ISSN 2079-3316 **ПРОГРАММНЫЕ СИСТЕМЫ: ТЕОРИЯ И ПРИЛОЖЕНИЯ** **т. 12**, № 2(49), с. 73–103

ББК **32.972.11**
ГРНТИ **50.41.17**
УДК **004.451**

И. Е. Захаров, О. А. Панарин, С. Г. Рыкованов, Р. Р. Загидуллин, А. К. Малютин, Ю. Н. Шкандыбин, А. Е. Ермекова

Мониторинг приложений на кластере ZHORES в Сколтехе

Аннотация. Стандартные инструменты мониторинга для кластерных вычислительных систем позволяют оценить работу системы в целом, но не позволяют анализировать работу приложений по отдельности. Система мониторинга для измерения ресурсов, затребованных каждым приложением в отдельности разработана в Сколтехе для высокопроизводительного кластера ZHORES. Система мониторинга собирает как обычные метрики загрузки процессоров и графических ускорителей, так и счетчики событий ЦПУ/ГПУ, которые позволяют более детально анализировать тип ресурса, затребованный приложением. Сервисные программы, развернутые на каждом узле кластера, посылают результаты измерений в единую базу данных временных рядов с шагом в одну секунду. Эти данные затем анализируются статистическими методами в режиме оффлайн для выделения характеристик, связанных с использованием вычислительных ресурсов каждым приложением. Мониторинг позволяет выявлять неэффективное программное обеспечение, производить тонкую настройку работы кластера, а также улучшать работу высокопроизводительной системы в целом.

Ключевые слова и фразы: кластер, высокопроизводительные вычисления, мониторинг приложений, счетчики событий ЦПУ/ГПУ, база данных временных рядов.

Введение

Эффективное использование ресурсов суперкомпьютера — один из краеугольных камней высокопроизводительных вычислений (High Performance Computing — HPC). Зачастую приходится рассматривать суперкомпьютер как единое целое с точек зрения как пользователя, так и системного администратора. Так в работе [1] обсуждается важность и необходимость использования целостного системного мониторинга с

© И. Е. Захаров, О. А. Панарин, С. Г. Рыкованов, Р. Р. Загидуллин, А. К. Малютин, Ю. Н. Шкандыбин, А. Е. Ермекова, 2021
© Сколковский институт науки и технологий, 2021
© Программные системы: теория и приложения (дизайн), 2021
10.25209/2079-3316-2021-12-2-73-103

точки зрения разработки всесторонней модели поведения системы, в которой «система» обозначает как аппаратные, так и программные компоненты. В данной статье рассматривается реализация системы мониторинга на кластере ZHORES Сколтеха исходя из наших задач.

Высокопроизводительный вычислительный кластер ZHORES, развернутый в Сколковском институте науки и технологий (Сколтех), уже рассмотрен в работе [**2**]. Это установка петафлопсного уровня производительности[1], которая на момент написания статьи занимает 8-е место в списке ТОП-50 Российских суперкомпьютеров[2]. На кластере развернута система мониторинга Zabbix[3] и база данных, связанная с планировщиком Slurm [**3**], которые в совокупности позволяют создать полную картину по загрузке узлов и решают задачу планирования и управления пользовательской нагрузкой на кластере.

Решение установить именно Zabbix исходит из популярности этой системы мониторинга, наличия локальной экспертизы по использованию и настройке Zabbix, а также политики Сколтеха, предпочитающей инсталляцию систем с открытым кодом для поддержки собственных специалистов для полноценного обслуживания систем кластера.

Альтернативой является система мониторинга Ganglia [**4**]. Ganglia была изначально создана для высокопроизводительных систем, таких как ZHORES, и 90% HPC кластеров используют эту систему мониторинга[4]. Zabbix формально позиционируется как система для производственных кластеров[5], а не HPC кластеров. Несмотря на такое позиционирование системы мониторинга Zabbix, он более предпочтителен в связи с его постоянным обновлением.

Последняя версия обновления Zabbix была сделана 22 марта 2021 года, в то время как Ganglia 3.7.2 выпустила обновление 14 июня 2016 года и больше не менялась. Скорость обновлений важна для понимания возможности поддержки кластера, в том числе с точки зрения совместимости с необходимыми версиями операционной системы. На кластере ZHORES установлена Zabbix версия 5.2.0, которая была выпущена в октябре 2020 года, ядро Linux 5.4.15, выпуск которого был в январе 2020, и CentOS 7, полностью обновленный в августе 2020 года. Другой популярной альтернативой является система мониторинга

[1]Установка имеет около 100 узлов с процессорами фирмы Intel (ЦПУ) и около 25 узлов смешанного типа используют графические ускорители фирмы Nvidia (ГПУ).

[2]`http://top50.supercomputers.ru`.

[3]`http://www.zabbix.com`.

[4]Статистика 2018 года, цитируется по работе [**1**].

[5]Enterprise cluster, см., например, `https://www.zabbix.com`.

Nagios[6]. Выбор между Zabbix и Nagios труднее обосновать, здесь играют роль наши предпочтения, хотя есть и объективные критерии[7].

Другая важная задача состоит в получении информации о работе приложений на кластере. Этой задаче посвящена настоящая статья. Информация по приложениям может помочь оптимизировать их работу или подсказать решения по управлению запуском приложений. Недостаточно знать, что кластер загружен заданиями и нет сбоев в системе, но также необходимо изучить природу нагрузки на функциональные единицы узлов, выявить узкие места и устранить их для увеличения пропускной способности кластера. Исследования показывают, что объединение самых различных измерений необходимо для полного анализа производительности[8]. Стандартный мониторинг такими системами как Zabbix или любыми другими инструментами администратора[9] собирает метрики по загрузке процессоров и узлов кластера, но не позволяет разделить эти метрики по приложениям, которые создают нагрузку.

Анализ использования ресурсов приложениями осуществляются профилировщиками, такими как perf/perfmon, Scalasca[10], Speedshop[11] и методами, собранными в библиотеках типа PAPI[12], инструментами типа Intel Vtune и PCM[13]. Эти инструменты позволяют детально проанализировать работу приложений и воссоздать полную картину загрузки ресурса отдельным приложением. Мы здесь не обсуждаем многочисленные инструменты, предназначенные для работы со специальными приложениями, например, позволяющие оптимизировать работу интернет-серверов и другие виды приложений[14].

---

[6] www.nagios.org

[7] Tim Keary, Nagios vs Zabbix, Augus 28, 2020, https://www.compar-itech.com/net-admin/nagios-vs-zabbix/.

[8] HOPSA – the joint RF/EU project: Holistic Performance System Analysis, http://hpc.msu.ru/node/84

[9] M. Hasan, Most Comprehensive List of Linux Monitoring Tools For SysAdmin www.ubuntupit.com/most-comprehensive-list-of-linux-monitoring-tools-for-sysadmin (accessed: 20/10/2020)

[10] https://www.scalasca.org/.

[11] https://openspeedshop.org/.

[12] https://icl.utk.edu/papi/.

[13] T. Willhalm, R. Dementiev, Intel® performance counter monitor — a better way to measure CPU utilization, Published:08/16/2012, Last Updated:01/05/2017, https://software.intel.com/en-us/articles/intel-performance-counter-monitor.

Более новая ветка инструментов для программирования: Processor Counter Monitor, https://github.com/opcm/pc.

[14] https://www.softwaretestinghelp.com/top-10-application-per-formance-monitoring-tools/.

На кластере ZHORES нас в первую очередь интересуют приложения, которые относятся к моделированию физических процессов и машинному обучению [2]. Такие приложения характеризуются высокой интенсивностью использования процессора и каналов памяти, а также использованием графических ускорителей с высокой интенсивностью счета. При параллельных расчетах может использоваться библиотека передачи сообщений, работающая через сеть Infiniband и глобальная файловая система, с которой загружаются данные для машинного обучения. Для оценки работы таких приложений мы заинтересованы в измерениях получаемых с помощью аппаратных счетчиков встроенных в структуру процессоров, графических ускорителей и сетевых интерфейсов высокопроизводительной сети. Таким образом возможен непосредственный учет аппаратных ресурсов, задействованных предлагаемой нагрузкой. Эти данные собираются вышеупомянутыми инструментами анализа производительности и, в принципе, могут быть скоррелированы с данными мониторинга загрузки аппаратных ресурсов собираемые операционной системой для более полного анализа. Однако инструменты индивидуального анализа, упомянутые выше, обладают высокой трудоемкостью и не позволяют автоматизировать статистический анализ приложений на кластере.

Задача автоматизации мониторинга приложений и объединения с мониторингом функциональных узлов кластера решалась и ранее [5–9]. Важно, что мониторинг приложений на кластере ZHORES должен работать с заранее неизвестной нагрузкой задачами, подготовленными пользователями на своих условиях. Этим он отличается от мониторинга специализированных приложений (интернет-сервер и тому подобные приложения), для которых есть такие инструменты как Elastic APM[15]. Успех таких инструментов связан с тем, что приложения заранее подготавливаются к использованию с монитором их активности, что возможно сделать только для выделенных приложений или специфичных фреймворков.

Расширению функционала Zabbix для автоматизации мониторинга приложений на кластере ZHORES препятствует фундаментальная несовместимость требований к сбору и хранению данных. Zabbix не рассчитан на очень короткий интервал сбора данных, обычно интервал составляет от одной до нескольких минут. Соответственно, база данных MYSQL, которую использует Zabbix, не рассчитана на большую скорость приема данных. Для мониторинга счетных

[15] `https://www.elastic.co/apm`.

приложений на кластере ZHORES характерны времена сбора данных с частотой в 1 Гц и необходимость доступа к аппаратным счетчикам событий с их динамичным перепрограммированием.

Возвращаясь к неполному списку проектов по автоматизации мониторинга приложений [5–9], упомянем, что они все написаны для определенного кластера исходя из невозможности адаптации предыдущих таких проектов и являются собственностью своих владельцев. Нет стандартного или наиболее популярного пакета автоматического мониторинга приложений для Linux кластеров. Следовательно, вопрос о поддержке и оперативном обновлении играет основную роль для принятия решения об установке стороннего пакета или разработки своей системы. Для кластера ZHORES мониторинг приложений является проектом исследовательским, поэтому полный доступ к исходному коду и возможность гибкой адаптации очень важны. При этом используется опыт предыдущих проектов мониторинга.

Мы сохраняем схему решения, при которой на каждом из узлов работает сервисный процесс, собирающий весь набор информации о работе узла и программ пользователя на узле. Стандартным требованием является низкая загрузка ЦПУ (≈2–3%) сервисным процессом. Вся информация посылается в центральную базу данных, которая должна выдерживать поток данных реального времени со всех серверов. В такой простой схеме именно производительность базы данных является критическим фактором.

В работе [9] описана система DiMMon мониторинга, которая разработана для МГУ и рассмотрена возможность уменьшения потока данных и, соответственно, уменьшения давления на центральную базу путем разделения потоков к нескольким базам данных и использования сервисных процессов сбора данных для операций над измеренными значениями на вычислительных узлах без передачи в центральную базу данных. Для относительно небольшого кластера как ZHORES (около 100 узлов) такая оптимизация не имеет большого смысла, так как выбранная база данных Akumuli[16] справляется с предложенным потоком. Многие сложности реализации, обсуждаемые в других проектах [5–9], отпадают в нашем исследовательском проекте, не нуждающемся в отказоустойчивости мониторинга приложений.

База данных временных рядов Akumuli уже используется в ряде отечественных систем мониторинга, например Botikmon3, где она интегрирована в распределенное NoSQL хранилище [10]. Уникальным вкладом нашей работы являются производимые измерения

[16] `https://akumuli.org/`.

и последующий анализ временных рядов с фокусом внимания на приложения.

В нашей системе мониторинга собирается как можно большее количество различных метрик, включающее в себя аппаратные счетчики ЦПУ и ГПУ. На кластере ZHORES используются вычислительные блоки Intel и Nvidia, поэтому мы уделяем внимание решениям именно этих производителей.

Использование аппаратных счетчиков является популярной темой обсуждения для ЦПУ. Фирма Intel предоставила[17] соответствующие расширения архитектуры IA64/32, на основе которой строятся библиотеки доступа. Для ГПУ аппаратные измерения температуры, потребляемой мощности и другие подобные виды счетчиков известны давно и являются частью многих систем мониторинга [**11**]. Однако только недавно (в 2020 году) фирма Nvidia сделала доступными микро-архитектурные счетчики, позволяющие оценить как приложения нагружают функциональные блоки внутри ГПУ [**14**]. По нашим данным, отслеживание микроархитектурных счетчиков ГПУ и их анализ в совокупности со счетчиками ЦПУ и операционной системы не рассматривалось в предыдущих работах по автоматическому мониторингу приложений. Именно собранные метрики и последующий статистический анализ собранных метрик составляют уникальный вклад этой работы.

В параграфе 1 мы рассматриваем разработанную систему мониторинга более подробно и описываем собираемые метрики с измерением работы самого сервисного процесса сбора информации. В параграфе 2 рассматривается статистический анализ данных, отнесенный к узлам кластера, затем в параграфе 3 обсуждается статистический анализ данных по приложениям. Для визуальной оценки получаемых результатов используются в основном диаграммы размаха[18]. Заключение содержит сделанные выводы и планируемые дополнительные исследования.

1. Работа системы мониторинга

Система мониторинга приложений создана авторами на основе описанной ранее системы для мобильных комплексов PMON [**12**]. В

[17] Intel 64 and IA32 Architectures Performance Monitoring Events, 2017 Rev. 1.0; Intel 64 and IA—32 Architectures Software Developer's Manual, Vol. 3B System Programming Guide, Part 2. `https://www.intel.ru/content/www/ru/ru/architecture-and-technology/64-ia-32-architectures-software-developer-vol-3b-part-2-manual.html`.

[18] `https://datavizcatalogue.com/RU/metody/diagramma_razma-ha.html`

частности, используется база данных Akumuli[19] временных рядов (TSDB). В отличие от мобильных систем, где необходима множественная избыточность, TSDB Akumuli развернута только на одном узле и все сервисные программы, собирающие данные с узлов, посылают их в единую базу данных, где собираются временные ряды со всеми метриками, помеченными именем узла и именем программ, запущенных на соответствующем узле. Эти метки затем используются для выделения связанных данных. С каждого сервера собираются все обычные метрики, относящиеся к работе системы, в том числе времена загрузки процессора (user/system) и загрузка оперативной памяти. Для мониторинга приложений дополнительно к системным метрикам собираются данные по активным программам.

1.1. Метрики, собираемые с приложений

На каждом узле с запуском операционной системы поднимается сервисный процесс, который собирает все метрики с узла и метрики активных процессов на узле. Для этого каждую секунду опрашивается таблица процессов (/proc) и собираются специфичные данные, относящиеся к процессам в системе. Некоторые из собираемых параметров собраны в таблице 1. Здесь загрузка (`occupancy`) рассчитывается по формуле, указанной в комментарии таблицы 1 по аналогии с оценкой холостого хода процессора. Из этого расчета также оценивается, сколько ядер использует приложение (метрика `xtask`), так как совокупное время счета (`selftime` из /proc) может быть больше времени нахождения процесса в системе (`lifetime`), если используются несколько ЦПУ для счета.

Кроме этого, считываются программируемые счетчики ЦПУ, позволяющие оценить работу самих узлов процессора. Доступ к этим счетчикам осуществляется через библиотеку LIKWID [**13**], которая позволяет считать события в процессоре. Некоторые из этих метрик собраны в таблице 2. Счетчики событий в процессорах делятся на две категории: есть фиксированные счетчики (для циклов) и счетчики, которые можно программировать для учета определенных событий. Программируемые счетчики не могут считать все события сразу. В системе мониторинга PMON счетчики перепрограммируются на разные события каждую секунду и цикл завершается после 6 секунд. Таким образом, можно получить среднюю картину использования ресурсов в процессоре в предположении, что программа достаточное время

[19] `https://akumuli.org/`

Таблица 1. Метрики процессов, доступные в операционной системе

Метрика	Комментарий
`occupancy`	$= \frac{\texttt{selftime}}{\texttt{lifetime} \times \texttt{xtask}}$, где $\texttt{selftime} = \texttt{user} + \texttt{system}$, $\texttt{lifetime} = \texttt{timestamp} - \texttt{starttime}$ (утилизация ЦПУ программой: $0 \leqslant \texttt{occupancy} \leqslant 100\%$)
`xtask`	$= ceil(\texttt{occupancy})$ (количество используемых ядер процессоров)
`rss`	размер данных в оперативной памяти [МБ]
`runtime`	время работы программы [с]
`Prog`	название программы
`pid`, `uid`, `pgid`	идентификаторы процесса, пользователя и группы
`cpu`	идентификатор ЦПУ, на котором работает процесс

Таблица 2. Метрики процессов, доступные со счетчиков в ядрах ЦПУ

`CPI`	кол. проц. циклов на инструкцию (cycle per instruction)
`L3_load`	скорость загрузки данных L3 [МБ/с]
`L3_mem`	скорость загрузки данных L3 в оперативную память [МБ/с]
`Cycles_used`	доля циклов, выполняющая инструкции программы [%]
`Cycles_stall`	доля циклов, работающая вхолостую (все причины) [%]
`Cycles_stall_MEM`	доля циклов, работающая вхолостую из-за задержки памяти [%]
`Freq`	Частота работы процессора (ядра) [МГц]

(>10 секунд) однородно использует ресурсы. Особенность работы с библиотекой LIKWID заключается в том, что счетчики учитывают события в ядрах процессора, никак не привязанных к приложению. Чтобы зафиксировать эту связь, программа мониторинга использует идентификатор вычислительного ядра, на котором исполняется приложение[20], чтобы привязать значения счетчиков именно этого ядра к метрикам процесса. Здесь делается допущение, что приложение не

[20]информация из /proc/PID/stat поле 39 processor — CPU number last executed on, см. документацию man proc.

мигрирует с ядра на ядро за интервал счета (1 секунда), что характерно для счетных и долго работающих приложений.

Подобным образом считываются параметры нагрузки и счетчики ГПУ ускорителей. На кластере ZHORES используются ГПУ Nvidia V100. Для снятия данных по текущим параметрам работы (потребляемая мощность, температура, частота работы и загрузка графического конвейера и памяти) используется библиотека NVML, а чтение микро-архитектурных счетчиков событий в ГПУ поддерживаются библиотекой DCGM [14]. С помощью этих инструментов собираются метрики, обозначенные в таблицах 3 и 4. По микроархитектурным

Таблица 3. Метрики ГПУ, получаемые пакетом NVML

`Power`	потребляемая электрическая мощность [Вт]
`Temp`	температура устройства [°C]
`Clock_G`	рабочая частота графического конвейера [МГц]
`Clock_M`	рабочая частота графической памяти [МГц]
`Util_G`	утилизация графического конвейера [%]
`Util_M`	утилизация графической памяти [%] (от 16 ГБ V100)

счетчикам ГПУ собираются следующие метрики: Также, в случае с

Таблица 4. Параметры работы сервисного процесса мониторинга

`SM_Active`	доля циклов ядро CUDA использовалось в вычислениях [%]
`SM_Occup`	доля активных вычислительных устройств [%]
`GR_Active`	доля времени использования графического конвейера [%]
`DRAM_Active`	доля циклов интерфейс памяти устройства перемещал данные [%]
`tensor`	доля тензорных операций [%]
`fp16`	доля операций с точностью 16 бит [%]
`fp32`	доля операций с точностью 32 бит [%]
`fp64`	доля операций с точностью 64 бит [%]
`PCIE_RX/TX`	скорость приема/передачи данных по шине PCIe [МБ/с]
`NVLI_RX/TX`	скорость приема/передачи данных по шине NVLink [МБ/с]

метриками ГПУ, данные не привязаны к конкретным процессам, а собираются на фоне работающих приложений и привязываются к процессам в предположении, что планировщик распределяет задания по одному приложению на узел. Поэтому можно связать ГПУ данные с приложением по временной метке.

1.2. Оценка работы программы мониторинга

Система статистической оценки работы приложения была применена к самой программе мониторинга. Из интересных метрик можно выделить параметр `occupation` загрузки ЦПУ и другие параметры, собранные в таблице 5. В среднем процесс мониторинга занимает

Таблица 5. Параметры работы сервисного процесса мониторинга

	mean	std	min	50%	75%	95%	max
occupation [%]	8.45	1.54	6.21	8.34	9.48	10.65	23.12
CPI	2.28	8.51	0.00	0.97	1.63	7.81	856.87
Cycles_stall [%]	30.59	33.54	0.00	0.00	62.14	88.33	99.59
L3_MEM [МБ/с]	17.78	220.05	0.00	0.60	1.22	33.49	25425.40
Cycles_stall_MEM	25.45	15.12	0.00	22.24	33.95	53.12	96.56

$8.45 \pm 1.54\%$ времени одного ядра ЦПУ (mean), в редких случаях может доходить до 23% (max, <5% случаев). В среднем требуется 2.28 циклов на одну инструкцию при лучших показателях ЦПУ Intel в 0.5 циклов на инструкцию для счетных программ. Это связано с тем, что программа мониторинга в основном читает и переформатирует информацию, и в этом процессе много циклов ожидания. Это выражается в среднем значении в 30.6% холостых циклов, в основном по доступу к памяти (25.45%). Загрузка шины памяти `L3_MEM` минимальна и не мешает работе приложений.

Загрузка ЦПУ процессом мониторинга может и должна быть уменьшена путем оптимизации программы, эта работа планируется на ближайшее время.

1.3. База данных мониторинга

На кластере ZHORES каждый из 100 узлов посылает каждую секунду пакеты, содержащие цифровые значения и метаданные 127 — и метрик общим объемом в 22 КБ на единый узел с развернутой базой TSDB Akumuli[21], по административной сети в формате Ethernet 10G. Таким образом, нагрузка на эту сеть составляет 1.8 МБ/с. Полный объем данных за день составляет 155 ГБ, но часть из них относится к протоколу redis[22], который естественно отбрасывается после обработки.

[21] Используется версия 0.8.80, актуальная на конец 2020 года.
[22] `https://redis.io/topics/protocol`.

Затем база данных эффективно сжимает информацию. Сама база данных развернута с фиксированным объемом в 128 ГБ. Такого объема хватает на 10 дней сбора данных, после чего самые старые данные автоматически замещаются более новыми данными мониторинга. Сервер базы TSDB Akumuli развернут в виртуальной машине с 4 — мя процессорами и 160 ГБ оперативной памяти. В обычном режиме при загрузке данных в базу, нагрузка на процессор составляет меньше 10 процентов и используется половина оперативной памяти. Однако при опросе собранных данных, два ЦПУ загружены полностью, и используется вся оперативная память. Количество процессоров, используемых для выдачи данных, регулируется так, чтобы оставались ресурсы для продолжения загрузки данных, посылаемых с узлов кластера в штатном режиме. Это стандартные настройки Akumuli.

Опрос производится по протоколу HTTP (POST) на выделенный порт 8181 в формате сбора данных JSON[23]. Движок базы данных Akumuli имеет ограничения на количество одновременно выдаваемых данных, поэтому приходится запускать запрос на выдачу данных несколько раз на один и тот же временной интервал. Данные получаются в текстовом формате.

Так как в базе данных Akumuli самые старые данные замещаются более новыми данными с периодом 10 дней, мы используем получающиеся текстовые файлы для длительного хранения в глобальной файловой системе кластера ZHORES. Для реализации выбрана наиболее простая система хранения долгосрочной информации в текстовых файлах, индексированных по дате, хотя известны успешные реализации долгосрочного хранения с базой данных [9]. В нашем случае дополнительное исследование должно показать для каких данных долгосрочное хранение имеет смысл и возможна ли реализация с добавлением выбранных данных к другим базам, уже поддерживаемым на кластере ZHORES (Zabbix — MySQL и Slurm — Elastic). На этом этапе главное направление исследований — это статистический анализ данных. Статистический анализ данных из этих файлов рассматривается в разделах 2 и 3.

1.4. Визуальный мониторинг приложений

Стандартная система графического отображения Grafana, подключенная к TSDB Akumuli, позволяет визуально отследить работу приложений в реальном времени. Пример показан на рисунке 1.

[23] `https://json.org/json-en.html`.

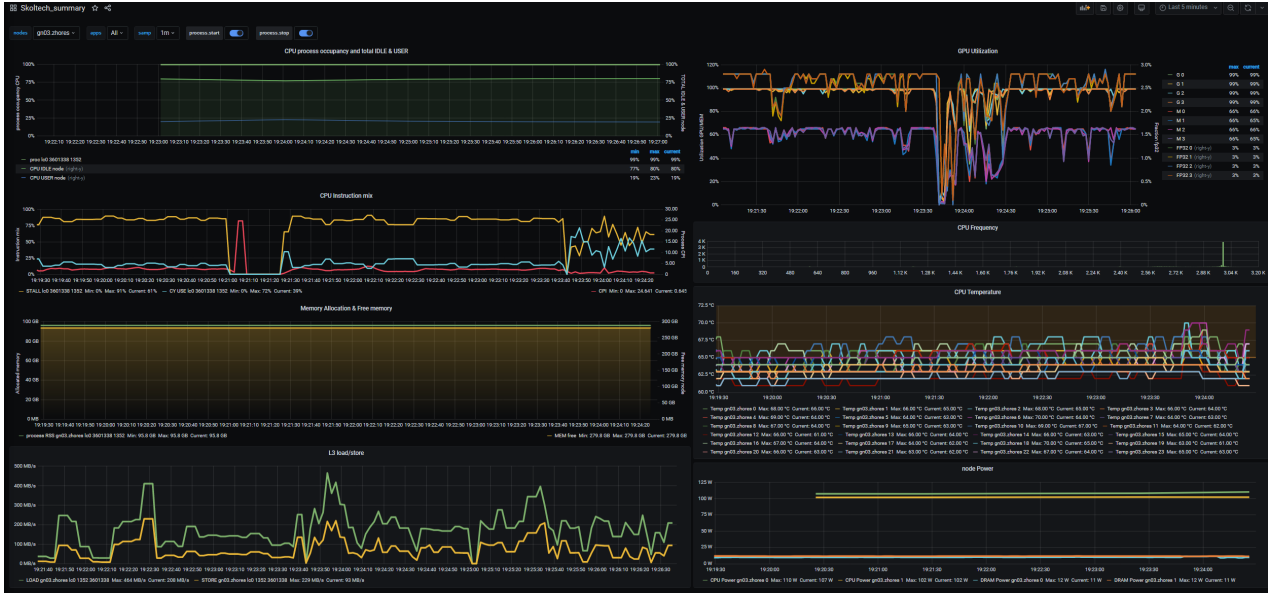

Рисунок 1. Пример представления информации в окне Grafana

Интерфейс Grafana содержит интерактивное меню, предлагающее выбрать приложение из выпадающего списка и имя узла, с которого берутся данные. По оси абсцисс во всех графиках выложено время регистрации событий. В левой колонке рисунка 1 показаны некоторые метрики, имеющие отношение к приложению, в том числе CPI, загрузка памяти и скорость работы с памятью; в правой колонке отображены системные параметры узла, такие как распределение частоты работы ЦПУ, температуры ЦПУ ядер, электрическое потребление процессоров и памяти узла.

Верхний правый график показывает работу графического ускорителя: загрузку ГПУ карт по вычислительному конвейеру и по памяти (левая ордината [%]); правая ордината показывает процент инструкций вычислений с ординарной точностью.

2. Статистический анализ данных по узлам кластера

Данные, выдаваемые TSDB Akumuli, сбрасываются в текстовые файлы и фильтруются на корректность и целостность. Затем отфильтрованные данные поступают в программу для статистического анализа, написанную на языке Python3 и использующую библиотеки Pandas[24]. Файлы данных загружаются вызовом read_csv() с указанием ожидаемого формата, точно соответствующим представлению файлов на диске. При этом индекс массива конвертируется из временной метки операционной системы (Linux Epoch time) вызовом to_datetime (...,origin="unix"..). Несколько массивов данных, соответствующих разным файлам на диске, сводятся в один массив вызовом merge (...,on=["timestamp«, »prog«, »host "uid", ...,], how="inner") с указанием точного соответствия (inner) общих колонок данных (on=[...]). Размер данных за 1 день составляет примерно 1 Гб. При этом при объединении за период больше месяца, с 4 апреля по 10 мая, размер занимаемой памяти составил 2.5 Гб. Анализ данных с помощью диаграмм размаха[25] показывает загрузку графического конвейера (см. рисунок 2) и памяти ГПУ (см. рисунок 3) на всех гибридных узлах кластера ZHORES за период с 5 апреля по 10 мая 2021 года.

По горизонтали в этих рисунках — название узла; по вертикали — загрузка в [%]. Диаграмма размаха передает размер статистического распределения измеряемой величины в прямоугольнике, где горизонтальная линия обозначает медиану[26], а размер прямоугольника — верхний и нижний квартиль (75% и 25% соответственно). Отдельные значения, не укладывающиеся в прямоугольник, передаются как черные кружки вне прямоугольников.

---

[24]The pandas development team, https://doi.org/10.5281/zenodo.3509134.

[25]https://datavizcatalogue.com/RU/metody/diagramma_razma-ha.html

[26]Медиана — это число в середине упорядоченного набора чисел: половина данных находится ниже этого значения, а половина выше.

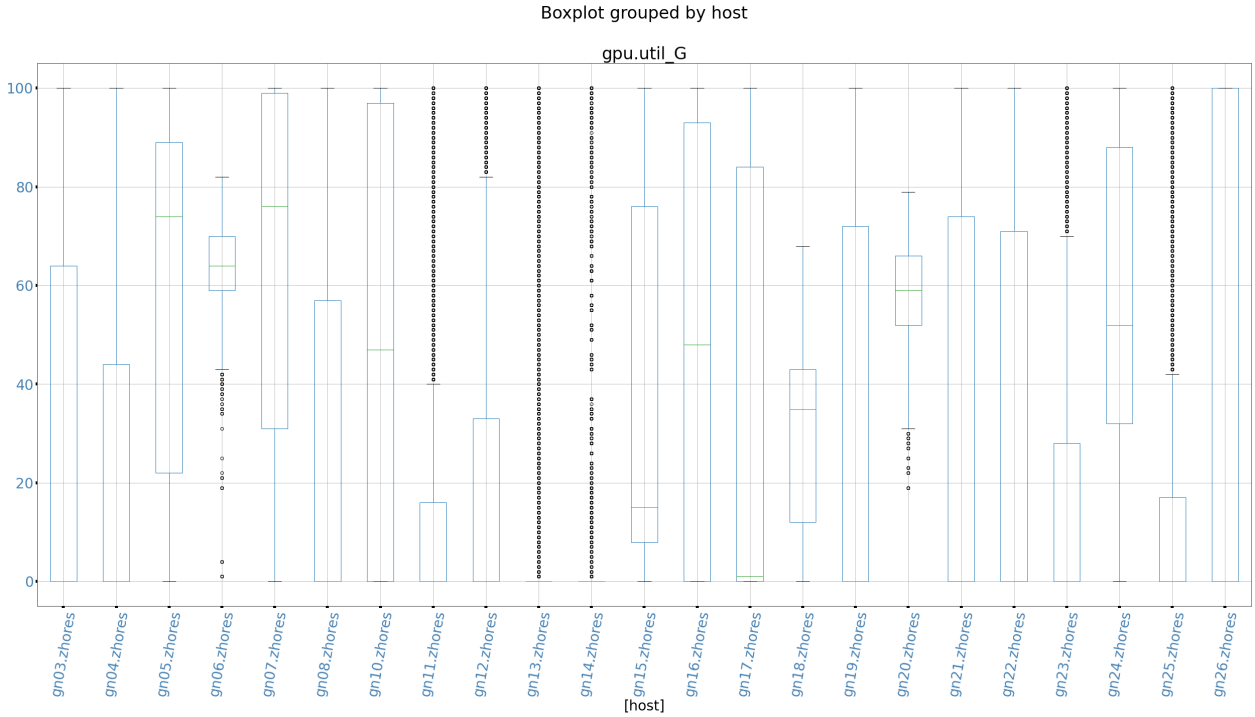

Рисунок 2. Загрузка конвейера ГПУ на гибридных узлах кластера ZHORES

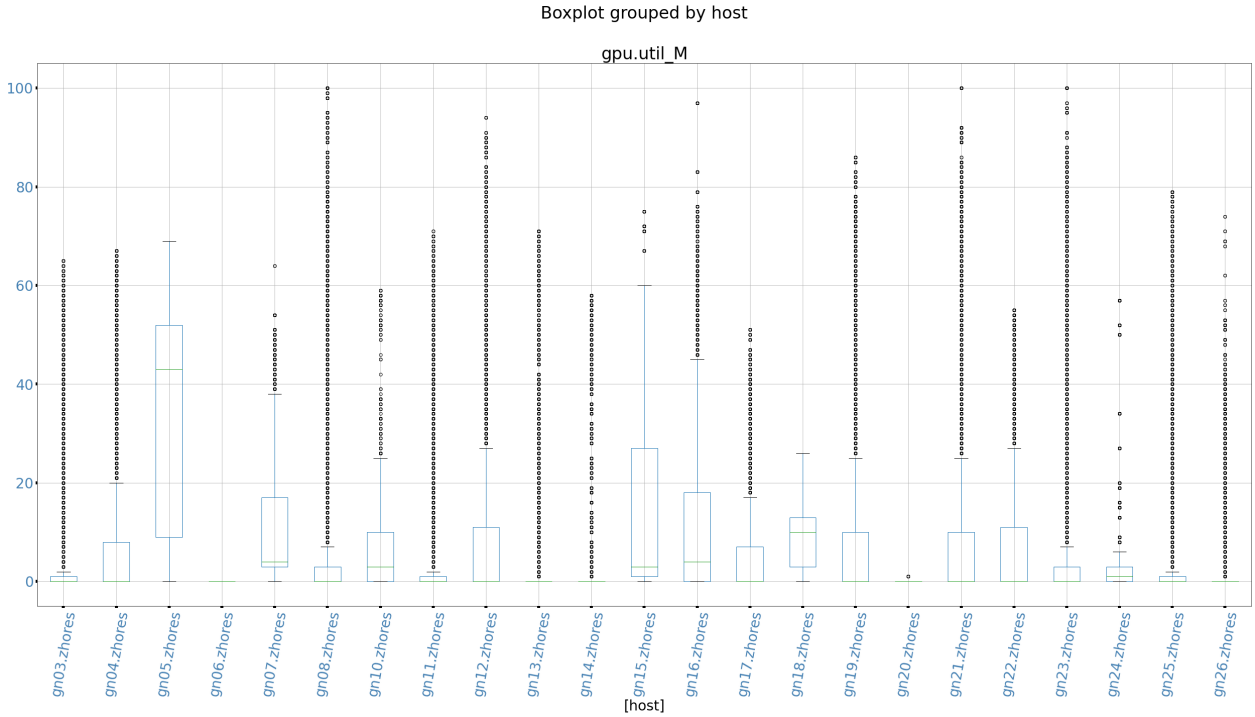

Рисунок 3. Загрузка памяти ГПУ на гибридных узлах кластера ZHORES

3. Результат статистического анализа приложений

Вычислительные приложения используют ЦПУ/ГПУ длительное время и нагружают ресурсы. Поэтому были выделены приложения на гибридных узлах кластера с нагрузкой на ЦПУ (occup) >50%, занятостью ГПУ ускорителя (utilG)> 50% и временем работы больше 10 минут. В таблице 6 приведены значения загрузки памяти ЦПУ (rss) и ГПУ (utilM [%] от 16 ГБ памяти Nvidia V100). В таблице

Таблица 6. Метрики вычислительных приложений, взятых для анализа

	ЦПУ			ГПУ		
	L3_MEM, Max [ГБ/с]	RSS, Max [ГБ]	Cy_stall, Mean [%]	PCI/RX, Max [ГБ/с]	PCI/TX, Max [ГБ/с]	UtilM, Max [%]
Rotamers	2.58	3.95	6.9	0.26	0.052	50.0
gmx_mpi	7.10	46.84	2.8	3.03	0.91	100.0
lazycon	10.88	82.26	27.4	2.80	0.86	82.0
lc0	0.28	357.58	10.3	0.08	0.14	69.0
lmp_md	2,22	0.25	8.7	0.48	0.07	1.0
python	5.95	64.98	22.5	9.50	4.36	100.0
python3	6.08	15.80	38.0	5.04	0.70	100.0
vasp_gpu	5.86	68.56	20.9	6.90	4.36	100.0

приведены средние значения холостых циклов Cy_stall и максимальные значения для загрузки памяти ЦПУ/ГПУ этими приложениями, а также максимальные значения передачи данных в память ЦПУ (L3_MEM) и в память ГПУ через шину PCIe RX/TX. Следует отметить эффективность вычислительных задач: Cy_stall <50% значит, что эффективность ЦПУ (100%−Cy_stall) больше 50%. В этом ряду выделяется программа lc0 (из матобеспечения leela) работающая с эффективностью 90% на ЦПУ и в среднем 70% на ГПУ (это значение не представлено в таблице 6). Такая эффективность коррелирует с очень низкими измеренными потребностями по скорости передачи данных как в память ЦПУ (значение L3_MEM), так и по шине PCIe в память ГПУ. Для программ с не слишком большой зависимостью от входных данных (такие как GROMACS:gmx_mpi[27] и VASP:vasp_gpu[28]),

[27]GROMACS — пакет программ для молекулярного моделирования.
[28]VASP — пакет программ для моделирования материалов.

интерпретация данных не представляет трудностей[29]. Трудность представляет интерпретация нагрузки для программ на языке Python, так как нагрузка определяется программой пользователя, указанной в аргументах запуска интерпретатора языка Python, которая не анализируется сервисным процессом из соображений экономии процессорного времени на собственно мониторинг. Поэтому в дальнейшем анализе программы обозначены дополнительно идентификатором пользователя из предположения, что каждый пользователь имеет стабильный набор скриптов на языке Python.

Анализ загрузки памяти ГПУ представлен на рисунке 4, где проведено разделение по идентификатору пользователя (uid). Доля использования операций с одинарной точностью представлена на рисунке 5. Доля программ, использующих двойную точность, на ГПУ невелика (см. рисунок 6), и таких операций меньше 10%. Подобным образом обстоит дело с тензорными операциями, которые впервые стали поддерживаться в ГПУ ускорителе V100. Это показано на рисунке 7. Внедрение тензорных операций требует переписывания исходного кода, и мы полагаем, пользовательские приложения еще не используют эту возможность для увеличения производительности. Вероятно, выявленные тензорные операции используются в библиотеках Nvidia, которые подгружаются приложениями.

Подобная картина наблюдается и для операций с 16-ти битными числами (не показаны на диаграммах). Только программа lc0 (leela) использует до 20% операций fp16 в вычислениях. Из полученных данных можно сделать вывод, что за рассмотренный период времени подавляющее большинство операций на ГПУ — это операции с одинарной точностью. Но это действительно зависит от рассмотренного интервала.

3.1. Использование шины PCIe

Через шины PCIe данные закачиваются в память ГПУ (метрика PCI_RX) и результаты вычислений копируются в память ЦПУ (метрика PCI_TX). На рисунках 8 и 9 показано распределение загрузки этого ресурса по приложениям.

Из анализа видно, что в среднем программы потребляют 20% от пропускной способности шины PCIe, хотя отдельные загрузки могут доходить до 100% потребления (черные кружки).

[29]Сами пакеты могут иметь большой разброс параметров, но в практическом применении пользователи осуществляют множественный запуск для изучения похожих явлений. Поэтому можно считать нагрузку на процессор достаточно стабильной.

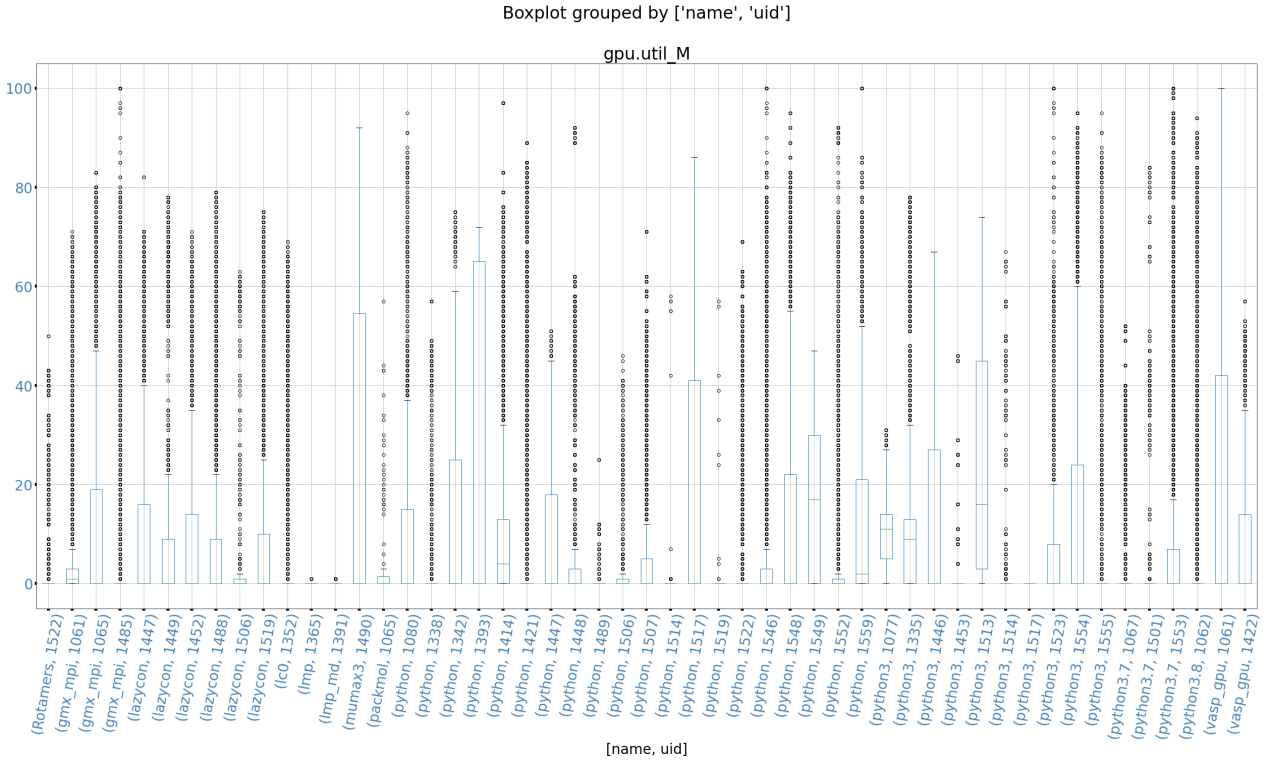

Рисунок 4. Загрузка памяти ГПУ по приложениям в [%] от 16 ГБ (V100)

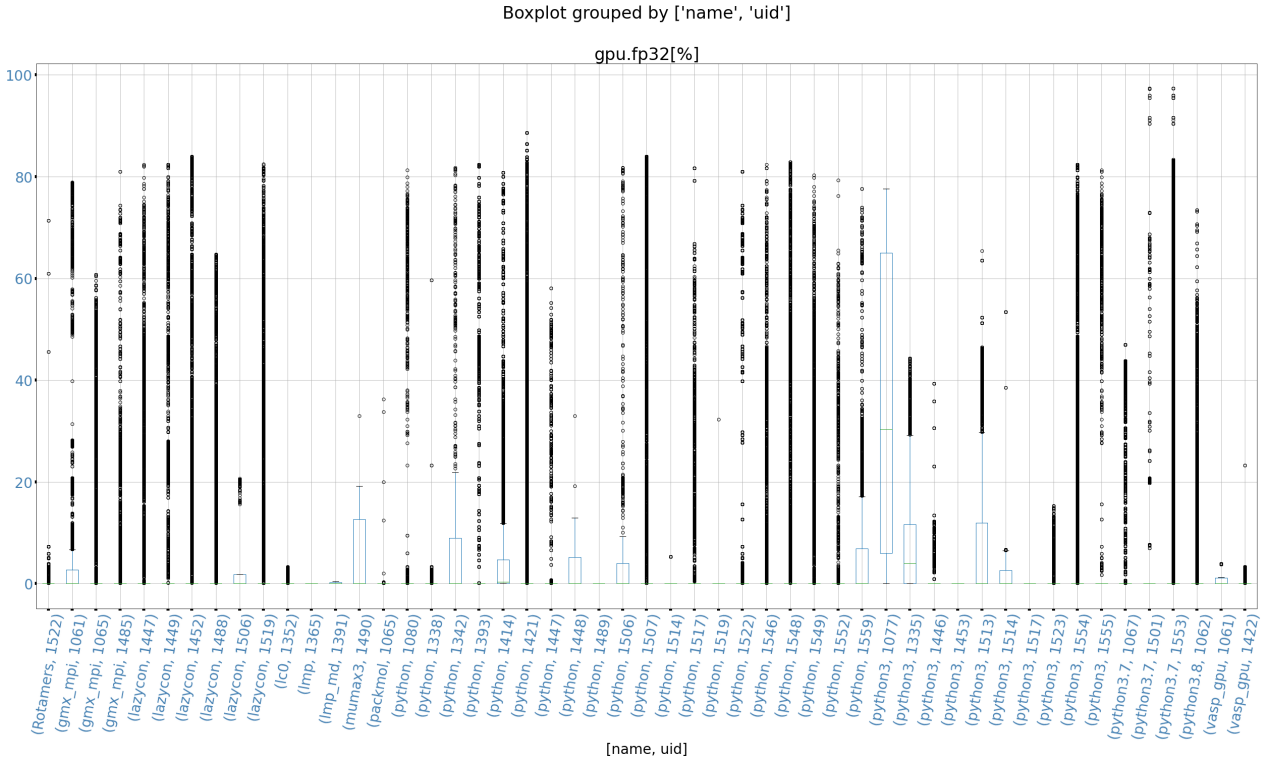

Рисунок 5. Доля использования операций с одинарной точностью (fp32)

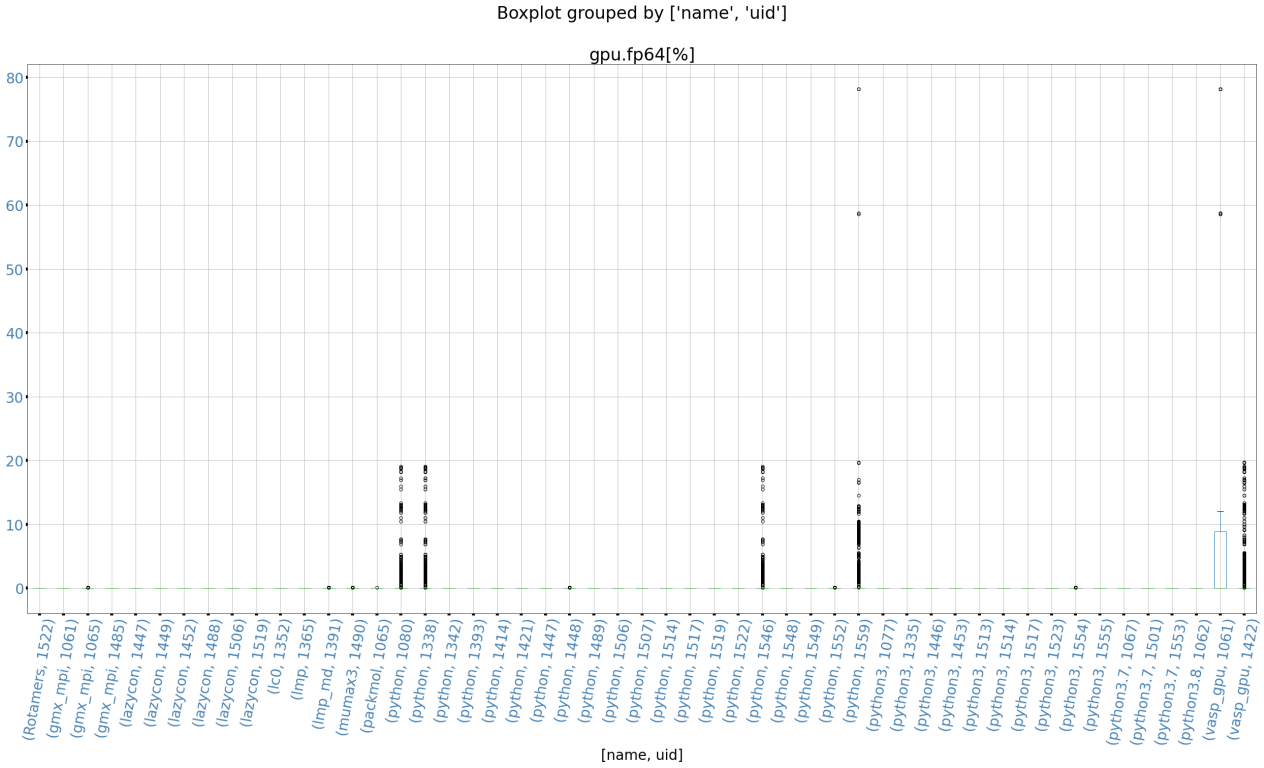

Рисунок 6. Доля использования операций с двойной точностью (fp64)

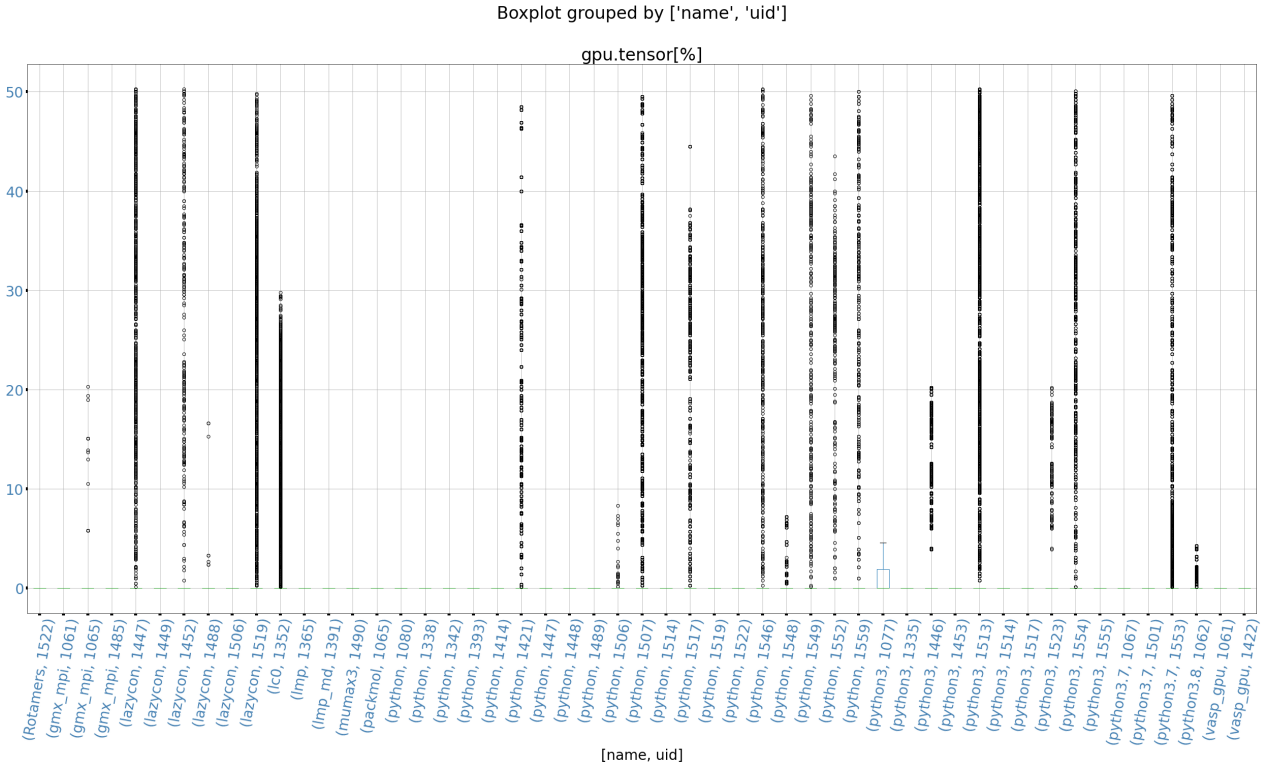

Рисунок 7. Процент тензорных операций в приложениях на ГПУ NvidiaV100

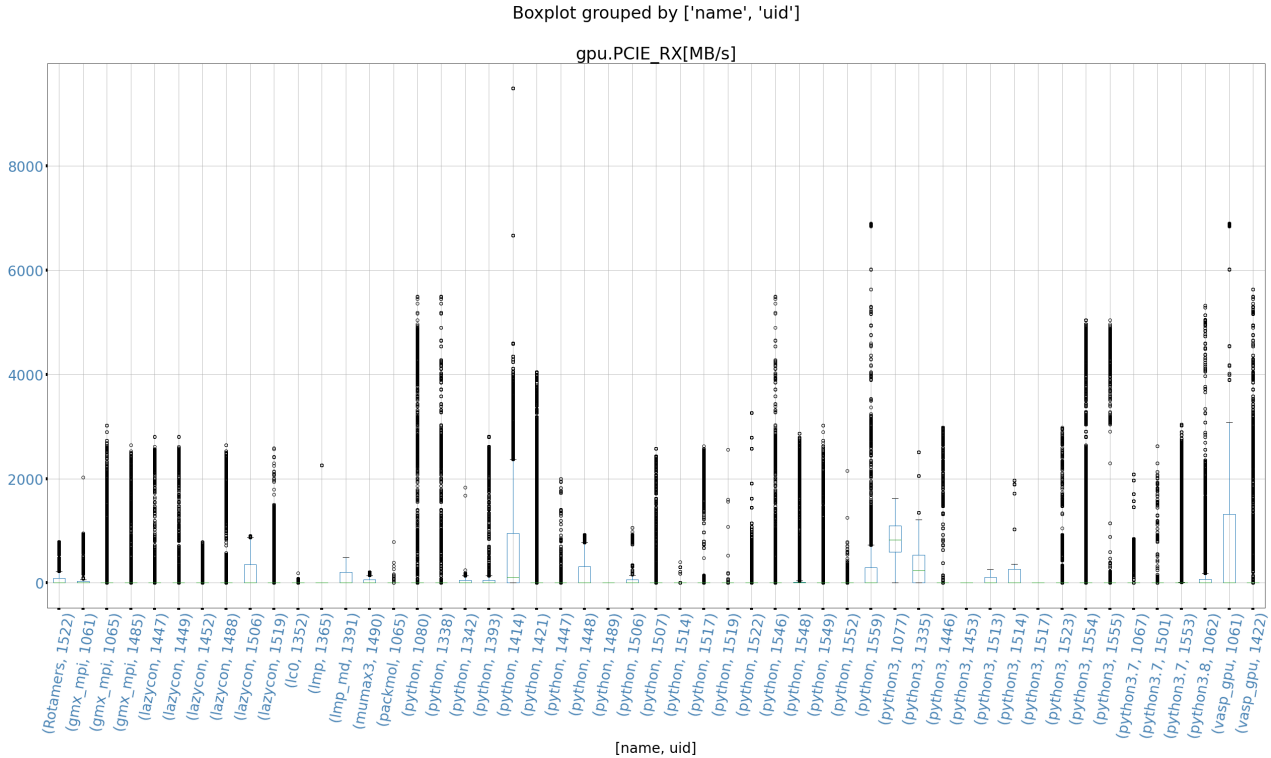

Рисунок 8. Загрузка данных в память ГПУ по шине PCIe

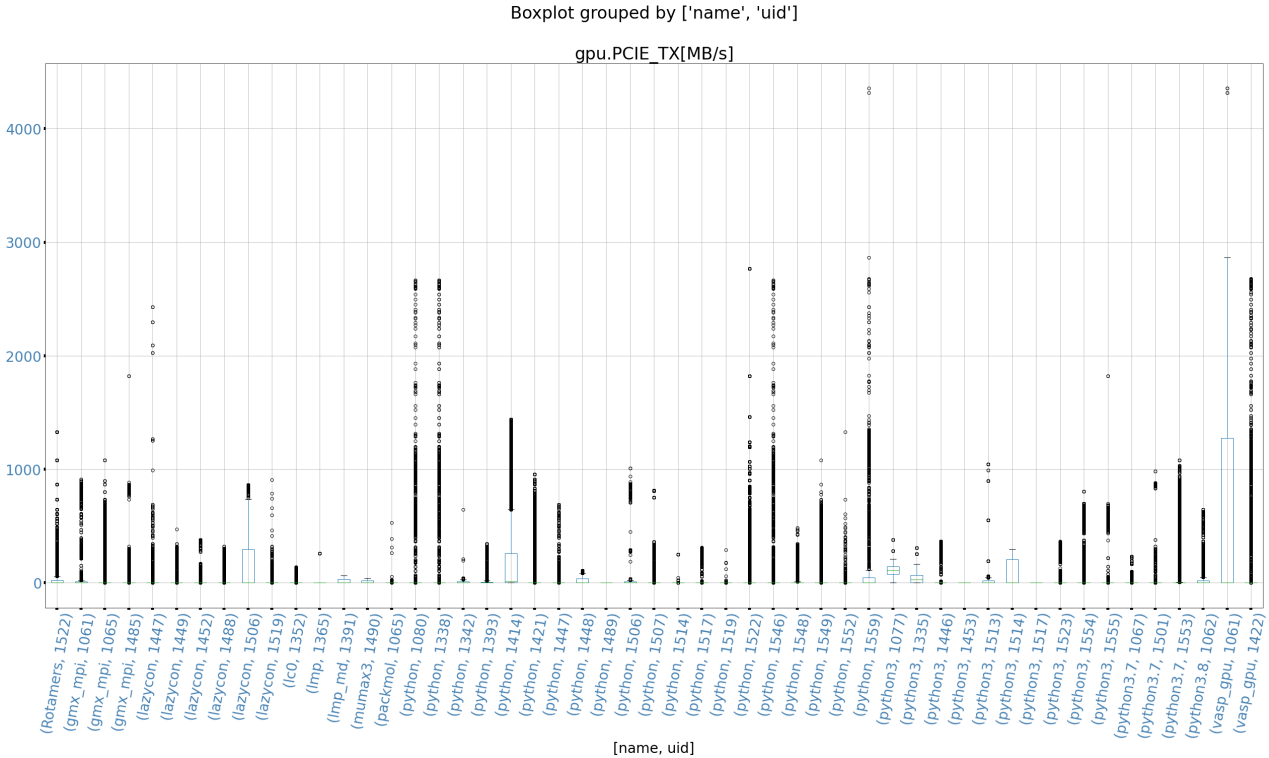

Рисунок 9. Передача результатов вычислений из ГПУ по шине PCIe

Заключение

В статье рассмотрена система мониторинга приложений на кластере ZHORES в Сколтехе. Система разработана с учетом специфики кластера ZHORES, используя предыдущий опыт разработки таких систем. Отличительной особенностью разработанной системы мониторинга является сбор аппаратных счетчиков событий ЦПУ и ГПУ и возможность корреляции этих измерений с другими событиями на кластере.

Подтверждена достаточность построения системы мониторинга по «классической» схеме, использующей сервисные процессы на каждом узле и центральную базу данных Akumuli на выделенном узле, для легкой загрузки суммарного потока данных со всех узлов. Преимущество такой схемы заключается в простоте реализации и в простоте обслуживания, так как база данных Akumuli имеет фиксированный размер, и старые данные стираются по мере поступления новых данных с циклом в 10 дней.

Предметом дальнейших исследований будет реализация долгосрочного хранения важной части собранных данных.

Статистический анализ собираемых данных обеспечил возможность оценки работы узлов с точки зрения использования функциональных блоков (ЦПУ, ГПУ, память и шины соединений).

Дальнейшие исследования направляются на возможность использования этой информации для управления кластером и выявления дополнительных ресурсов для увеличения пропускной способности кластера. Например, выявление не влияющих друг на друга приложений позволит запускать их в фоновом режиме на одном узле [**15**].

Показана возможность классификации нагрузки по микроархитектурным особенностям каждого приложения. Рассмотрены комбинации метрик, позволяющие получить максимум информации по узким местам производительности приложений [**16**, **17**].

Продемонстрировано автоматическое нахождение легкодоступных возможностей повышения производительности приложений.

Дальнейшие исследования идут в направлении интерпретации получаемых данных для уточнения такой классификации и автоматизации отчетности по приложениям пользователя. Это делается для того, чтобы мотивировать пользователей на оптимизацию своих приложений, что в свою очередь должно привести к увеличению производительности и эффективности приложений и, следовательно, кластера в целом.

Список литературы

[1] F. Ciorba. “The importance and need for system monitoring and analysis in HPC operations and research”, *Proceedings of the 3rd bwHPC-Symposium* (Heidelberg, 2016), heiBOOKS, Heidelberg, 2017, ISBN 978-3-946531-70-8, pp. 7–16. doi arXiv 1807.03112 [cs.DC] ↑73,74

[2] I. Zacharov, R. Arslanov, M. Gunin, D. Stefonishin, A. Bykov, S. Pavlov, O. Panarin, A. Maliutin, S. Rykovanov, M. Fedorov. ““Zhores” — Petaflops supercomputer for data-driven modeling, machine learning and artificial intelligence installed in Skolkovo Institute of Science and Technology”, *Open Engineering*, **9**:1 (2019), pp. 512–520. doi ↑74,76

[3] A. Yoo, M. Jette, M. Grondona. “SLURM: Simple Linux Utility for Resource Management”, *JSSPP 2003: Job Scheduling Strategies for Parallel Processing* (June 24, 2003, Seattle, WA, USA), Lecture Notes in Computer Science, vol. **2862**, Springer, Berlin–Heidelberg, 2003, ISBN 978-3-540-20405-3, pp. 44–60. doi ↑74

[4] F. D. Sacerdoti, M. J. Katz, M. L. Massie, D. E. Culler. “Wide area cluster monitoring with Ganglia”, *Proc. IEEE International Conference on Cluster Computing* (1–4 Dec. 2003, Hong Kong, China), 2003, ISBN 0-7695-2066-9, pp. 289–298. doi ↑74

[5] E. Birngruber, P. Forai, A. Zauner. “Total recall: holistic metrics for broad systems performance and user experience visibility in a data-intensive computing environment”, *HUST ’15: Proceedings of the Second International Workshop on HPC User Support Tools* (November, 2015, Austin, Texas, USA), Association for Computing Machinery, New York, 2015, ISBN 978-1-4503-4000-7, 12 pp. doi ↑76,77

[6] R. Mooney, K. P. Schmidt, R. S. Studham. “NWPerf: a system wide performance monitoring tool for large Linux clusters”, *2004 IEEE Int. Conf. on Cluster Computing* (20–23 Sept. 2004, San Diego, CA, USA), 2004, ISBN 0-7803-8694-9. doi ↑76,77

[7] J. C. Browne, R. L. DeLeon, Charng-Da Lu, M. D. Jones, S. M. Gallo, A. Ghadersohi, A. K. Patra, W. L. Barth, J. Hammond, Th. R. Furlani, R. T. McLay. “Enabling comprehensive data-driven system management for large computational facilities”, *SC ’13: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis* (17–22 Nov. 2013, Denver, CO, USA), 2013, ISBN 978-1-4503-2378-9, 11 pp. doi ↑76,77

[8] T. Evans, W. L. Barth, J. C. Browne, R. L. DeLeon, T. R. Furlani, S. M. Gallo, M. D. Jones, A. K. Patra. “Comprehensive Resource Use Monitoring for HPC Systems with TACC Stats”, *2014 First International Workshop on HPC User Support Tools* (21-21 Nov. 2014, New Orleans, LA, USA), 2014, pp. 13-21. doi ↑76,77

[9] K. Stefanov, Vl. Voevodin, S. Zhumatiy, V. Voevodin. “Dynamically reconfigurable distributed modular monitoring system for supercomputers

(DiMMon)”, YSC 2015. 4th International Young Scientists Conference on Computational Science, *Procedia Computer Science*, **66** (2015), pp. 625–634. ↑76,77,83

[10] Н. С. Живчикова, Ю. В. Шевчук. «Подсистема архивации данных системы мониторинга Botikmon3», *Научный сервис в сети Интернет*, Труды XX Всероссийской научной конференции (17–22 сентября 2018 г., г. Новороссийск), ИПМ им. М.В.Келдыша, М., 2018, с. 223–229. ↑77

[11] R. Bridges, N. Imam, T. Mintz. “Understanding GPU power: A survey of profiling, modeling, and simulation methods”, *ACM Computing Surveys*, **49**:3 (2016), 41. ↑78

[12] О. Панарин, И. Захаров. «Особенности мониторинга мобильных систем обработки информации», *Электронные библиотеки*, **23**:4, Тематический выпуск «Научный сервис в сети Интернет». Часть 2 (2020), с. 835–847. ↑78

[13] J. Treibig, G. Hager, G. Wellein. “LIKWID: A lightweight performance-oriented tool suite for x86 multicore environments”, *2010 39th International Conference on Parallel Processing Workshops* (13–16 Sept. 2010, San Diego, CA, USA), 2010, 10 pp. arXiv 1004.4431 [cs.DC] ↑79

[14] Nvidia DCGM (20/10/2020); NVIDIA Management Library (NVML) (20/10/2020). ↑78,81

[15] G. Zellweger, D. Lin, T. Roscoe. “So many performance events, so little time”, *APSys ’16: Proceedings of the 7th ACM SIGOPS Asia-Pacific Workshop on Systems* (August, 2016, Hong Kong), Association for Computing Machinery, New York, ISBN 978-1-4503-4265-0, 9 pp. ↑96

[16] D. Eklov, N. Nikoleris, E. Hagersten. *A profiling method for analyzing scalability bottlenecks on multicores*, Technical Report 2012-030, Department of Information Technology, Uppsala University, 2012, 12 pp. ↑96

[17] A. Yasin. “A Top-Down method for performance analysis and counters architecture”, *2014 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)* (23–25 March 2014, Monterey, CA, USA), 2014, pp. 35–44. ↑96

Поступила в редакцию	26.01.2021
Переработана	29.03.2021
Опубликована	05.06.2021

Рекомендовал к публикации *к.ф.-м.н. А. В. Климов*

Пример ссылки на эту публикацию:

И. Е. Захаров, О. А. Панарин, С. Г. Рыкованов, Р. Р. Загидуллин, А. К. Малютин, Ю. Н. Шкандыбин, А. Е. Ермекова. «Мониторинг приложений на кластере ZHORES в Сколтехе». *Программные системы: теория и приложения*, 2021, **12**:2(49), с. 73–103.

10.25209/2079-3316-2021-12-2-73-103

http://psta.psiras.ru/read/psta2021_2_73-103.pdf

Об авторах:

Игорь Евгеньевич Захаров

старший научный сотрудник Сколковского института науки и технологий, специалист в области высоко-производительных компьютерных систем и системного программирования.

0000-0002-3256-6514

e-mail: i.zacharov@skoltech.ru

Олег Анатольевич Панарин

менеджер по информационным сервисам и обработке данных Сколковского института науки и технологий, специалист в области высокопроизводительных компьютерных систем и систем мониторинга.

0000-0003-0894-1916

e-mail: o.panarin@skoltech.ru

Сергей Георгиевич Рыкованов

PhD по физике, доцент Сколковского института науки и технологий, заведующий лабораторией высокопроизводительных вычислений и больших данных, специалист в теоретической и вычислительной физике, прикладной математике и высокопроизводительных вычислениях.

0000-0002-3268-2267

e-mail: S.Rykovanov@skoltech.ru

Ришат Раилевич Загидуллин

магистр прикладной математики и информатики, аспирант Сколковского института науки и технологий

0000-0002-4124-3485

e-mail: R.Zagidullin@skoltech.ru

Антон Константинович Малютин

Старший инженер исследователь Сколковского института Науки и Технологий, специалист в области высоко-производительных компьютерных систем, системный администратор суперкомпьютера ZHORES

0000-0003-4487-3324

e-mail: A.Maliutin@skoltech.ru

Юрий Николаевич Шкандыбин

архитектор суперкомпьютера «Жорес», проектирование HPC решений, аппаратная и программная оптимизация серверов, программных и виртуальных окружений.

0000-0003-2970-2287

e-mail: Y.Shkandybin@skoltech.ru

Асель Ермековна Ермекова

бакалавр ядерной физики, магистрант Сколковского института Науки и Технологий.

0000-0002-2064-6103

e-mail: Assel.Yermekova@skoltech.ru

CSCSTI 50.41.17
UDC 004.451

Igor E. Zacharov, Oleg A. Panarin, Sergey G. Rykovanov, Rishat R. Zagidullin, Anton K. Maliutin, Yuri N. Shkandybin, Assel Ye. Yermekova. *Monitoring applications on the ZHORES cluster at Skoltech.*

Abstract. Standard monitoring tools for cluster computing systems allow assessing the performance of the whole system, but do not allow to analyze the performance of applications individually. A monitoring system for measuring the resources requested by each application separately was written in Skoltech for the high-performance Zhores cluster. The monitoring system collects both, the usual metrics of CPU and GPU utilization, as well as the CPU and GPU event counters which allow a more detailed analysis of the resources requested by the application. Service programs deployed on each node in the cluster send measurements to a common time series database in one second increments. These data are analyzed offline to isolate the characteristics associated with the use of computing resources by each application. This should reveal suboptimal applications, allow fine-tuning of the cluster functions and improve the HPC system overall.

Key words and phrases: cluster, high performance computing, application monitoring, CPU / GPU event counters, time series database.

2020 *Mathematics Subject Classification*: 65Y05; 68M20, 68M99

© I. E. Zacharov, O. A. Panarin, S. G. Rykovanov, R. R. Zagidullin, A. K. Maliutin, Yu. N. Shkandybin, A. Ye. Yermekova, 2021
© Skolkovo Institute of Science and Technology, 2021
© Program Systems: Theory and Applications (design), 2021
10.25209/2079-3316-2021-12-2-73-103

References

[1] F. Ciorba. “The importance and need for system monitoring and analysis in HPC operations and research”, *Proceedings of the 3rd bwHPC-Symposium* (Heidelberg, 2016), heiBOOKS, Heidelberg, 2017, ISBN 978-3-946531-70-8, pp. 7–16. arXiv 1807.03112 [cs.DC] ↑73,74

[2] I. Zacharov, R. Arslanov, M. Gunin, D. Stefonishin, A. Bykov, S. Pavlov, O. Panarin, A. Maliutin, S. Rykovanov, M. Fedorov. ““Zhores” — Petaflops supercomputer for data-driven modeling, machine learning and artificial intelligence installed in Skolkovo Institute of Science and Technology”, *Open Engineering*, **9**:1 (2019), pp. 512–520. ↑74,76

[3] A. Yoo, M. Jette, M. Grondona. “SLURM: Simple Linux Utility for Resource Management”, *JSSPP 2003: Job Scheduling Strategies for Parallel Processing* (June 24, 2003, Seattle, WA, USA), Lecture Notes in Computer Science, vol. **2862**, Springer, Berlin–Heidelberg, 2003, ISBN 978-3-540-20405-3, pp. 44–60. ↑74

[4] F. D. Sacerdoti, M. J. Katz, M. L. Massie, D. E. Culler. “Wide area cluster monitoring with Ganglia”, *Proc. IEEE International Conference on Cluster Computing* (1–4 Dec. 2003, Hong Kong, China), 2003, ISBN 0-7695-2066-9, pp. 289–298. ↑74

[5] E. Birngruber, P. Forai, A. Zauner. “Total recall: holistic metrics for broad systems performance and user experience visibility in a data-intensive computing environment”, *HUST ’15: Proceedings of the Second International Workshop on HPC User Support Tools* (November, 2015, Austin, Texas, USA), Association for Computing Machinery, New York, 2015, ISBN 978-1-4503-4000-7, 12 pp. ↑76,77

[6] R. Mooney, K. P. Schmidt, R. S. Studham. “NWPerf: a system wide performance monitoring tool for large Linux clusters”, *2004 IEEE Int. Conf. on Cluster Computing* (20–23 Sept. 2004, San Diego, CA, USA), 2004, ISBN 0-7803-8694-9. ↑76,77

[7] J. C. Browne, R. L. DeLeon, Charng-Da Lu, M. D. Jones, S. M. Gallo, A. Ghadersohi, A. K. Patra, W. L. Barth, J. Hammond, Th. R. Furlani, R. T. McLay. “Enabling comprehensive data-driven system management for large computational facilities”, *SC ’13: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis* (17–22 Nov. 2013, Denver, CO, USA), 2013, ISBN 978-1-4503-2378-9, 11 pp. ↑76,77

[8] T. Evans, W. L. Barth, J. C. Browne, R. L. DeLeon, T. R. Furlani, S. M. Gallo, M. D. Jones, A. K. Patra. “Comprehensive Resource Use Monitoring for HPC Systems with TACC Stats”, *2014 First International Workshop on HPC User Support Tools* (21-21 Nov. 2014, New Orleans, LA, USA), 2014, pp. 13-21. ↑76,77

[9] K. Stefanov, Vl. Voevodin, S. Zhumatiy, V. Voevodin. “Dynamically reconfigurable distributed modular monitoring system for supercomputers (DiMMon)”, YSC 2015. 4th International Young Scientists Conference on Computational Science, *Procedia Computer Science*, **66** (2015), pp. 625–634. ↑76,77,83

[10] N. S. Zhivchikova, Yu. V. Shevchuk. “Podsistema arkhivatsii dannykh sistemy monitoringa Botikmon3”, *Nauchnyy servis v seti Internet*, Trudy XX Vserossiyskoy nauchnoy konferentsii (17–22 sentyabrya 2018 g., g. Novorossiysk), IPM im. M.V.Keldysha, M., 2018, pp. 223–229. URL ↑77

[11] R. Bridges, N. Imam, T. Mintz. “Understanding GPU power: A survey of profiling, modeling, and simulation methods”, *ACM Computing Surveys*, **49**:3 (2016), 41. ↑78

[12] O. Panarin, I. Zakharov. “Osobennosti monitoringa mobil’nykh sistem obrabotki informatsii”, *Elektronnyye biblioteki*, **23**:4, Tematicheskiy vypusk “Nauchnyy servis v seti Internet”. Chast’ 2 (2020), pp. 835–847. ↑78

[13] J. Treibig, G. Hager, G. Wellein. “LIKWID: A lightweight performance-oriented tool suite for x86 multicore environments”, *2010 39th International Conference on Parallel Processing Workshops* (13–16 Sept. 2010, San Diego, CA, USA), 2010, 10 pp. arXiv 1004.4431 [cs.DC] ↑79

[14] Nvidia DCGM (20/10/2020); NVIDIA Management Library (NVML) (20/10/2020). ↑78,81

[15] G. Zellweger, D. Lin, T. Roscoe. “So many performance events, so little time”, *APSys ’16: Proceedings of the 7th ACM SIGOPS Asia-Pacific Workshop on Systems* (August, 2016, Hong Kong), Association for Computing Machinery, New York, ISBN 978-1-4503-4265-0, 9 pp. ↑96

[16] D. Eklov, N. Nikoleris, E. Hagersten. *A profiling method for analyzing scalability bottlenecks on multicores*, Technical Report 2012-030, Department of Information Technology, Uppsala University, 2012, 12 pp. ↑96

[17] A. Yasin. “A Top-Down method for performance analysis and counters architecture”, *2014 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)* (23–25 March 2014, Monterey, CA, USA), 2014, pp. 35–44. ↑96

Sample citation of this publication:

Igor E. Zacharov, Oleg A. Panarin, Sergey G. Rykovanov, Rishat R. Zagidullin, Anton K. Maliutin, Yuri N. Shkandybin, Assel Ye. Yermekova. “Monitoring applications on the ZHORES cluster at Skoltech”. *Program Systems: Theory and Applications*, 2021, **12**:2(49), pp. 73–103. (*In Russian*).

10.25209/2079-3316-2021-12-2-73-103

http://psta.psiras.ru/read/psta2021_2_73-103.pdf