2023, **24** (1), 24–36. doi 10.26089/NumMet.v24r103

doi 10.26089/NumMet.v24r103 УДК 519.68

Создание переносимого программного комплекса для мониторинга и анализа производительности суперкомпьютерных приложений

В. В. Воеводин
Московский государственный университет имени М. В. Ломоносова,
Научно-исследовательский вычислительный центр, Москва, Российская Федерация
ORCID: 0000-0003-1897-1828, e-mail: vadim@parallel.ru

К. С. Стефанов
Московский государственный университет имени М. В. Ломоносова,
Научно-исследовательский вычислительный центр, Москва, Российская Федерация
ORCID: 0000-0002-0930-2713, e-mail: cstef@parallel.ru

Аннотация: Современные суперкомпьютеры востребованы в самых разных областях науки и техники. Однако их вычислительные ресурсы зачастую используются не в полной мере. Причина нередко кроется в низкой эффективности выполнения пользовательских приложений. Решить возникшую проблему весьма непросто, что связано как с чрезвычайной сложностью строения современных суперкомпьютеров, так и с недостатком теоретических знаний и практического опыта в создании высокоэффективных параллельных приложений у пользователей вычислительных систем. Более того, пользователи зачастую и не знают, что их приложения работают неэффективно. Поэтому важно, чтобы администраторы суперкомпьютеров могли постоянно контролировать и анализировать весь поток выполняющихся приложений. Для этих целей можно использовать различные существующие системы мониторинга и анализа производительности, однако подобные решения в большинстве своем либо не предоставляют достаточный функционал в части изучения производительности, либо не переносимы. В данной работе описывается прототип разрабатываемого программного комплекса, который предоставляет широкие возможности по сбору и автоматическому анализу данных о производительности приложений и при этом является переносимым.

Ключевые слова: параллельные вычисления, суперкомпьютер, мониторинг, анализ данных, производительность, переносимость.

Благодарности: Исследование выполнено в рамках научной программы Национального центра физики и математики (проект “Национальный центр исследования архитектур суперкомпьютеров”).

Для цитирования: Воеводин В.В., Стефанов К.С. Создание переносимого программного комплекса для проведения мониторинга и анализа производительности суперкомпьютерных приложений // Вычислительные методы и программирование. 2023. **24**, № 1. 24–36. doi 10.26089/NumMet.v24r103.

© В. В. Воеводин, К. С. Стефанов

2023, **24** (1), 24–36. doi 10.26089/NumMet.v24r103

Development of a portable software solution for monitoring and analyzing the performance of supercomputer applications

Vadim V. Voevodin
Lomonosov Moscow State University, Research Computing Center, Moscow, Russia
ORCID: 0000-0003-1897-1828, e-mail: vadim@parallel.ru
Konstantin S. Stefanov
Lomonosov Moscow State University, Research Computing Center, Moscow, Russia
ORCID: 0000-0002-0930-2713, e-mail: cstef@parallel.ru

Abstract: Modern supercomputers are used in various areas of science and technology. However, their computational resources are often not fully utilized. The reason often lies in the low efficiency of user applications. However, it is very difficult to solve this problem, which is due both to the extreme complexity of the structure of modern supercomputers, and to the lack of theoretical knowledge and practical experience in creating highly efficient parallel applications among users of computing systems. Moreover, users are often not even aware that their applications are not working efficiently. Therefore, it is important for supercomputer administrators to be able to constantly monitor and analyze the entire flow of running applications. For these purposes, different existing systems for monitoring and analyzing performance can be used, however, most of such solutions do not provide sufficient functionality for studying performance, or are not portable. This paper describes a prototype of the software package being developed, which provides wide opportunities for collecting and automatically analyzing application performance data and is portable at the same time.

Keywords: parallel computing, supercomputer, monitoring, data analysis, performance, portability.

Acknowledgements: The study was carried out within the framework of the scientific program of the National Center for Physics and Mathematics (“National Center for Supercomputer Architecture Research” project)

For citation: V. V. Voevodin, K. S. Stefanov, “Development of a portable software solution for monitoring and analyzing the performance of supercomputer applications,” Numerical Methods and Programming. **24** (1), 24–36 (2023). doi 10.26089/NumMet.v24r103.

1. Введение. Вопрос эффективности использования суперкомпьютерных ресурсов давно является актуальной темой, исследуемой многими учеными по всему миру. Можно выделить две основные причины этого. Во-первых, область высокопроизводительных вычислений становится все более востребованной [1], и современные суперкомпьютеры используются для моделирования в самых разных предметных областях. При этом вычислительные ресурсы стóят очень дорого [2], и соответственно, любой заметный простой ресурсов приводит к значительным финансовым потерям. Во-вторых, архитектура современных суперкомпьютеров настолько сложна, что написать параллельную программу, которая будет решать реальную научную задачу и при этом будет обладать высокой эффективностью (т.е. будет аккуратно учитывать все тонкости используемого аппаратного обеспечения), очень непросто.

В работе [3] отмечено, что пользователи не всегда уделяют достаточно внимания вопросам эффективности выполнения своих приложений и поэтому могут быть не в курсе наличия в них проблем с производительностью. В такой ситуации забота об обеспечении и поддержании высокой эффективности функционирования вычислительных систем на практике во многом ложится на плечи системных администраторов и менеджеров суперкомпьютерных центров (хотя отметим, что они и так заинтересованы в

2023, **24** (1), 24–36. doi 10.26089/NumMet.v24r103

этом, поскольку повышение эффективности приводит к тому, что при тех же физических ресурсах центра решается больше задач, можно обслужить больше пользователей и т.п.).

Для решения этих вопросов необходимо иметь постоянный контроль за реальной производительностью суперкомпьютера, в первую очередь — за эффективностью выполняемых на нем пользовательских приложений. Сразу отметим, что мы в данной работе не затрагиваем вопросы обеспечения надежности и работоспособности суперкомпьютера, которые уже достаточно успешно решаются с помощью различных систем мониторинга (Zabbix, Nagios и т.д.) и другого системного программного обеспечения.

Чтобы контролировать эффективность использования вычислительных ресурсов, необходимо успешно решать две важные задачи — постоянно собирать данные о производительности выполнения приложений, а также уметь извлекать из этой информации полезную информацию, т.е. уметь анализировать собираемые данные. Для решения первой задачи требуется система мониторинга, которая умеет собирать данные о производительности, вторая задача требует наличия аналитической системы. В настоящее время существует ряд решений как для первой, так и второй задачи, которые могут использоваться для поставленных целей, однако все они либо предоставляют достаточно небольшой функционал, либо не являются переносимыми.

Нами ведется разработка программного комплекса, призванного устранить оба этих недостатка. Данный комплекс, состоящий из системы мониторинга и системы анализа, будет переносимым и при этом предоставлять широкий функционал по интеллектуальному автоматическому анализу данных о качестве работы суперкомпьютера. Он строится на основе существующих решений DiMMon и TASC. Связка DiMMon–TASC развивалась в течение длительного времени на суперкомпьютере Ломоносов-2, при этом задача по обеспечению переносимости на другие системы не ставилась. Эта статья описывает действия, при помощи которых мы из комплекса, предназначенного для конкретной вычислительной системы, создаем переносимый комплекс с тем же функционалом, пригодный для работы на целом классе суперкомпьютеров.

Основной результат, представляемый в данной статье, заключается в создании прототипа указанного комплекса. На текущий момент этот комплекс позволяет собирать данные с вычислительных узлов о производительности выполняемых на них приложений, а затем на основе этих данных автоматически обнаруживать потенциальные проблемы с производительностью на основе заданного набора правил. Комплекс успешно прошел первичную апробацию, в дальнейшем мы планируем выложить его исходный код в открытый доступ.

Далее статья устроена следующим образом. В разделе 2 приведено описание смежных работ в данной области. Раздел 3 посвящен описанию текущей архитектуры разрабатываемого программного комплекса. Краткое описание проведенной апробации комплекса на практике приведено в разделе 4. Раздел 5 содержит основные выводы, а также наши дальнейшие планы по развитию данного решения.

2. Обзор существующих решений. Как было сказано ранее, разрабатываемый программный комплекс состоит из двух основных частей — системы мониторинга и системы анализа. Рассмотрим далее, какие имеются решения по каждому из двух направлений.

Существует ряд решений, предназначенных для мониторинга производительности суперкомпьютеров. Обзор таких систем со сравнением по многим параметрам приведен в работе [4]. В дополнение к сведениям, приведенным в указанном обзоре, мы рассмотрим системы мониторинга производительности с точки зрения их переносимости.

Переносимость систем мониторинга можно рассматривать с разных точек зрения. Можно рассматривать с точки зрения аппаратных платформ и программного окружения, для использования на которых предназначена та или иная система. Можно рассматривать с точки зрения того, насколько подробна документация по настройке системы мониторинга под конкретную вычислительную систему. Наконец, можно рассматривать, насколько легко убедиться в том, что система работает корректно.

С точки зрения программных и аппаратных платформ, все использующиеся в настоящее время системы мониторинга, из перечисленных в указанном обзоре, предполагают использование в рамках ОС Linux и на платформах с архитектурой x86_64. Система Performance Co-Pilot [5] дополнительно может использоваться на системах с архитектурой ARM64 и Power, а система LIKWID Monitoring Stack [6] также может работать на системах, основанных на ARM64.

Подробного описания процесса настройки на конкретную вычислительную систему мы не нашли ни для одной из рассматриваемых систем. Как правило, имеется инструкция по сборке и установке системы, но подробности по настройке обычно распределены по частям документации, связанным с настройкой

2023, **24** (1), 24–36. doi 10.26089/NumMet.v24r103

конкретных подсистем. Поэтому развертывание любой из систем мониторинга производительности представляет собой трудоемкую задачу, и упрощение этого процесса облегчит применение средств мониторинга производительности на большем числе вычислительных систем.

Проверка корректности работы уже установленной системы тоже представляет собой нетривиальную задачу. Для большей части систем никаких специальных средств для этого не существует, есть лишь несколько исключений. Для систем Ganglia [7] и Ovis-2 [8] в комплекте идет набор тестов для компонентов этих программных систем. Эти тесты предназначены для выполнения в сборочной среде. Для суперкомпьютеров обычно сборочная среда совпадает или сильно похожа на ту среду, в которой выполняются пользовательские приложения, поэтому такие тесты можно считать хорошим шагом к проверке корректности работы систем мониторинга. Вопрос стоит в полноте покрытия функциональности системы мониторинга имеющимися тестами, но проверка такой полноты является нетривиальной задачей.

Для систем LIKWID Monitoring Stack и Performance Co-Pilot разработан полноценный конвейер Continuous Integration (CI, непрерывная интеграция). Он предполагает автоматическую компиляцию и выполнение тестов после внесения изменений в исходный код системы. При этом тесты выполняются на нескольких аппаратных и программных платформах. Такая регулярная проверка позволяет сократить количество ошибок в релизных версиях указанных систем и, кроме того, дает хорошую основу для тестирования корректности работы после установки системы мониторинга на новой вычислительной установке.

Отметим, что системы, имеющие конвейер CI, — это те же самые системы, которые могут работать на разных аппаратных платформах. Можно сказать, что усложнение разработки из-за поддержки нескольких аппаратных платформ должно подталкивать разработчиков программных систем к использованию практик непрерывной интеграции для сокращения числа ошибок при разработке. Однако исследование того, насколько такая связь часто встречается, выходит за рамки данной работы.

Таким образом, можно сказать, что хотя разработка большинства систем мониторинга производительности для суперкомпьютеров ведется, вероятно, с прицелом на переносимость систем (никто не заявляет, что их система мониторинга предназначена исключительно для вычислительной системы самих разработчиков), тем не менее вопросам легкости переноса систем мониторинга на другие вычислительные системы не уделяется много внимания. Есть обоснованное предположение, что такие вопросы начинают интересовать разработчиков, когда они сами сталкиваются с необходимостью переноса их систем на другие вычислительные системы или аппаратные платформы. Отметим, что для систем суперкомпьютерного мониторинга общего назначения ситуация заметно лучше. Вероятно, это связано с более широкой областью использования таких систем и, как следствие, с необходимостью прикладывать больше усилий для обеспечения переносимости.

С точки зрения анализа качества использования суперкомпьютерных ресурсов, набор существующих решений и исследований не так велик. В основной массе подобный анализ выполняется вручную, требует существенных усилий и не является переносимым, хотя и позволяет досконально изучить происходящее на выбранной системе или ряде систем [9–11]. Также существует ряд программных инструментов, которые позволяют с разных сторон оценивать производительность всего потока суперкомпьютерных приложений и детально рассматривать эффективность работы отдельных приложений [12–15], однако они либо позволяют выявлять, на наш взгляд, лишь достаточно простые случаи снижения производительности, либо вообще не предоставляют такой возможности, перекладывая задачу выявления проблем с производительностью на самого пользователя. В нашем случае интерес представляет детальный анализ, позволяющий автоматически обнаруживать широкий спектр проблем с производительностью в параллельных приложениях.

Отметим, что в данном случае мы не рассматриваем многообразие существующих решений для анализа производительности отдельных приложений (профилировщики, средства отладки и трассировки, эмуляторы), поскольку они применяются на следующем этапе анализа — когда уже стало понятно, что требуется изучить одно определенное приложение. Наше же решение применяется на начальном этапе, когда нужно анализировать весь поток выполняющихся приложений и выявлять в нем приложения с потенциальными проблемами с производительностью, требующие дальнейшего отдельного анализа.

3. Архитектура разрабатываемого переносимого комплекса. Общая архитектура разрабатываемого комплекса показана на рис. 1. Комплекс состоит из двух основных подсистем — мониторинга (`Monitoring subsystem`, ее модули отмечены желтым на рисунке) и анализа (`Analysis subsystem`, модули отмечены зеленым). На каждом вычислительном узле суперкомпьютера запускается агент мониторинга (`Monitoring agent`, `MA`), который постоянно собирает с него данные о производительности выполняе-

2023, **24** (1), 24–36. doi 10.26089/NumMet.v24r103

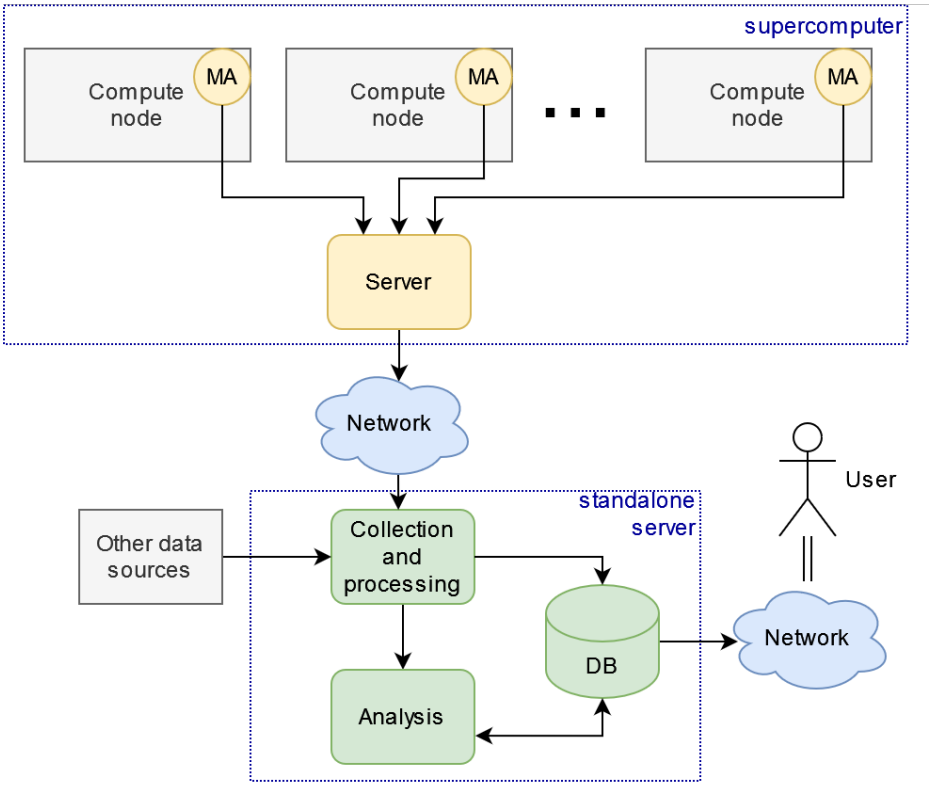

Рис. 1. Общая архитектура разрабатываемого комплекса

Fig. 1. General architecture of the developed complex

мых на нем пользовательских заданий и отправляет в серверную часть системы мониторинга. Серверная часть (`Server`) нужна для первичной обработки поступающих данных, например для вычисления метрик по заданиям; также она может применяться для контроля за работой агентов. Отметим, что если такой функционал не является необходимым на суперкомпьютере, то возможна работа системы мониторинга без серверной части. В этом случае контроль за работоспособностью агентов может осуществляться с помощью скриптов в прологе/эпилоге менеджера ресурсов.

Данные от серверной части подсистемы мониторинга (или напрямую от агентов мониторинга) поступают в подсистему анализа. Отметим, что взаимодействие двух подсистем, хотя и является ключевым, устроено очень просто: данные от подсистемы мониторинга по сети постоянно передаются в подсистему анализа, при этом со стороны подсистемы мониторинга требуется только указать нужные IP адреса и номера портов, куда передавать данные. В случае подсистемы анализа все еще проще: необходимо указать номера портов, на которых подсистема будет ожидать данные от подсистемы мониторинга.

Данные, поступающие в подсистему анализа, при необходимости объединяются с данными от других источников (таких как данные о запуске заданий, данные о проектах и т.д.), обрабатываются в модуле `Collection and processing` и затем анализируются в модуле `Analysis` для выявления проблем с производительностью в заданиях. Полученные результаты анализа могут быть доступны пользователю разрабатываемого комплекса либо через web-интерфейс, либо напрямую из базы данных (БД, DB). Отметим, что подсистема анализа получает все требуемые данные через сеть, поэтому она может работать на отдельном служебном сервере, не обязательно являющемся частью инфраструктуры суперкомпьютера. Во многих случаях это оказывается полезным свойством, поскольку процесс хранения и обработки большого объема данных (объем данных от системы мониторинга может быть очень большим) требует немало ресурсов. Также отметим, что реализация большей части функционала подсистемы анализа выполнена с использованием Docker-контейнеров, что заметно упрощает процесс ее установки и настройки.

На данный момент мы рассматриваем переносимость разрабатываемого решения в рамках систем, для которых выполняются следующие требования: кластерная архитектура, использование на узлах ОС

2023, **24** (1), 24–36. doi 10.26089/NumMet.v24r103

Linux семейства Red Hat, использование менеджера ресурсов Slurm, запрет на выполнение нескольких пользовательских заданий одновременно на одном вычислительном узле.

Описав общую архитектуру предлагаемого комплекса, перейдем к более детальному рассмотрению схемы работы каждой из подсистем в отдельности.

3.1. Подсистема мониторинга. Прототип подсистемы мониторинга, основанный на разработанной ранее в НИВЦ МГУ системе DiMMon [16], построен по модульному принципу: получение информации от ОС и аппаратуры и обработка данных осуществляются в отдельных модулях. Набор модулей и связи между ними задаются конфигурационным файлом, который представляет собой сценарий на языке Lua.

В общем случае система мониторинга, построенная на основе DiMMon, состоит из агентов, работающих на вычислительных узлах, и серверной части. Агенты на узлах получают информацию о состоянии узла и выполняемых на нем операциях, осуществляют ее первичную обработку и затем передают для дальнейшей обработки и сохранения в серверную часть. Серверная часть осуществляет дополнительную обработку, в частности собирает агрегированную информацию из данных от нескольких вычислительных узлов. В настоящий момент, в рамках разработки создаваемого прототипа, функции серверной части системы мониторинга выполняет подсистема анализа, и данные от агентов на вычислительных узлах напрямую поступают в подсистему анализа и обрабатываются там. Поэтому подсистема мониторинга не имеет в нашем случае самостоятельной серверной части и состоит только из агентов, работающих на вычислительных узлах. Однако в дальнейшем будет добавлена возможность применения при необходимости серверной части мониторинга, что расширит функционал разрабатываемого комплекса.

Схема агента мониторинга разработанной подсистемы представлена на рис. 2. `Timer` — это модуль, отвечающий за выполнение действий с заданной периодичностью. В нашей подсистеме используются три модуля `Timer`. Модуль с периодом 1 с используется для активации модулей `Sensor`. Модули с периодом 60 с и 600 с используются для активации модулей `Derivative`, которые по сигналу `Timer` передают накопленные за период усреднения (между сигналами `Timer`) данные для последующей обработки. Количество модулей `Timer` и период между их сигналами могут быть легко изменены при необходимости.

`Sensor` — это модуль, который получает данные от ОС и аппаратуры вычислительного узла. Такие модули бывают двух типов: `Gauge` и `Counter`. `Gauge` выдает данные, которые непосредственно могут использоваться в дальнейшем. Примерами данных, с которыми работает этот тип модуля, являются объем свободной оперативной памяти, загрузка процессора и т.п. `Counter` — это модуль, который выдает количество событий определенного вида, произошедшее с какого-то момента в прошлом. Примерами данных, с которыми работает этот тип модуля, являются счетчики объема данных, переданных через сетевой интерфейс, количество промахов в кеш-память, количество выполненных инструкций процессора. Обычно абсолютные значения для таких данных не представляют интерес, и для последующего анализа нужна скорость изменения значений, для чего нужно вычислить производную по времени. Полученный резуль-

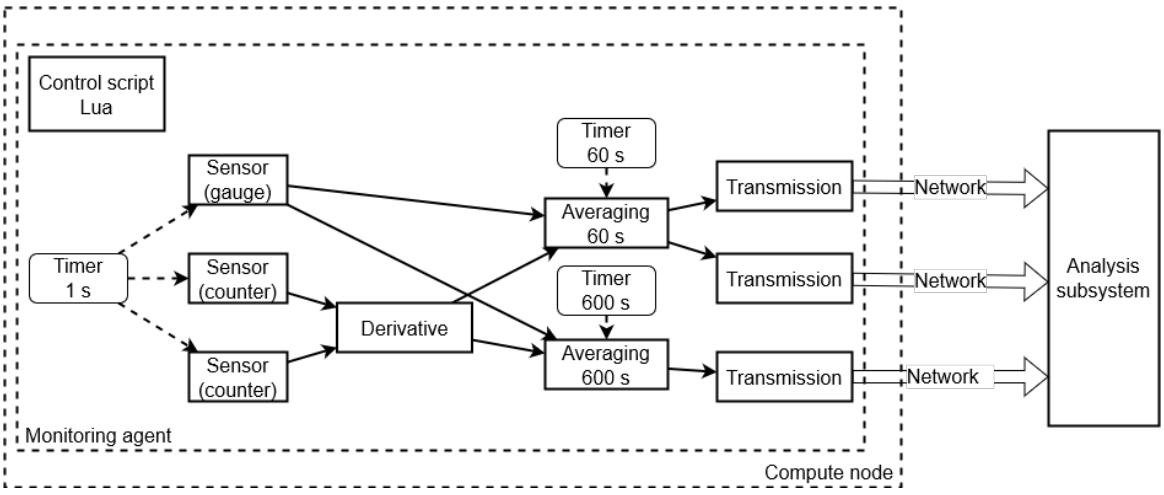

Рис. 2. Схема агента мониторинга

Fig. 2. Monitoring agent schema

2023, **24** (1), 24–36. doi 10.26089/NumMet.v24r103

тат представляет собой, например, скорость передачи данных через сетевой интерфейс. Указанный на рис. 2 модуль `Derivative` выполняет такое преобразование.

Модуль `Averaging` производит усреднение данных по промежутку времени, определяемому сигналами от `Timer`. Для промежутка усреднения для каждого вида данных вычисляется три значения: минимум, максимум и среднее из всех значений, поступивших за промежуток усреднения.

После усреднения и других необходимых преобразований данные с помощью модуля `Transmission` передаются в подсистему анализа.

На текущий момент подсистема мониторинга настроена на сбор следующих данных:

- загрузка процессора (CPU user load, system, idle, nice, iowait);
- данные от процессорных датчиков (частота L1 и LLC кеш-промахов; число выполненных инструкций в секунду; число тактов, когда процессор не простаивал);
- данные о работе коммуникационной сети (объем переданных/полученных байт/пакетов в секунду);
- аналогичные данные для сети I/O;
- данные о загрузке GPU (загрузка GPU процессора, загрузка GPU памяти);
- load average – среднее число активных процессов на узел;
- объем доступной оперативной памяти;
- уровень активности использования распределенной файловой системы Lustre;
- количество ошибок от подсистемы ECC (Error Checking and Correction) контроллера оперативной памяти.

3.2. Подсистема анализа. Прототип подсистемы для проведения анализа производительности суперкомпьютерных приложений построен на основе существующего программного комплекса TASC [17], применяемого на суперкомпьютере Ломоносов-2. Общая архитектура подсистемы анализа представлена на рис. 3. Описанные в предыдущем разделе данные поступают в подсистему с заданной частотой и сохраняются в PostgreSQL базе данных (`DB0` на схеме). Процесс импорта данных организован модулем `DAS2` на схеме (DAS — data import scenario, сценарий импорта данных). Этот модуль постоянно ждет данные на указанном порту, при получении распаковывает и обрабатывает полученный пакет и затем сохраняет данные в БД.

Поскольку поток данных от системы мониторинга может быть большим, размер `DB0` может быстро расти. При этом к `DB0` постоянно выполняются запросы на чтение. Поэтому, чтобы работа с `DB0` не стала узким местом, было решено дополнительно использовать еще две PostgreSQL базы данных — `DB1` и `DB2`. `DB1` хранит те же данные и с той же частотой, что и `DB0`, только заметно более длительное время (`DB0` — $\sim$3 недели, `DB1` — $\sim$3 месяца). Поскольку более старые данные нужны гораздо реже, то и запросы к `DB1` случаются не так часто. `DB2` предназначена для постоянного хранения всех собранных данных. Однако в этом случае приходится хранить данные с меньшей гранулярностью, иначе объем `DB2` станет слишком большим. На текущий момент `DB2` хранит данные с гранулярностью 10 минут. Отметим, что во все три БД идут свои потоки данных от подсистемы мониторинга; для этого организованы три идентичных экземпляра `DAS2`, отличающиеся только номером порта, с которого приходят данные, а также информацией о доступе к БД.

Процесс получения, обработки и анализа данных о выполняемых на суперкомпьютерной системе заданиях устроен следующим образом. Менеджер ресурсов (в нашем случае используется Slurm) имеет возможность запуска произвольных команд при старте и завершении задания (в прологе и эпилоге задания). Мы используем эту возможность для запуска `DAS1`. Этот модуль передает данные о задании (время постановки в очередь, старта и завершения задания; число и список узлов; используемый раздел; имя пользователя и т.д.) в `Module1` посредством брокера сообщений RabbitMQ для того, чтобы запустить основную часть обработки. Использование RabbitMQ позволяет существенно упростить процесс взаимодействия модулей, при этом также помогая обеспечить постоянную работоспособность (например, за счет реализации очереди сообщений, которая обеспечивает корректную обработку нескольких одновременно пришедших сообщений).

`Module1` разбирает данные от Slurm и сохраняет нужную информацию в основную базу данных MongoDB. Затем он передает управление в `Module2`, также используя RabbitMQ. `Module2` отвечает за получение данных о производительности для конкретного задания. Для этого `Module2` при получении

2023, **24** (1), 24–36. doi 10.26089/NumMet.v24r103

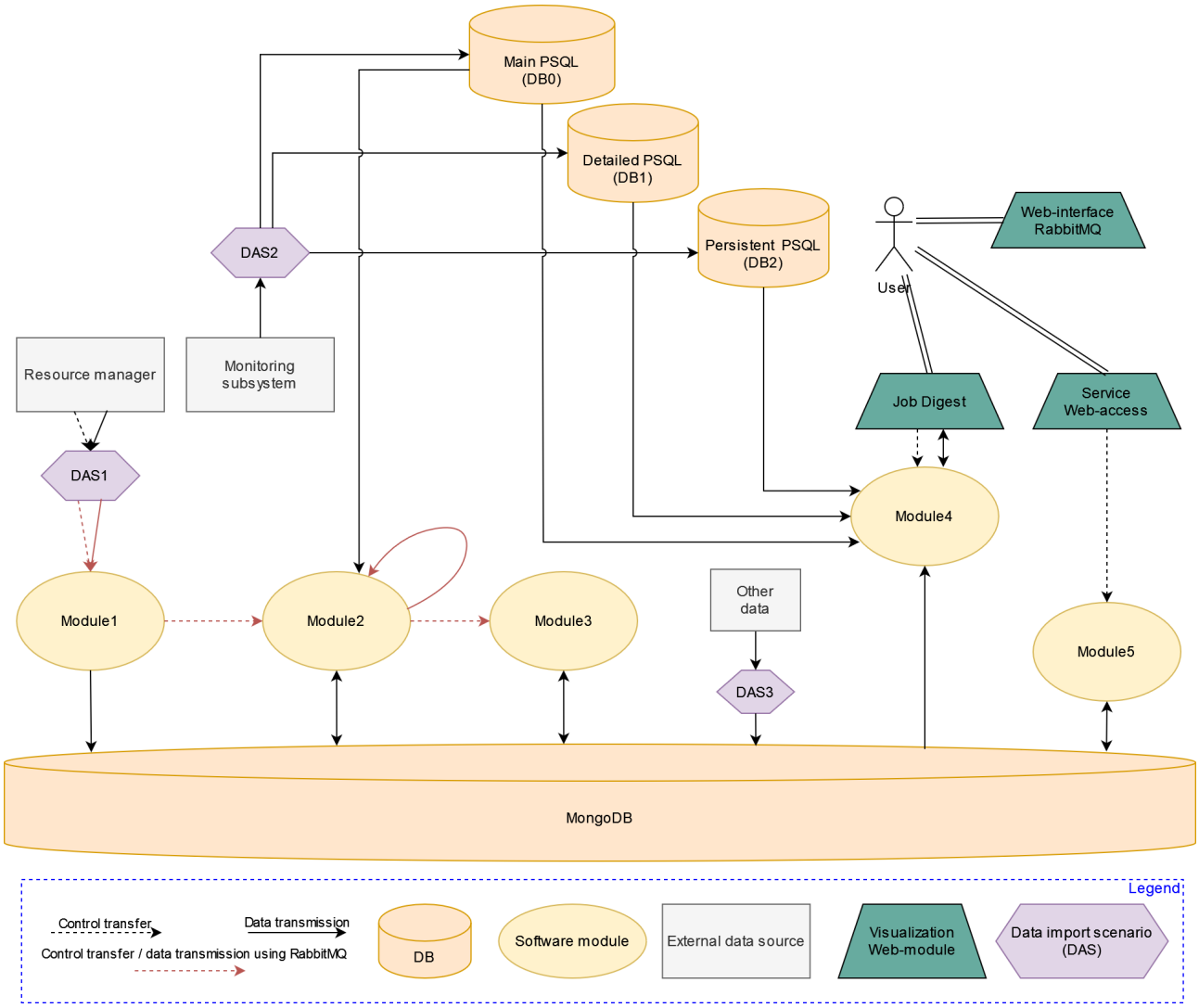

Рис. 3. Общая архитектура подсистемы анализа

Fig. 3. General architecture of the analysis subsystem

запроса от `Module1` получает нужные данные из PostgreSQL базы `DB0` (фильтруя по времени выполнения и списку узлов), на основе этих данных вычисляет интегральные значения (среднее, минимум, максимум, средний минимум и средний максимум) и сохраняет их также в MongoDB. Дополнительно `Module2` вычисляет вспомогательные метрики на основе полученных данных. К таким метрикам относятся: размер пакетов, передаваемых/получаемых по коммуникационным сетям; отношение частоты L1 к частоте LLC кеш-промахов (помогает оценивать локальность использования памяти); сетевая локальность (насколько далеко расположены узлы, задействованные в программе). Если задание еще не завершилось, этот модуль заново запускает сам себя через заданный промежуток времени (по умолчанию — 1 час), чтобы обновить данные о производительности и метрики по заданию.

Затем управление с помощью RabbitMQ передается в `Module3` — основной аналитический модуль подсистемы анализа, который для выбранного задания проверяет срабатывание правил. В подсистеме анализа задан набор правил — проверок наличия проблем с производительностью. Каждое правило описывает одну определенную проблему и задает критерий для ее обнаружения. Информацию об обнаруженных проблемах `Module3` также записывает в MongoDB. Всего на текущий момент задано ∼20 правил. Пример конкретного правила: если задание запущено в специальном разделе для GPU-заданий, однако практически не использует графические процессоры, то необходимо поменять раздел для такого задания. Другой пример: если задание активно работает с MPI сетью, но сетевая локальность плохая (узлы су-

2023, **24** (1), 24–36. doi 10.26089/NumMet.v24r103

перкомпьютера расположены далеко друг от друга), то при запуске задания менеджером ресурсов был выбран неудачный набор узлов; рекомендуется при необходимости явно указывать узлы, на которых будет производиться запуск программы, либо пытаться оптимизировать работу с MPI. Более детально правила описаны в [18].

После того как для выбранного задания отработали все указанные выше модули, работа с заданием выполнена. Далее необходимо иметь возможность получать результаты анализа и просто собранную информацию по заданию. Для этих целей создан `Module4`, который отвечает за интеграцию с системой генерации отчетов JobDigest [19], разработанной ранее в НИВЦ МГУ. `Module4` позволяет при запросе пользователя через web-браузер отображать детальную информацию о задании: интегральные характеристики производительности и данные о запуске (берутся из MongoDB), а также графики по каждой характеристике производительности, показывающие динамику изменения ее значений за время выполнения задания (берутся из PostgreSQL).

В рамках подсистемы анализа также предоставляется единый API для импорта других возможных входных данных — `DAS3`. В частности, единый API применяется в Суперкомпьютерном центре МГУ для получения данных о принадлежности пользователей к научным проектам, организациям и предметным областям; данных об исполняемых и объектных файлах, в частности данных об используемых библиотеках, прикладных пакетах и компиляторах; данных о доступности и загрузке служебных серверов.

Помимо этого, в рамках подсистемы анализа существует несколько служебных модулей. Так, `Module5` посредством модуля `Service Web-access` предоставляет доступ для проверки срабатывания правил по выбранному заданию, а также онлайн-конструктор для удобного просмотра статистики и проверки корректности работы правил. Также доступен web-интерфейс для получения информации о работе RabbitMQ и его администрирования, для чего доступен модуль `Web-interface RabbitMQ`.

4. Апробация развертывания комплекса. Приведем краткое описание проведенной на текущий момент апробации. Основная ее цель заключалась в том, чтобы отработать процесс установки прототипа разрабатываемого комплекса на новой системе и внести необходимые правки для упрощения этого процесса в будущем.

Апробация выполнялась в два этапа. На первом этапе проводилась апробация только подсистемы анализа, с использованием существующих входных данных. Для этого был развернут и настроен экземпляр подсистемы анализа на виртуальной машине с доступом к реальным данным от суперкомпьютера Ломоносов-2. При этом не требовалась установка подсистемы мониторинга, поскольку данные мониторинга можно было получать с помощью существующей системы мониторинга DiMMon, установленной на суперкомпьютере Ломоносов-2. На втором этапе проводилась апробация всего прототипа разрабатываемого комплекса. Для этого был создан виртуальный тестовый кластер, на котором была выполнена установка обеих подсистем и настроено их взаимодействие. В этом случае все входные данные, необходимые для работы прототипа, собирались с нуля, причем как состав данных, так и архитектура кластера существенно отличались от использованных на первом этапе.

На первом этапе апробации было необходимо развернуть по возможности полный аналог существующей подсистемы анализа, используемой на суперкомпьютере Ломоносов-2. Это была первая попытка переноса, и по ее результатам было внесено немало модификаций в исходный код подсистемы, а также была создана детальная инструкция по ее развертыванию на новой системе. Среди прочего, были выполнены следующие работы:

- подготовлены сценарии для автоматизации установки подсистемы;
- устранены ошибки при начальной настройке БД PostgreSQL;
- упрощен процесс конфигурирования;
- определены специфичные требования к файловой системе раздела, где располагается БД MongoDB (файловая система должна поддерживать операцию fallocate, поскольку этого требует подсистема хранения WiredTiger);
- оптимизированы базы данных PostgreSQL и MongoDB за счет тонкой настройки ряда конфигурационных параметров. В частности, при работе с PostgreSQL нами были подобраны параметры по числу соединений, размеру служебных буферов, числу параллельных процессов и др., а при работе с MongoDB подобран подходящий размер объема внутренней кеш-памяти.

Последний пункт оказался очень важным, поскольку производительность выполнения запросов к базам данных достаточно быстро может стать узким местом, особенно при работе с суперкомпьютером

2023, **24** (1), 24–36. doi 10.26089/NumMet.v24r103

большого масштаба. В рамках апробации для подсистемы анализа были выделены две виртуальные машины суммарно с двенадцатью ядрами Intel Xeon E5-2660 и 85 ГБ ОЗУ, и этих ресурсов с запасом хватило для обработки всего потока данных с суперкомпьютера Ломоносов-2.

После того как была проверена работоспособность подсистемы анализа в простых условиях (с использованием существующих данных и в условиях, аналогичных используемым ранее), был проведен второй этап апробации. Цель второго этапа — проверить и при необходимости доработать переносимость всего прототипа комплекса, состоящего из подсистемы мониторинга и подсистемы анализа, при использовании другой суперкомпьютерной системы и других входных данных. Для этого было реализовано развертывание комплекса на тестовом макете. Тестовый макет представляет собой набор виртуальных машин: одна машина используется в качестве головного узла виртуального кластера, еще четыре — в качестве виртуальных вычислительных узлов.

Поскольку макет работает на виртуальных машинах, была доступна лишь небольшая часть входных данных: несколько видов загрузки процессора, load average и объем доступной памяти. Однако такое существенное отличие от первого этапа апробации позволило провести дополнительные проверки, в том числе проверку корректности работы правил в существенно отличных условиях.

В отличие от первого этапа, здесь все части прототипа настраивались с нуля, в том числе установка подсистемы мониторинга, импорт данных в подсистему анализа, настройка работы подсистемы анализа. Далее использовались более новые версии программного обеспечения (в том числе более новая версия менеджера ресурсов Slurm), а также специально менялись конфигурационные параметры (имена баз данных, логины, пароли и порты) в целях тестирования и проверки работоспособности прототипа при изменении настроек.

В процессе апробации были внесены некоторые дополнительные правки в исходный код, а также дополнена инструкция по установке. Отдельно было уделено внимание настройке и отработке переносимости “смысловой” части подсистемы анализа, непосредственно отвечающей за автоматическое обнаружение потенциальных проблем с производительностью в суперкомпьютерных приложениях. В результате этого была разработана детальная инструкция по настройке данной части.

5. Заключение. В статье представлен прототип программного комплекса для проведения мониторинга и анализа производительности суперкомпьютерных приложений. В качестве основных преимуществ создаваемого комплекса, основанного на разработанных ранее инструментах DiMMon и TASC, можно выделить:

- его переносимость — он разрабатывается специально для применения на разных суперкомпьютерных системах;
- гибкость настройки как организации сбора данных, так и проводимого анализа собранной информации;
- широкий функционал по автоматическому обнаружению проблем с производительностью в пользовательских приложениях.

Была проведена апробация комплекса на существующих данных и затем на тестовом макете, на основе которой был в существенной мере упрощен и задокументирован процесс его установки, а также внесены заметные изменения в его исходный код, позволившие сделать комплекс более переносимым.

В дальнейшем планируется направить основные усилия на подготовку дистрибутива для распространения разрабатываемого комплекса. Помимо этого, планируется расширение функционала данного комплекса, в частности за счет добавления методов гибкого и адаптивного анализа поведения и качества работы суперкомпьютера в целом, а также реализации более удобных и функциональных методов тестирования корректности работы комплекса.

Список литературы

1. High Performance Computing Market Size to Surpass USD 64.65 Bn by 2030. https://www.globenewswire.com/news-release/2022/04/04/2415844/0/en/High-Performance-Computing-Market-Size-to-Surpass-USD-64-65-Bn-by-2030.html. Дата обращения: 3 января 2023.
2. *Belkina Yu., Nikitenko D.* Computing cost and accounting challenges for octoshell management system // Proceedings of the 4th Ural Workshop on Parallel, Distributed, and Cloud Computing for Young Scientists (2018). CEUR Workshop Proceedings. 2018. **2281**. 146–158. http://ceur-ws.org/Vol-2281/paper-15.pdf.

2023, **24** (1), 24–36. doi 10.26089/NumMet.v24r103

3. *Nikitenko D.A., Shvets P.A., Voevodin V.V.* Why do users need to take care of their HPC applications efficiency? // Lobachevskii Journal of Mathematics. 2020. **41**, N 8. 1521–1532. doi 10.1134/s1995080220080132.
4. *Stefanov K.S., Pawar S., Ranjan A., Wandhekar S., Voevodin V.V.* A review of supercomputer performance monitoring systems // Supercomputing Frontiers and Innovations. 2021. **8**, N 3. 62–81. doi 10.14529/jsfi210304.
5. Performance Co-Pilot. http://pcp.io/. Дата обращения: 4 января 2023.
6. *Röhl T., Eitzinger J., Hager G., Wellein G.* LIKWID monitoring stack: a flexible framework enabling job specific performance monitoring for the masses // 2017 IEEE International Conference on Cluster Computing (CLUSTER). 2017. 781–784. doi 10.1109/CLUSTER.2017.115.
7. *Massie M.L., Chun B.N., Culler D.E.* The Ganglia distributed monitoring system: design, implementation, and experience // Parallel Computing. 2004. **30**, N 7. 817–840. doi 10.1016/j.parco.2004.04.001.
8. *Brandt J.M., Debusschere B.J., Gentile A.C., Mayo J.R., Pebay P.P., Thompson D., Wong M.H.* Ovis-2: a robust distributed architecture for scalable RAS // Proc. 2008 IEEE International Symposium on Parallel and Distributed Processing. 2008. 1–8. doi 10.1109/IPDPS.2008.4536549.
9. *Jones M.D., White J.P., Innus M., DeLeon R.L., Simakov N., Palmer J.T., Gallo S.M., Furlani T.R., Showerman M., Brunner R., Kot A., Bauer G., Bode B., Enos J., Kramer W.* Workload analysis of Blue Waters // arXiv preprint: 1703.00924v1 [cs.DC]. doi 10.48550/arXiv.1703.00924. Ithaca: Cornell Univ. Library, 2017.
10. *Simakov N.A., White J.P., DeLeon R.L., Gallo S.M., Jones M.D., Palmer J.T., Plessinger B., Furlani T.R.* A workload analysis of NSF's innovative HPC resources using XDMoD // arXiv preprint: 1801.04306v1 [cs.DC]. doi 10.48550/arXiv.1801.04306. Ithaca: Cornell Univ. Library, 2018.
11. *Hart D.L.* Measuring TeraGrid: workload characterization for a high-performance computing federation // The International Journal of High Performance Computing Applications. 2011. **25**, N 4. 451–465. doi 10.1177/1094342010394382.
12. *Gallo S.M., White J.P., DeLeon R.L., Furlani T.R., Ngo H., Patra A.K., Jones M.D., Palmer J.T., Simakov N., Sperhac J.M., Innus M., Yearke T., Rathsam R.* Analysis of XDMoD/SUPReMM data using machine learning techniques // Proc. 2015 IEEE International Conference on Cluster Computing. 2015. 642–649. doi 10.1109/CLUSTER.2015.114.
13. *Palmer J.T., Gallo S.M., Furlani T.R., Jones M.D., DeLeon R.L., White J.P., Simakov N., Patra A.K., Sperhac J., Yearke T., Rathsam R., Innus M., Cornelius C.D., Browne J.C., Barth W.L., Evans R.T.* Open XDMoD: a tool for the comprehensive management of high-performance computing resources // Computing in Science & Engineering. 2015. **17**, N 4. 52–62. doi 10.1109/MCSE.2015.68.
14. *Evans T., Barth W.L., Browne J.C., Deleon R.L., Furlani T.R., Gallo S.M., Jones M.D., Patra A.K.* Comprehensive resource use monitoring for HPC systems with TACC stats // Proceedings 1st International Workshop on HPC User Support Tools. 2014. 13–21. doi 10.1109/HUST.2014.7.
15. *Kostenetskiy P., Shamsutdinov A., Chulkevich R., Kozyrev V., Antonov D.* HPC TaskMaster — Task Efficiency Monitoring System for the Supercomputer Center // Communications in Computer and Information Science. Vol. 1618. Cham: Springer, 2022. 17–29. doi 10.1007/978-3-031-11623-0_2.
16. *Stefanov K., Voevodin Vl., Zhumatiy S., Voevodin Vad.* Dynamically reconfigurable distributed modular monitoring system for supercomputers (DiMMon) // Procedia Computer Science. 2015. **66**. 625–634. doi 10.1016/j.procs.2015.11.071.
17. *Shvets P., Voevodin V., Zhumatiy S.* HPC software for massive analysis of the parallel efficiency of applications // Communications in Computer and Information Science. Vol. 1063. Cham: Springer, 2019. 3–18. https://link.springer.com/chapter/10.1007/978-3-030-28163-2_1. Cited January 7, 2023.
18. *Shvets P., Voevodin V., Zhumatiy S.* Primary automatic analysis of the entire flow of supercomputer applications // Proceedings of the 4th Ural Workshop on Parallel, Distributed, and Cloud Computing for Young Scientists. 2018. 20–32. http://ceur-ws.org/Vol-2281/paper-03.pdf.
19. *Nikitenko D., Antonov A., Shvets P., Sobolev S., Stefanov K., Voevodin Vad., Voevodin Vl., Zhumatiy S.* JobDigest — detailed system monitoring-based supercomputer application behavior analysis // Communications in Computer and Information Science. Vol. 793. Cham: Springer, 2017. 516–529. doi 10.1007/978-3-319-71255-0_42.

Поступила в редакцию
07 ноября 2022 г.

Принята к публикации
21 декабря 2022 г.

2023, **24** (1), 24–36. doi 10.26089/NumMet.v24r103

Информация об авторах

Вадим Владимирович Воеводин — к.ф.-м.н., ст. научн. сотр.; Московский государственный университет имени М. В. Ломоносова, Научно-исследовательский вычислительный центр, Ленинские горы, 1, стр. 4, 119234, Москва, Российская Федерация.

Константин Сергеевич Стефанов — к.ф.-м.н., вед. спец.; Московский государственный университет имени М. В. Ломоносова, Научно-исследовательский вычислительный центр, Ленинские горы, 1, стр. 4, 119234, Москва, Российская Федерация.

References

1. High Performance Computing Market Size to Surpass USD 64.65 Bn by 2030. https://www.globenewswire.com/news-release/2022/04/04/2415844/0/en/High-Performance-Computing-Market-Size-to-Surpass-USD-64-65-Bn-by-2030.html. Cited January 3, 2023.
2. Yu. Belkina and D. Nikitenko, "Computing Cost and Accounting Challenges for Octoshell Management System," in *Proc. 4th Ural Workshop on Parallel, Distributed, and Cloud Computing for Young Scientists, Yekaterinburg, Russia, November 15, 2018*. CEUR Workshop Proc. **2281**, 146–158 (2018). http://ceur-ws.org/Vol-2281/paper-15.pdf.
3. D. A. Nikitenko, P. A. Shvets, and V. V. Voevodin, "Why Do Users Need to Take Care of Their HPC Applications Efficiency?," Lobachevskii J. Math. **41** (8), 1521–1532 (2020). doi 10.1134/s1995080220080132.
4. K. S. Stefanov, S. Pawar, A. Ranjan, et al., "A Review of Supercomputer Performance Monitoring Systems," Supercomput. Front. Innov. **8** (3), 62–81 (2021). doi 10.14529/jsfi210304.
5. Performance Co-Pilot. http://pcp.io/. Cited January 4, 2023.
6. T. Röhl, J. Eitzinger, G. Hager, and G. Wellein, "LIKWID Monitoring Stack: A Flexible Framework Enabling Job Specific Performance Monitoring for the Masses," in *Proc. 2017 IEEE Int. Conf. on Cluster Computing (CLUSTER), Honolulu, USA, September 5–8, 2017* (IEEE Press, New York, 2017), pp. 781–784. doi 10.1109/CLUSTER.2017.115.
7. M. L. Massie, B. N. Chun, and D. E. Culler, "The Ganglia Distributed Monitoring System: Design, Implementation, and Experience," Parallel Comput. **30** (7), 817–840 (2004). doi 10.1016/j.parco.2004.04.001.
8. J. M. Brandt, B. J. Debusschere, A. C. Gentile, et al., "Ovis-2: A Robust Distributed Architecture for Scalable RAS," in *Proc. IEEE Int. Symp. on Parallel and Distributed Processing, Miami, USA, April 14—18, 2008* (IEEE Press, New York, 2008), doi 10.1109/IPDPS.2008.4536549.
9. M. D. Jones, J. P. White, M. Innus, et al., *Workload Analysis of Blue Waters*, arXiv preprint: 1703.00924v1 [cs.DC] (Cornell Univ. Library, Ithaca, 2017). https://arxiv.org/abs/1703.00924. Cited January 4, 2023.
10. N. A. Simakov, J. P. White, R. L. DeLeon, et al., *A Workload Analysis of NSF's Innovative HPC Resources Using XDMoD*, arXiv preprint: 1801.04306v1 [cs.DC] (Cornell Univ. Library, Ithaca, 2018). https://arxiv.org/abs/1801.04306. Cited January 4, 2023.
11. D. L. Hart, "Measuring TeraGrid: Workload Characterization for a High-Performance Computing Federation," Int. J. High Perform. Comput. Appl. **25** (4), 451–465 (2011). doi 10.1177/1094342010394382.
12. S. M. Gallo, J. P. White, R. L. DeLeon, et al., "Analysis of XDMoD/SUPReMM Data Using Machine Learning Techniques," in *2015 IEEE Int. Conf. on Cluster Computing, Chicago, USA, September 8–11, 2015* (IEEE Press, New York, 2015), pp. 642–649. doi 10.1109/CLUSTER.2015.114.
13. J. T. Palmer, S. M. Gallo, T. R. Furlani, et al., "Open XDMoD: A Tool for the Comprehensive Management of High-Performance Computing Resources," Comput. Sci. Eng. **17** (4), 52–62 (2015). doi 10.1109/MCSE.2015.68.
14. T. Evans, W. L. Barth, J. C. Browne, et al., "Comprehensive Resource Use Monitoring for HPC Systems with TACC Stats," in *Proc. First Int. Workshop on HPC User Support Tools, New Orleans, USA, November 21–21, 2014* (IEEE Press, New York, 2014), pp. 13–21. doi 10.1109/HUST.2014.7.
15. P. Kostenetskiy, A. Shamsutdinov, R. Chulkevich, et al., "HPC TaskMaster — Task Efficiency Monitoring System for the Supercomputer Center," in *Communications in Computer and Information Science* (Springer, Cham, 2022), Vol. 1618, pp. 17–29. doi 10.1007/978-3-031-11623-0_2.
16. K. Stefanov, Vl. Voevodin, S. Zhumatiy, and Vad. Voevodin, "Dynamically Reconfigurable Distributed Modular Monitoring System for Supercomputers (DiMMon)," Procedia Comput. Sci. **66**, 625–634 (2015). doi 10.1016/j.procs.2015.11.071.

2023, **24** (1), 24–36. doi 10.26089/NumMet.v24r103

17. P. Shvets, V. Voevodin, and S. Zhumatiy, "HPC Software for Massive Analysis of the Parallel Efficiency of Applications," in *Communications in Computer and Information Science* (Springer, Cham, 2019), Vol. 1063, pp. 3–18. https://link.springer.com/chapter/10.1007/978-3-030-28163-2_1. Cited January 7, 2023.
18. P. Shvets, V. Voevodin, and S. Zhumatiy, "Primary Automatic Analysis of the Entire Flow of Supercomputer Applications," in *Proc. 4th Ural Workshop on Parallel, Distributed, and Cloud Computing for Young Scientists, Yekaterinburg, Russia, November 15, 2018.* CEUR Workshop Proc. **2281**, 20–32 (2018). http://ceur-ws.org/Vol-2281/paper-03.pdf.
19. D. Nikitenko, A. Antonov, P. Shvets, et al., "JobDigest — Detailed System Monitoring-Based Supercomputer Application Behavior Analysis," in *Communications in Computer and Information Science* (Springer, Cham, 2017), Vol. 793, pp. 516–529. doi 10.1007/978-3-319-71255-0_42.

Received
November 07, 2022

Accepted for publication
December 22, 2022

Information about the authors

Vadim V. Voevodin — Ph.D., Senior Researcher; Lomonosov Moscow State University, Research Computing Center, Leninskie Gory, 1, building 4, 119234, Moscow, Russia.

Konstantin S. Stefanov — Ph.D., Leading Specialist; Lomonosov Moscow State University, Research Computing Center, Leninskie Gory, 1, building 4, 119234, Moscow, Russia.