating the energy and its gradient in MP2 on ARM systems, the memory bandwidth was found to be important long before the advent of the A64FX [9].

The mini-application for the fast multipole method (miniFMM) parallelized in OpenMP is referred here to computational chemistry, since this method can also be applied to large molecular systems (although, due to general mathematics, it can also work at the level of galaxy). In [83] , it was found that a node on the Fugaku A64FX/2.0 GHz was far ahead of a server with two 28-core ThunderX2/2.0 GHz, but was noticeably inferior in performance to a server with two 12-core Xeon Skylake/2.6 GHz (probably Xeon 6126) up to 24 threads, and at 48 threads it was already ahead due to the larger number of processor cores.

The tasks of computational chemistry also include the application BUDE (Bristol University Docking Engine), also parallelized in OpenMP, where the interaction energy of a protein with a ligand is calculated without the use of quantum mechanics. Fugaku A64FX/2.0 GHz in comparison with the servers indicated in the previous paragraph looked similar in performance to miniFMM: it was far ahead of ThunderX2, and up to 24 threads was significantly inferior in performance to the server with Skylake, surpassing it by 48 threads [83]. In [102] the performance of a BUDE-based mini-application (miniBUDE) is reviewed.

Molecular dynamics applications belong to the computationally intensive HPC group, and run efficiently on the GPU.

Table 9. Performance in different power modes (PM) on applications of computational chemistry and bioinformatics in relation to the normal mode N

Application	B		N+E		B+E	
	П	М	П	М	П	М
GENESIS	1,09	1,20	1,0	0,8	1,09	0,96
Genomon	1,10	1,17				
NTChem	1,08	1,21	0,57	0,69	0,62	0,83
RSDFT	1,06	1,20	0,71	0,8	0,77	0,9

B — boost mode; E — eco mode;

GENESIS — molecular dynamics application for biomolecular systems (https://www.r-ccs.riken.jp);

Genomon — bioinformatics application (https://genomon.hgc.jp/);

RSDFT — quantum chemical application.

A detailed analysis of molecular dynamics calculations using the LAMMPS program and ways to possibly increase performance at A64FX/2.2 GHz, including using vectorization and various parallelization variants, was carried out in [**103**]. For another famous molecular dynamics application, GROMACS, which focuses on biomolecular systems, the use of C++17 overrides the possibility of using the Cray 10.0.1 and ARM 20.3 compilers, and gcc 10.2.0 was used. On the A64FX/1.8 GHz, GROMACS (for a system of 87k atoms) reached 20.2 ns/day, and the dual-processor variant with the Xeon 8160 (48 cores in total) reached 75.6 ns/day. The reasons for Skylake's large performance gain are not clear; perhaps this was facilitated by the absence of vectorization of elementary functions in gcc [**39**].

In conclusion of the part about applications in computational chemistry, let us also consider the data from [**34**], see Table 9, which slightly illustrate the possibilities of using different power consumption modes in the A64FX, including for applications in the field of bioinformatics. Here you can see that the boost mode has always given a performance boost compared to the normal one, but for the GENESIS application, the combination of boost and eco-modes is optimal, which does not give a noticeable decrease in performance. And for quantum chemical applications, turning on the eco-mode gave a large decrease in performance.

Among the applications in computational chemistry, one of the main parts is quantum chemistry, where quantum mechanics is applied to

chemical objects, but purely quantum mechanical calculations that are not related to quantum chemistry were also carried out on the A64FX.

In [64], using the BLIS library, calculations were carried out according to the improved method of the authors — on the A64FX, Xeon 8260, and EPYC 7702, but the performance between different processors was not directly compared. In [104], in the DCA++ application for quantum Monte Carlo calculations with parallelization on OpenMP, A64FX and ThunderX2 were compared using ARMclang 20.3 and ArmPL 20.3. The OpenMP SIMD directives were also used here, but from the point of view of comparative performance, only the obvious advance of the A64FX (by several times) relative to ThunderX2 is shown.

The above data indicates that modern x86-64 processors (there is no data comparing the performance of A64FX with the newer Xeon Ice Lake) often outperform A64FX in computational chemistry tasks. The traditional estimates that, in a typical case, 2 GB of memory per processor core are needed for quantum chemical calculations, are unacceptable for a multi-core A64FX and require additional analysis. According to another rough estimate, good for a typical quantum-chemical calculation of the ratio of 1 GB/s in memory per 1 GFLOPS, the A64FX processor looks more attractive than others. But the user will still watch the efficiency and performance of the application.

In [41], BWA-MEM2 software tools were ported to the A64FX for solving genomics problems related to HPC; however, the achieved performance of the A64FX on Fugaku was lower than on the Xeon 8160. Although the memory bandwidth in the A64FX is much higher, the large memory latencies in the A64FX could have an effect here [41].

As in quantum chemistry, quantum mechanics is also used in QCD, the application performance of which was actively studied on the A64FX with calculations on the FX700 and FX1000. In the used lattice QCD model, it is important to multiply a matrix by a vector [43, 87, 105]. The performance of SpMV on the A64FX was discussed above. Here we will focus on the performance in lattice QCD. It is important to note here that the use of A64FX for QCD calculations was assumed to be very promising, and a special QPACE 4 project was created to create a supercomputer for parallel QCD calculations based on A64FX, in which Cray also participated. This system, using the FX700, where 64 nodes are connected via Infiniband EDR, went live in 2020 [54].

In [43], within the framework of the GRID application (C++, with OpenMP + MPI parallelization) for lattice QCD, the performance of the Domain Wall (DW) kernel was studied. Using interleaved storage form RIRI (R-real, I-imaginary), the A64FX (from Fugaku) achieved performance of about 440 GFLOPS. But the RRII form was found to be more effective; it achieved 182 GFLOPS on the CMG (on 12 cores), and 712 GFLOPS on the entire A64FX, 12% more than with RIRI.

In [43] also investigated the effect of choosing different A64FX power consumption control options for RIRI and RRII. Power consumption with RIRI was higher than with RRII. Unlike RIRI, when working with RRII, it was possible to reduce power consumption by reducing the clock frequency (from 2.2 to 2.0 GHz) and enabling eco-mode, while performance was not significantly reduced. In [87], for similar DW calculations, it was found possible to reduce the power consumption on such an A64FX by 18%.

The A64FX performance data for various lattice sizes (on one of the coordinates) using the RRII scheme was compared with the performance on the V100 and a two-socket server with a Xeon 9242 (96 processor cores in total), but they used only the RIRI scheme. With the RRII, the A64FX was almost always the fastest; with the RIRI, depending on the size of the grid, sometimes was faster than the A64FX, sometimes was faster the V100, and the Xeon Cascade Lake was significantly inferior to them. But to achieve high performance for QCD problems on the A64FX in [43], it was necessary to upgrade the code.

In [43, 87], a decrease in the achievable performance of lattice QCD on the A64FX was noted due to large command delays and paucity of resources for OoO, including the small size of the ROB buffer.

This performance data on Fugaku with A64FX/2.2 GHz was compared with performance on the QPACE 4 supercomputer with A64FX/1.8 GHz [53, 54]. On QPACE 4, calculations were made using SP and DP, but the performance data given is for DP. With the RIRI scheme on the Wilson Dslash (WD) kernel, 376 GFLOPS per node were obtained, and the dependence of performance with SP and DP depending on the number of nodes (up to 32) was studied. In the DW (Domain Wall) kernel, 475 GFLOPS per node were obtained, and the same dependence on the number of nodes as in WD was investigated. The RRII scheme also gave better performance here than RIRI; about 20% higher performance was obtained on the DW kernel than with RIRI (different lattice variants were studied).

Although the clock speed of the A64FX on the QPACE 4 is 1.22 times lower than the Fugaku, the resulting performance is down to around 25%. However, some A64FX features that were used on Fugaku were not available on QPACE 4, including sector cache or eco-modes [**54**]. In [**53**, **54**] it was noted that the optimization of the codes of these kernels required manual work, and in [**54**] — the lack of progress in the work of compilers with SVE over the past year.

For lattice QCD, GRID is not the only application for the A64FX; [**75**] also mentions Bridge++, BQCD, and LDDHMC. In addition, there is a special library developed by Fujitsu for QCD problems — QWS [**106**] . The optimization methods used in QWS are general, except for the use of ACLE and tools for working with TofuD [**76**].

Using QWS and BiCGstab (biconjugate gradient stabilized method) with OpenMP + MPI parallelization on 147456 Fugaku nodes in boost mode (at 2.2 GHz) without enabling eco-mode, the 102 PFLOPS SP performance indicated above in section 3.1. was obtained, and energy efficiency 5 GFLOPS/W [**75**]. Detailed information about this calculation and performance data, starting from the two CMGs involved, is given in [**76**].

In [**105**], for lattice QCD, the performance of their object-oriented code, and when using BiCGstab with mixed precision, was compared, among other things, on A64FX and V100; while their code gave better performance and scaling. It was also noted here that the code needs to be adapted to the specific architecture being used in order to achieve maximum performance.

The next group of applications, the performance of which was obtained on the A64FX, relates to relativistic mechanics and astrophysics. In [**39**, **107**], the performance on the A64FX in Ookami was studied in the FLASH application, which is oriented to use in various fields of physics, including astrophysics, but problems with compilers, including problems in the generation of codes with SVE support, did not give interesting results.

In [**108**], the performance of the A64FX (at Apollo 80, at Ookami) was studied within the framework of the VPIC 2.0 application using the Kokkos tools and focused on solving problems of relativistic plasma physics (they are also important for astrophysics). Figure 11 compares A64FX/1.8GHz performance with EPYC (Zen 2) and Xeon Cascade Lake performance.

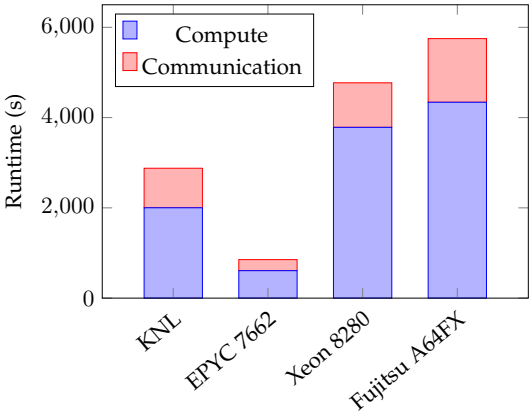


Figure 11. Performance comparison on VPIC in clusters: compute and communication time (Figure 11 is based on Figure 10 in [108]

x86-64 processors were used in clusters with dual processor nodes connected by Infiniband HDR. The processor time on eight EPYC 7742s was 6 times less than on eight A64FX, and the communication time on the EPYC system was also several times less. The 8 Xeon 8280 processors also significantly outperformed the eight A64FX in these parameters. Due to the inefficient implementation of vectorization in Kokkos, only EPYC gave comparable performance to the V100 and A100 GPUs [108].

A large number of applications for which data on their performance on the A64FX have become available do not use either quantum or relativistic physics. They cover different areas such as CFD and weather prediction.

Thus, in [39, 52, 58] on A64FX/1.8 GHz, performance data were obtained on the well-known weather prediction application SWIM, parallelized in OpenMP. In [56], a performance study was done on the A64FX (on Fugaku) of the famous Nek5000 application for turbulent atmospheric phenomena. The performance of the Nek5000 has been studied over a wide range of Fugaku nodes. In CEED's BP3 benchmark, which uses similar math, the A64FX is compared to the V100 on Summit and Xeon Gold 6130/2.1GHz (16 cores). A64FX and V100's performance was sometimes close, and sometimes V100 outperformed A64FX by 2-3 times, while Skylake always lagged behind. In [89], Nekbone mini-application for

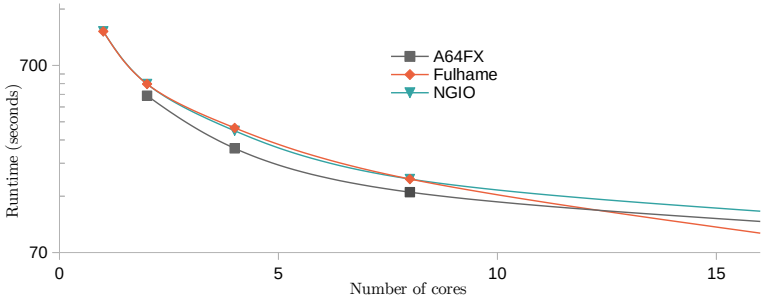


FIGURE 12. Performance in COSA

Nek5000 for A64FX/2.2 GHz achieved 176 GFLOPS, more than dual processor servers: 127 GFLOPS with Xeon 8260M (also only 48 cores/2.4 GHz), and with ThunderX2 (total 64 cores/2.2 GHz) — 122 GFLOPS.

The performance of the same computing systems was compared on another CFD application, COSA, in [**89**], but here these computing systems acted as cluster nodes with TofuD interconnections for A64FX, Infiniband EDR for ThunderX2, OmniPath for Xeon 8260M. In Figure 12, which shows the graphs of the calculation time versus the number of nodes used, the system with ThunderX2 is marked as Fulhame, and with Xeon — as NGIO. A64FX here requires less time to calculate with a small number of nodes. But in general, the performance of these computing systems when working with COSA is quite close. Another CFD application, ShallowWaters.jl (for liquid circulation), created in [**109**] on A64FX to work with half precision, accelerated the calculation by 3.6 times compared to DP, which may become topical for weather and climate modeling in the future.

For CFD on the A64FX, a lot of performance data has been obtained. Thus, in [**83**], the obtained data on A64FX/2.0 GHz performance on LULESH and CloverLeaf applications (parallelized in MPI+OpenMP) are compared with performance on dual-processor servers: with Xeon Skylake with 24 cores (in total) at 2.6 GHz (probably Xeon 6126) and 28-core ThunderX2/2.0 GHz. In LULESH, the A64FX was inferior in performance to both ThunderX2 and Skylake — also because memory latency is important in this application; similar data is available in [**38**]. In terms of performance in the proxy application, LULESH A64FX/1.8

GHz was significantly slow for a two-processor server with Xeon 6130 (32 cores per server) [59].

And in CloverLeaf, with a different number of OpenMP threads and MPI processes, A64FX was always ahead of ThunderX2, and usually ahead of Skylake as well [83]. According to [38], the performance comparability of A64FX with CloverLeaf was achieved only when working on two dual-processor servers with Skylake. This advantage of the A64FX is due to the fact that the performance of the CloverLeaf mini-application is limited by memory bandwidth [34].

On the A64FX/2.2 GHz, the performance of another CFD-related application, OPenSBLI, was studied. Its originality for HPC is associated with the use of Python source code, which, in order to solve specific differential equations written in Einstein notation, generates C code that will use MPI + OpenMP parallelization [89]. Table 6 above shows the performance obtained in clusters containing up to 8 nodes: with A64FX (with connected TofuD nodes) and clusters of two-socket servers: with Xeon 8260M (total 48 cores/2.4 GHz) — with OmniPath interconnect, and with ThunderX2 (total 64 cores/2.2GHz) — with Infiniband EDR interconnect. Here, for any number of nodes, a cluster with A64FX is several times behind in performance than alternative clusters with the same number of nodes; a preliminary analysis of the reasons for this suggested a possible modification of the OPenSBLI code in order to improve performance on the A64FX [89].

The PENNANT mesh physics mini-application [110] is based on the FLAG code used for radiative CFD problems. In [39,52,58] the performance in PENNANT on A64FX/1.8 GHz with OpenMP parallelization (up to 48 cores) was studied.

In a number of papers [39,52,58,60] there are data comparing the performance of A64FX with other processors and GPUs in the minimod mini-application for seismic modeling problems [60]. In [39,52,58] the data on the acceleration achieved on the A64FX with parallelization on OpenMP depending on the number of processor cores used is given.

In [60], MPI parallelization was used, and the value of the achieved speedup was analyzed depending on the number of cores (up to 48) on A64FX and EPYC 7702 (Roma, 64 cores); on the A64FX, the acceleration with the increase in the number of cores became much greater than on the Roma. As for the performance of minimod, calculations

were performed many times faster on A64FX than on Roma, and the AMD server was sometimes significantly ahead, and sometimes lagged behind the two-processor server with Xeon 5112 (24 cores in total). The A64FX has always been ahead of the x86-64, but lagged behind the V100 in performance [60].

In [38, 71] performance data on the A64FX from Fugaku is presented for another mini-application parallelized on MPI + OpenMP — Tealeaf, which solves the linear heat equation. The performance obtained at the same time on the A64FX (dependence on the number of OpenMP threads and MPI rank was also studied) was higher than that of a two-processor server with Xeon Skylake (24 cores/2.6 GHz in total) and a two-processor server with ThunderX2 (56 cores in total). /2.0 GHz). At the same time, the resulting performance of the A64FX was twice that of two dual-processor servers with Skylake. But in this mini-application, performance is limited by memory bandwidth [38].

The A64FX performance data of applications/mini-applications just reviewed differed in that they do not use either quantum or relativistic mechanics. Below is data on other applications studied on the A64FX and relatively less used. For example, in [84], adaptive optics problems were used on the A64FX, which must be solved in real time. A64FX/2.2GHz performance benchmarked against dual-socket servers with Xeon 6248 (40 cores/2.5GHz total), EPYC 7702 (128 cores total/2.2GHz), and GPU A100, V100, and AMD MI100. The computation time on the A64FX was comparable to that obtained on those GPUs (and on the NEC SX-Aurora TSUBASA), and usually significantly less than on x86-64 servers. The calculation time here strongly depends on the efficiency of the implementation of matrix-vector multiplication.

In [89] on a cluster system with A64FX/2.2 GHz in nodes connected by TofuD, the performance of the minikab (Mini Krylov ASiMoV Benchmark) mini-application using the conjugate gradient method and parallelized in MPI + OpenMP was studied in comparison with a cluster of two-processor servers with ThunderX2/2.2GHz 32-core processors (with Infiniband EDR). Used up to 6 nodes with ThunderX2 and up to 8 nodes with A64FX (up to 384 cores in each cluster). With an equal number of processor cores, the cluster with A64FX significantly outperformed the cluster with ThunderX2 in terms of performance. This also corresponds to the comparative estimates of minikab performance on a single processor core

presented in [89]: in A64FX it is twice as fast as in ThunderX2 and 7% faster than Xeon 8260M.

Due to the recent extension of AI workloads, which could provide a potentially wider market than traditional HPCs, the performance of the A64FX has also been studied on such workloads. For AI, half-precision calculations are more often important, especially with bfloat16 — with the same 8 bits for the exponent as in SP — which is not typical for traditional HPC with DP; therefore, data on such performance are considered very briefly and at the end of this section (3.3.) of the review.

Fugaku's performance data on the world-famous HPL-AI supercomputer benchmark is shown above in Section 3.1. In general, AI was also used in the task of emulating the A64FX itself — the resulting latencies in command execution were checked [44].

[111] obtained performance data on Fugaku for a ResNet-50 residual neural network for image classification, and in the Chainer application using deep learning (in Python using NumPy), linear performance scaling (number of processed images per second) was obtained with an increase in the number of Fugaku nodes up to 8. ResNet-50 is now generally used, for example, for the tasks of detecting COVID-19 in x-rays [112]. In [113], A64FX/2.2 GHz performance on ResNet-50 is compared with the performance of a two-socket server with Xeon 8268 (48 cores/2.9 GHz total). The performance of the A64FX was only slightly lagging behind, but the energy efficiency of the A64FX was 2.8 times higher than that of the Xeon.

Based on the performance data in sections 2 and 3 above, it is clear that the Fujitsu processor almost always outperformed the ThunderX2 significantly, which was to be expected. All of the benchmark and application performance data for the A64FX suggests that the A64FX takes precedence over x86-64 server processors, usually when high memory bandwidth is required to achieve high performance. And in most applications and benchmarks, modern x86-64 processors from Intel and AMD get better performance. GPUs in their natural areas of application usually have a much higher performance than the A64FX. In addition, to achieve high performance, A64FX often requires modifying the source code of programs.

Conclusion

The A64FX's performance and data of energy efficiency indicate that the use of the A64FX in supercomputers is now likely to expand. And we can draw a more general conclusion, including on the basis of data on the ThunderX2 and A64FX ARM processors used in supercomputers, that the hegemony of x86-64 processors in HPC will clearly decrease — this has actually been said, for example, in the documents of the European HPC-PRACE infrastructure. But the data of the A64FX microarchitecture also shows its certain disadvantages compared to modern x86-64 server processors, including increased latencies in a number of instructions and when working with memory, and the paucity of OoO resources, which can be partly dealt with. In addition, the software development tools for the A64FX are still in need of improvement. Simply translating source code with A64FX compilers without converting the source code for optimization may not give the expected performance. The performance benefits on the A64FX come when the benefit comes from the drastically higher bandwidth of the HBM2.

In benchmarks and applications A64FX, perhaps, loses more often in performance to Xeon processors starting with Skylake (and there are no comparisons at all with Ice Lake, where the highest performance should appear) and EPYC Roma/Milan. It seems that the performance analysis should explicitly include data on the price and actually consumed energy, that is, also a cost indicator. And now it is impossible to determine which of the AArch64 processors will lead in the near future in the process of pushing x86-64 out of absolute leadership in HPC: Fujitsu A64FX, SiPearl Rhea, Nvidia Grace or some other (for example, Huawei Kunpeng 930 — if its production starts soon).

There are publications about studies conducted, including in RIKEN, on possible ways of developing the A64FX, for which processor emulation was used (Gem5 with added SVE emulation). In [**37**], for a 1024-bit SVE, a possible performance improvement with reduced power consumption was shown. And in [**5**], both the possibility of simply increasing the number of cores and increasing the length of SVE vectors up to 2048 bits were studied using 5 and 3 nm technologies. It can be assumed that this will be a possible main focus for Fujitsu for future based on A64FX.

References

[1] A. Tekin, Tuncer Durak A., C. Piechurski, D. Kaliszan, Aylin Sungur F., F. Robertsén, P. Gschwandtner. *State-of-the-art and trends for computing and interconnect network solutions for HPC and AI*, Technical report, PRACE, 2021, 38 pp. ↑132, 140, 141

[2] I. Liabotis. *EINFRA-4-2014: Pan-European high performance computing infrastructure and services*, PRACE-4IP-EINFRA-653838, PRACE, 2017, 71 pp. ↑132, 134

[3] C. Edwards. "Moore's Law: what comes next?", *Communications of the ACM*, **64**:2 (2021), pp. 12–14. ↑132

[4] J. Domke, E. Vatai, A. Drozd, P. Chen, Y. Oyama, L. Zhang, S. Salaria, D. Mukunoki, A. Podobas, M. Wahib, S. Matsuoka. "Matrix engines for high performance computing: A paragon of performance or grasping at straws?", 2021 IEEE International Parallel and Distributed Processing (IPDPS) (17–21 May 2021, Portland, OR, USA), 2021, pp. 1056–1065. ↑132, 141, 161

[5] E. Arima, Y. Kodama, T. Odajima, M. Tsuji, M. Sato. "Power/Performance /Area Evaluations for Next-Generation HPC Processors using the A64FX Chip", 2021 IEEE Symposium in Low-Power and High-Speed Chips (COOL CHIPS) (14–16 April 2021, Tokyo, Japan), 2021, pp. 1–6. ↑132, 161, 163, 184

[6] M. S. Gordon, G. Barca, S. S. Leang, D. Poole, A. P. Rendell, J. L. Galvez Vallejo, B. Westheimer. "Novel computer architectures and quantum chemistry", *The Journal of Physical Chemistry A*, **124**:23 (2020), pp. 4557–4582. ↑132, 139

[7] E. Calore, A. Gabbana, S. F. Schifano, R. Tripiccione. "ThunderX2 performance and energy-efficiency for HPC workloads", *Computation*, **8**:1 (2020), pp. 20. ↑132, 133

[8] A. Tiwari, K. Keipert, A. Jundt, J. Peraza, S. S. Leang, M. Laurenzano, M. S. Gordon, L. Carrington. "Performance and energy efficiency analysis of 64-bit ARM using GAMESS", *Proceedings of the 2nd International Workshop on Hardware-Software Co-Design for High Performance Computing*, Co-HPC '15 (15 November 2015, Austin, Texas, USA), ACM, New York, 2015, ISBN 978-1-4503-3992-6, 10 pp. ↑132

[9] K. Keipert, G. Mitra, V. Sunriyal, S. S. Leang, M. Sosonkina, A. P. Rendell, M. S. Gordon. "Energy-efficient computational chemistry: Comparison of x86 and ARM systems", *Journal of Chemical Theory and Computation*, **11**:11 (2015), pp. 5055–5061. ↑132, 174

[10] O. W. Saastad, K. Kapanova, S. Markov, C. Morales, A. Shamakina, N. Johnson, E. Krishnasamy, S. Varrette. *Best Practice Guide Modern Processors*, PRACE, 2020, 109 pp. ↑132, 134, 136, 137, 142, 143, 144

[11] A. S. Antonov, I. V. Afanas'yev, Vl. V. Voyevodin. "High-performance computing platforms: current status and development trends", *Vychislitel'nyye metody i programmirovaniye*, **22**:2 (2021), pp. 135–177 (in Russian). ↑132, 133, 134

[12] J. Xia, C. Cheng, X. Zhou, Y. Hu, P. Chun. "Kunpeng 920: The first 7nm chiplet-based 64-Core ARM SoC for cloud services", *IEEE Micro*, **41**:5 (2021), pp. 67–75. ↑132, 134

[13] J. Dongarra. *Report on the Fujitsu Fugaku system*, Tech Report No ICL-UT-20-06, University of Tennessee, Innovative Computing Laboratory, 2020, 18 pp. ↑132, 141, 157, 158, 160, 163, 169, 171

[14] W. Zhang, Z. Jiang, Z. Chen, N. Xiao, Y. Ou. "NUMA-Aware DGEMM based on 64-bit ARMv8 multicore processors architecture", *Electronics*, **10**:16 (2021), pp. 1984. ↑133, 134, 139

[15] V. Frolov, V. Galaktionov, V. Sanzharov. "RISC-V: the standard that changed the world of microprocessors", *Otkrytyye sistemy. SUBD*, 2020, no. 2, pp. 30–34 (in Russian). ↑133

[16] M. Kuz'minskiy. "Power10: resurgence of RISC", *Otkrytyye sistemy. SUBD*, 2021, no. 3, pp. 10–12 (in Russian). ↑133, 146

[17] L. Jiang, C. Yang, W. Ma. "Enabling highly efficient batched matrix multiplications on SW26010 many-core processor", *ACM Transactions on Architecture and Code Optimization*, **17**:1 (2020), pp. 1–23, 2. ↑133

[18] M. Kuz'minskiy. "Chinese processor-supercomputer way", *Otkrytyye sistemy. SUBD*, 2017, no. 1, pp. 8–11 (in Russian). ↑133

[19] M. Kuz'minskiy, *Otkrytyye sistemy. SUBD*, 2020, no. 2, pp. 12–15 (in Russian). ↑133

[20] P. Ouro, U. Lopez-Novoa, M. F. Guest. "On the performance of highly-scalable Computational Fluid Dynamics code on AMD, ARM and Intel processor-based HPC systems", *Computer Physics Communications*, **269** (2021), 108105. ↑133, 136, 141

[21] S. McIntosh-Smith, J. Price, T. Deakin, A. Poenaru. "A performance analysis of the first generation of HPC-optimized Arm processors", *Concurrency and Computation: Practice and Experience*, **31**:16 (2019), e5110. ↑133

[22] N. Stephens, S. Biles, M. Boettcher, J. Eapen, M. Eyole, G. Gabrielli, M. Horsnell, G. Magklis, A. Martinez, N. Premillieu, A. Reid, A. Rico, P. Walker. "The ARM scalable vector extension", *IEEE Micro*, **37**:2 (2017), pp. 26–39. ↑133

[23] V. Soria-Pardos, A. Armejach, D. Suárez, M. Moretó. "On the use of many-core Marvell ThunderX2 processor for HPC workloads", *The Journal of Supercomputing*, **77**:4 (2021), pp. 3315–3338. ↑134

[24] R. Sugumar. "ThunderX3 next-generation arm-based server", 2020 IEEE Hot Chips 32 Symposium (HCS) (16–18 Aug., 2020, Palo Alto, CA, USA), 2020, pp. 1–19. ↑134

[25] R. Sugumar, M. Shah, R. Ramirez. "Marvell ThunderX3: next-generation arm-based server processor", *IEEE Micro*, **41**:2 (2021), pp. 15–21. ↑134

[26] W. Gao, J. Fang, C. Huang, C. Xu, Z. Wang. "Optimizing barrier synchronization on ARMv8 many-core architectures", 2021 IEEE International

Conference on Cluster Computing (CLUSTER) (7–10 Sept. 2021, Portland, OR, USA), pp. 542-552. ↑134

[27] All SPEC CPU2017 results published by SPEC. ↑136

[28] J. D. McCalpin. "HPL and DGEMM performance variability on the Xeon Platinum 8160 processor", SC18: International Conference for High Performance Computing, Networking, Storage and Analysis (11–16 Nov. 2018, Dallas, TX, USA), pp. 225–237. ↑139

[29] I. S. Kruzhilov, M. B. Kuz'minskiy, A. M. Chernetsov, O. Yu. Shamayeva. "Basic linear algebra libraries for high performance computing", *Vestnik M·YeI*, 2018, no. 6, pp. 87–95 (in Russian). ↑139

[30] X. Álvarez-Farré, A. Gorobets, F. Trias, A. Oliva. "NUMA-aware strategies for the heterogeneous execution of SPMV on modern supercomputers", 14th WCCM-ECCOMAS Congress 2020 (19–24 July 2020, Paris, France), 10 pp. ↑139

[31] M. Mahmoud, M. Hoffmann, H. Reza. "Developing a new storage format and a warp-based SpMV kernel for configuration interaction sparse matrices on the GPU", *Computation*, **6**:3 (2018), pp. 45. ↑139

[32] Y. Zhang, W. Yang, K. Li, D. Tang, K. Li. "Performance analysis and optimization for SpMV based on aligned storage formats on an ARM processor", *Journal of Parallel and Distributed Computing*, **158** (2021), pp. 126–137. ↑139

[33] I. Afanasyev, D. Lichmanov. "Evaluating the performance of Kunpeng 920 processors on modern HPC applications", *PaCT 2021: Parallel Computing Technologies*, International Conference on Parallel Computing Technologies, Lecture Notes in Computer Science, vol. **12942**, Springer, Cham, 2021, ISBN 978-3-030-86359-3, pp. 301–321. ↑139

[34] M. Sato, Y. Ishikawa, H. Tomita, Y. Kodama, T. Odajima, M. Tsuji, H. Yashiro, M. Aoki, N. Shida, I. Miyoshi, K. Hirai, A. Furuya, A. Asato, K. Morita, T. Shimizu. "Co-Design for A64FX manycore processor and "Fugaku"", *SC '20: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (9–19 November, 2020, Atlanta, Georgia, USA), 2020, ISBN 978-1-7281-9998-6, pp. 1–15. ↑142, 144, 145, 161, 167, 170, 175, 181

[35] R. Okazaki, T. Tabata, S. Sakashita, K. Kitamura, N. Takagi, H. Sakata, T. Ishibashi, T. Nakamura, Y. Ajima, *Fujitsu Technical Review 2020-03*, 2020, 9 pp. ↑141, 142, 144, 145

[36] O. Perks. *AEM and SVE*, International Workshop on HighPerformance Computing and Programming on Quantum Chemistry and Physics 2020 (HPCPQCP2020), 2020, 68 pp. ↑143

[37] T. Odajima, Y. Kodama, M. Sato. "Performance and power consumption analysis of ARM scalable vector extension", *The Journal of Supercomputing*, **77**:6 (2021), pp. 5757–5778. ↑143, 184

[38] M. Sato, T. Odajima, Y. Kodama. "Performance evaluation of the supercomputer "Fugaku" and A64FX manycore processor", ScalA workshop 2020 (12th Nov, 2020), 27 pp. ↑144, 159, 161, 170, 171, 172, 180, 181, 182

[39] A. Burford, A. C. Calder, D. Carlson, B. Chapman, F. CoŞKun, T. Curtis, C. Feldman, R. J. Harrison, Y. Kang, B. Michalow-Icz, E. Raut, E. Siegmann, D. G. Wood, R. L. Deleon, M. Jones, N. A. Simakov, J. P. White, D. Oryspayev. *Ookami: deployment and initial experiences*, 2021. arXiv 2106.08987 ↑145, 147, 148, 151, 152, 154, 156, 157, 161, 162, 164, 168, 175, 178, 179, 181

[40] Huang et al H.. "Shuhai: a tool for benchmarking HighBandwidth memory on FPGAs", *IEEE Transactions on Computers*, **71**:5, pp. 1133-1144, 12 pp. ↑145

[41] Langarita Benitez R.. *Evaluation of genome alignment workflows on HPC processors*, Master thesis, Universitat Politècnica de Catalunya, 2021, 69 pp. ↑145, 153, 163, 176

[42] C. Alappat, J. Laukemann, T. Gruber, G. Hager, G. Wellein, N. Meyer, T. Wettig. "Performance modeling of streaming kernels and sparse matrix-vector multiplication on A64FX", 2020 IEEE/ACM Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS) (12 Nov. 2020, Atlanta, GA, USA), pp. 1–7. ↑146

[43] C. Alappat, N. Meyer, J. Laukemann, T. Gruber, G. Hager, G. Wellein, T. Wettig. *ECM modeling and performance tuning of SpMV and Lattice QCD on A64FX*, 2021. arXiv 2103.03013 ↑146, 149, 153, 155, 163, 164, 165, 166, 176, 177

[44] L. Li, S. Pandey, T. Flynn, H. Liu, N. Wheeler, A. Hoisie. *SimNet: computer architecture simulation using machine learning*, 2021. arXiv 2105.05821 ↑146, 183

[45] J. Schreier. *Optimization of small matrix multiplication kernels on Arm*, Bachelor's Thesis in Informatics, Technische Universität München, 2021, 47 pp. ↑146, 154, 155, 162

[46] D. Koo, J. Lee, J. Liu, E.-K. Byun, J.-H. Kwak, G. K. Lockwood, S. Hwang, K. Antypas, K. Wu, H. Eom. "An empirical study of I/O separation for burst buffers in HPC systems", *Journal of Parallel and Distributed Computing*, **148** (2021), pp. 96–108. ↑146

[47] H. Anzt, Y. M. Tsai, A. Abdelfattah, T. Cojean, J. Dongarra. "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 Nov. 2020, Atlanta, GA, USA), pp. 26–38. ↑146, 161, 163, 165

[48] L. Zhang, T. Okamoto, S. Ishii, K. Hirai, S. Sumimoto, B. Gerofi, M. Takagi, Y. Ishikawa. *OS enhancement in supercomputer Fugaku*, Fujitsu Technical Review, 2020, 7 pp. ↑147

[49] J. Domke, K. Matsumura, M. Wahib, H. Zhang, K. Yashima, T. Tsuchikawa, Y. Tsuji, A. Podobas, S. Matsuoka,. OS 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), pp. 78–88. doi ↑147

[50] J. Domke. *A64FX–Your Compiler You Must Decide!*, 2021. arXiv 2107.07157 ↑148, 149, 150, 156, 168, 174

[51] K. I. Ishikawa, I. Kanamori, H. Matsufuru, I. Miyoshi, Y. Mukai, Y. Nakamura, K. Nitadori, M. Tsuji. *102 PFLOPS lattice QCD quark solver on Fugaku*, 2021. arXiv 2109.10687 ↑148

[52] B. Michalowicz, E. Raut, Y. Kang, T. Curtis, B. Chapman, D. Oryspayev. *Comparing OpenMP implementations with applications across A64FX platforms*, 2021. arXiv 2107.10346 ↑148, 151, 152, 156, 179, 181

[53] N. Meyer, P. Georg, S. Solbrig, T. Wettig. *Grid on QPACE 4*, The 38th International Symposium on Lattice Field Theory (26–30 Jul., 2021), 1 pp. URL ↑148, 150, 162, 164, 177, 178

[54] N. Meyer. "Grid Lattice QCD framework on A64FX", R-CCS seminar (online) (June 30, 2021, University of Regensburg, Regensburg, Germany), 2021, 29 pp. URL ↑148, 150, 164, 176, 177, 178

[55] Arm Ltd. *SVE compilers and libraries*, Arm SVE Hackathon on Ookami, February 2021, 2019, 41 pp. URL ↑148, 149

[56] T. Kolev, P. Fischer, A. P. Austin, A. T. Barker, N. Beams, J. Brown, J.-S. Camier, N. Chalmers, V. Dobrev, Y. Dudouit, L. Ghaffari, S. Kerkemeier, Y.-H. 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, US Department of Energy, 2021, 51 pp. ↑150, 155, 156, 159, 179

[57] R. J. Harrison. *Performance engineering on A64FX with SVE intrinsics (Early experience on Ookami)*, 2021, 37 pp. URL ↑150, 151, 152, 155, 162, 164

[58] B. Michalowicz, E. Raut, Y. Kang, T. Curtis, B. Chapman, D. Oryspayev. *Comparing the behavior of OpenMP Implementations with various Applications on two different Fujitsu A64FX platforms*, 2021. arXiv 2106.09787 ↑151, 179, 181

[59] M. A. S. Bari, B. Chapman, A. Curtis, R. J. Harrison, E. Siegmann, N. A. Simakov, M. D. Jones. "A64FX performance: experience on Ookami", 2021 IEEE International Conference on Cluster Computing (CLUSTER) (7–10 Sept. 2021, Portland, OR, USA), pp. 711-718. ↑151, 152, 153, 154, 156, 161, 162, 168, 169, 181

[60] J. Meng, A. Atle, H. Calandra, M. Araya-Polo. *Minimod: A finite difference solver for seismic modeling*, 2020. arXiv 2007.06048 ↑151, 181, 182

[61] D. Bailey, T. Harris, W. Saphir, van der Wijngaart R., A. Woo, M. Yarro. *The NAS parallel benchmarks 2.0*, Report NAS-95-020, NASA Ames Research Center, 1995, 24 pp. URL ↑

[62] H. Murai. *Overview of software environmenton Fugaku*, 6th Meeting for Application Code Tuning on A64FX Computer Systems (June 30, 2021), 17 pp. ↑154

[63] S. Hammond, M. Curry, K. Davis, V.-Q. Dang, O. Guba, R. Hoekstra, J. Laros, K. Pedretti, D. Poliakoff, S. Rajamanickam, C. Trott, L. Vergiat-Berger, A. Younge. *Fugaku and A64FX Update*, 2021, 15 pp. ↑154, 157, 162

[64] R.-Q. G. Xu, T. Okubo, S. Todo, M. Imada. *Optimized implementation for calculation and fast-update of Pfaffians installed to the open-source fermionic variational solver mVMC*, 2021. arXiv 2105.13098 ↑155, 176

[65] Van Zee F. G., van de Geijn R. A.. "BLIS: a framework for rapidly instantiating BLAS functionality", *ACM Transactions on Mathematical Software*, **41**:3 (2015), pp. 1–33, 14. ↑155

[66] T. Imamura. *Development of EigenExa from K to Fugaku, and beyond Fugaku*, The 4th Meeting for Application Code Tuning on A64FX Computer Systems (March 17, 2021), 2021, 24 pp. ↑155, 160

[67] N. Shibata, F. Petrogalli. "SLEEF: A portable vectorized library of C standard mathematical functions", *IEEE Transactions on Parallel and Distributed Systems*, **31**:6 (2019), pp. 1316–1327. ↑155

[68] C. Feldman, B. Michalowicz, A. Calder. *Lessons Learned. An In-Depth Look at Running FLASH on Ookami*, 30 pp. ↑156

[69] A. Ruhela, S. Xu, K. V. Manian, H. Subramoni, D. K. Panda. "Analyzing and understanding the impact of interconnect performance on HPC, Big Data, and deep learning applications: a case study with InfiniBand EDR and HDR", 2020 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW) (18–22 May 2020, New Orleans, LA, USA), pp. 869–878. ↑157

[70] C. 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 (2022), pp. 805–817. ↑157

[71] S. Kudo, K. Nitadori, T. Ina, T. Imamura. "Implementation and numerical techniques for one EFlop/s HPL-AI benchmark on Fugaku", 2020 IEEE/ACM 11th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems (ScalA) (13 Nov. 2020, Atlanta, GA, USA), pp. 69–76. ↑158, 182

[72] S. Kudo, K. Nitadori, T. Ina, T. Imamura. "Prompt report on Exa-scale HPL-AI benchmark", 2020 IEEE International Conference on Cluster Computing (CLUSTER) (14–17 Sept. 2020, Kobe, Japan), pp. 418–419. ↑158

[73] L. Eysymont, A. Frolov, A. Semenov. "Graph500: adequate rating", *Otkrytyye sistemy. SUBD*, 2011, no. 7, pp. 14–17 (in Russian). ↑158

[74] M. Nakao, K. Ueno, K. Fujisawa, Y. Kodama, M. Sato. "Performance evaluation of supercomputer Fugaku using breadth-first search benchmark in Graph500", 2020 IEEE International Conference on Cluster Computing (CLUSTER) (14–17 Sept. 2020, Kobe, Japan), pp. 408–409. doi ↑159, 169

[75] Y. Nakamura. *Software development and performance of Fugaku and ARM architectures*, The 38th International Symposium on Lattice Field Theory (26–30 July 2021), 2021, 9 pp. URL ↑159, 178

[76] K. I. Ishikawa, I. Kanamori, H. Matsufuru, I. Miyoshi, Y. Mukai, Y. Nakamura, K. Nitadori, M. Tsuji. *102 PFLOPS Lattice QCD quark solver on Fugaku*, 2021. arXiv 2109.10687 ↑159, 178

[77] H. Yashiro, T. Koji, K. Yuta, K. Shuhei, M. Takemasa, I. Toshiyuki, M. Kazuo, N. Masuo, K. Chihiro, S. Masaki, T. Hirofumi. "The NICAM 3.5 km-1024 ensemble simulation: Performance optimization and scalability of NICAM-LETKF on supercomputer Fugaku", vEGU21, the 23rd EGU General Assembly (online 19–30 April, 2021), EGU21-4771. doi URL ↑159

[78] J. Dongarra, S. Hammarling, N. J. Higham, S. D. Relton, P. Valero-Lara, M. Zounon. "The design and performance of batched BLAS on modern high-performance computing systems", *Procedia Computer Science*, **108** (2017), pp. 495–504. doi ↑162

[79] T. Shimizu. "Supercomputer Fugaku: Co-designed with application developers/researchers", 2020 IEEE Asian Solid-State Circuits Conference (A-SSCC) (9–11 Nov. 2020, Hiroshima, Japan), pp. 1–4. doi ↑162, 163

[80] T. Deakin, J. Price, M. Martineau, S. McIntosh-Smith. "GPU-STREAM v2.0: Benchmarking the achievable memory bandwidth of many-core processors across diverse parallel programming models", *ISC High Performance 2016: High Performance Computing*, International Conference on High Performance Computing, Lecture Notes in Computer Science, vol. **9945**, Springer, Cham, 2016, ISBN 978-3-319-46079-6, pp. 489–507. doi ↑163

[81] L. W. McVoy, C. Staelin. "lmbench: portable tools for performance analysis", ATEC '96: Proceedings of the 1996 annual conference on USENIX Annual Technical Conference (22–26 January, 1996, San Diego, CA, USA), 17 pp. URL ↑163

[82] N. Gupta, R. Ashiwal, B. Brank, S. K. Peddoju, D. Pleiter. "Performance evaluation of parallex execution model on arm-based platforms", 2020 IEEE International Conference on Cluster Computing (CLUSTER) (14–17 Sept. 2020, Kobe, Japan), pp. 567–575. doi ↑163

[83] T. Odajima, Y. Kodama, M. Tsuji, M. Matsuda, Y. Maruyama, M. Sato. "Preliminary performance evaluation of the Fujitsu A64FX using HPC applications", 2020 IEEE International Conference on Cluster Computing (CLUSTER) (14–17 Sept. 2020, Kobe, Japan), pp. 523–530. doi ↑164, 165, 167, 174, 180, 181

[84] H. Ltaief, J. Cranney, D. Gratadour, Y. Hong, L. Gatineau, D. E. Keyes. *Meeting the real-time challenges of ground-based telescopes using low-rank matrix computations*, 2021. URL ↑164, 182

[85] M. Kuz'minskiy. "Vector processes vs. accelerators", *Otkrytyye sistemy. SUBD*, 2018, no. 1, pp. 10–10 (in Russian). URL ↑165

[86] B. Brank, S. Nassyr, F. Pouyan, D. Pleiter. "Porting applications to Arm-based processors", 2020 IEEE International Conference on Cluster Computing (CLUSTER) (14–17 Sept. 2020, Kobe, Japan), pp. 559–566. doi ↑165

[87] C. Alappat, N. Meyer, J. Laukemann, T. Gruber, G. Hager, G. Wellein, T. Wettig. "Execution-Cache-Memory modeling and performance tuning of sparse matrix-vector multiplication and Lattice quantum chromodynamics on A64FX", *Concurrency and Computation: Practice and Experience*, e6512 (to appear). doi ↑165, 166, 176, 177

[88] C. Alappat, J. Laukemann, T. Gruber, G. Hager, G. Wellein, N. Meyer, T. Wettig. "Performance modeling of streaming kernels and sparse matrix-vector multiplication on A64FX", 2020 IEEE/ACM Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS) (12 Nov., 2020, Atlanta, GA, USA), pp. 1–7. doi ↑166

[89] A. Jackson, M. Weiland, N. Brown, A. Turner, M. Parsons. "Investigating applications on the A64FX", 2020 IEEE International Conference on Cluster Computing (CLUSTER) (14–17 Sept. 2020, Kobe, Japan), pp. 549–558. doi ↑166, 167, 173, 179, 180, 181, 182, 183

[90] F. H. McMahon. *Livermore Fortran kernels: A computer test of numerical performance range*, Technical Report UCRL-53745, Lawrence Livermore National Laboratory, 1986, 204 pp. ↑167

[91] P. R. Luszczek, D. H. Bailey, J. J. Dongarra, J. Kepner, R. F. Lucas, R. Rabenseifner, D. Takahashi. "The HPC Challenge (HPCC) benchmark suite", SC '06: International Conference for High Performance Computing, Networking, Storage and Analysis (11–17 Nov., 2006, Tampa, Florida, USA), 2006, pp. 213. doi ↑168

[92] H. Jin, M. Frumkin, J. Yan. *The OpenMP implementation of NAS parallel benchmarks and its performance*, NAS Technical Report NAS-99-011, 1999, 26 pp. URL ↑168

[93] L. Li, S. Pandey, T. Flynn, H. Liu, N. Wheeler, A. Hoisie. *SimNet: accurate and high-performance computer architecture simulation using machine learning*, 2021. arXiv 2105.05821 ↑168

[94] Y. Kodama, M. Kondo, M. Sato. "Evaluation of SPEC CPU and SPEC OMP on the A64FX", 2021 IEEE International Conference on Cluster Computing (CLUSTER) (7–10 Sept. 2021, Portland, OR, USA), pp. 553–561. doi ↑168

[95] M. Nakao, K. Ueno, K. Fujisawa, Y. Kodama, M. Sato. "Performance of the supercomputer Fugaku for breadth-first search in Graph500 benchmark", *ISC High Performance 2021: High Performance Computing*, International Conference on High performance Computing, Lecture Notes in Computer Science, vol. **12728**, Springer, Cham, 2021, ISBN 978-3-030-78713-4, pp. 372–390. doi ↑169

[96] M. Nakao. *Performance tuning of Graph500 benchmark on supercomputer Fugaku*, The first Meeting for Application Code Tuning on A64FX Computer Systems (9 December, 2020), 23 pp. ↑169

[97] B. Bramas. *A fast vectorized sorting implementation based on the ARM scalable vector extension (SVE)*, 2021. arXiv 2105.07782 ↑169

[98] A. Furuya, A. Asami. *Fujitsu regarding OSS porting for Tomitake/FX1000-RIST co-creation* (in Japanese), 12 pp. ↑171

[99] Y. Zempo, N. Akino, M. Ishida, E. Tomiyama, H. Yamamoto. "Real-time and real-space program tuned in K-computer", *Journal of Physics: Conference Series*, **640**:1 (2015), 012066. ↑174

[100] T. Nakajima, M. Katouda, M. Kamiya, Y. Nakatsuka. "NTChem: A high-performance software package for quantum molecular simulation", *International Journal of Quantum Chemistry*, **115**:5 (2015), pp. 349–359. ↑174

[101] M. Sato, H. Murai, M. Nakao, K. Tsugane, T. Odajima, J. Lee. "XcalableMP 2.0 and future directions", *XcalableMP PGAS Programming Language*, ed. M. Sato, Springer, Singapore, 2021, ISBN 978-981-15-7683-6, pp. 245–262. ↑174

[102] 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*, International Conference on High Performance Computing, Lecture Notes in Computer Science, vol. **12728**, Springer, Cham, 2021, ISBN 978-3-030-78713-4, pp. 332–350. ↑174

[103] K. Watanabe. *Performance tuning on LAMMPS for A64FX system*, The 5th Meeting for Application Code Tuning on A64FX Computer Systems (27 April, 2021), 2021, 27 pp. ↑175

[104] J. Huber, W. Wei, G. Georgakoudis, J. Doerfert, O. Hernandez. "A case study of LLVM-based analysis for optimizing SIMD code generation", *IWOMP 2021: OpenMP: Enabling Massive Node-Level Parallelism*, International Workshop on OpenMP, Lecture Notes in Computer Science, vol. **12870**, Springer, Cham, 2021, ISBN 978-3-030-85262-7, pp. 142–155. ↑176

[105] I. Kanamori, K. I. Ishikawa, H. Matsufuru. "Object-oriented implementation of algebraic multi-grid solver for lattice QCD on SIMD architectures and GPU clusters", *ICCSA 2021: Computational Science and Its Applications*, International Conference on Computational Science and Its Applications, Lecture Notes in Computer Science, vol. **12953**, Springer, Cham, 2021, ISBN 978-3-030-86976-2, pp. 218–233. ↑176, 178

[106] Y. Nakamura, Y. Mukai, K.-I. Ishikawa, I. Kanamori, Lattice quantum chromodynamics simulation library for Fugaku and computers with wide SIMD. ↑178

[107] S. Cielo, O. Porth, L. Iapichino, A. Karmakar, H. Olivares, C. Xia. *Optimizing the hybrid parallelization of BHAC*, 2021. arXiv 2108.12240 ↑178

[108] R. Bird, N. Tan, S. V. Luedtke, S.-L. Harrell, M. Taufer, B. Albright. *VPIC 2.0: next generation particle-in-cell simulations*, 2021. arXiv:2102.13133 ↑178, 179

[109] M. Klöwer, S. Hatfield, M. Croci, P. D. Düben, T. N. Palme. *Fluid simulations accelerated with 16 bit: Approaching 4x speedup on A64FX by squeezing ShallowWaters.jl into Float16*, ESSOAr, 2021, 26 pp. ↑180

[110] C. R. Ferenbaugh. "PENNANT: an unstructured mesh mini-app for advanced architecture research", *Concurrency and Computation: Practice and Experience*, **27**:17 (2015), pp. 4555–4572. ↑181

[111] A. Nukariya, K. Akao, J. Takahashi, N. Fukumoto, K. Kawakami, A. Kuroda, K. Minami, K. Sato, S. Matsuoka. *HPC and AI initiatives for supercomputer Fugaku and future prospects*, Fujitsu Technical Review, 6 pp. ↑183

[112] S. Rajpal, N. Lakhyani, A.-K. Singh, R. Kohli, N. Kumar. "Using handpicked features in conjunction with ResNet-50 for improved detection of COVID-19 from chest X-ray images", *Chaos, Solitons & Fractals*, **145** (2021), 110749. ↑183

[113] T. Honda. *Development of a deep neural network library for A64FX*, R-CCS 4th Meeting for Application Code Tuning on A64FX Computer Systems (17 March 2021), 25 pp. ↑183

Received 17.12.2021;
approved after reviewing 21.12.2021;
accepted for publication 22.02.2022.

Recommended by *prof. S. 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.

0000-0002-3944-8203
e-mail: kus@free.net