УДК 519.68
DOI:10.25729/ESI.2024.33.1.002

Применение методов интеллектуального анализа данных для изучения свойств суперкомпьютерных приложений

Шайхисламов Денис Ильгизович, Воеводин Вадим Владимирович
Научно-исследовательский вычислительный центр МГУ имени М.В. Ломоносова,
Россия, Москва, *sdenis1995@gmal.com*

Аннотация. Для эффективного использования ресурсов суперкомпьютера необходимо постоянно анализировать различные аспекты качества работы современных высокопроизводительных систем. Одним из наиболее важных аспектов является эффективность выполнения параллельных приложений, работающих на суперкомпьютере. А для того, чтобы изучать данный аспект, зачастую полезно иметь информацию о том, насколько различные приложения схожи между собой. Ранее нами были предложены два подхода к сравнению суперкомпьютерных приложений: на основе статической информации об исполняемых файлах, а также динамики их работы во время исполнения. В данной работе будут показаны два полезных на практике метода применения этих подходов: кластеризация заданий и предсказание метрик оценок качества использования суперкомпьютерных ресурсов. С помощью кластеризации будет показано, как можно выявлять аномальные группы запусков заданий, например, в рамках всего потока суперкомпьютерных приложений либо в рамках запусков одного пользователя. С помощью предсказания метрик оценок качества использования суперкомпьютерных ресурсов будет показано, как, минимизируя влияние на запускаемые приложения, собирать статистику по эффективности выполнения пользовательских приложений. Данные методы были успешно апробированы на суперкомпьютере петафлопсного уровня производительности Ломоносов-2.

Ключевые слова: высокопроизводительные вычисления, эффективность приложений, суперкомпьютерный центр, аномальные запуски, интеллектуальный анализ данных

Цитирование: Шайхисламов Д.И. Применение методов интеллектуального анализа данных для изучения свойств суперкомпьютерных приложений / Д.И. Шайхисламов, В.В. Воеводин // Информационные и математические технологии в науке и управлении. – 2024. – № 1(33). – С. 20-30. – DOI:10.25729/ESI.2024.33.1.002.

Введение. Современные суперкомпьютеры требуют постоянного мониторинга и контроля качества их работы, без которого практически невозможно достичь эффективного использования суперкомпьютерных ресурсов. При этом необходимо анализировать самые разные аспекты качества работы таких систем: эффективность работы менеджера ресурсов, корректность конфигурации системного и прикладного программного обеспечения, оптимальность установленных квот и ограничений на использование ресурсов, адекватность выбранных разделов и правил их использования, а также многое другое.

Одним из самых главных таких аспектов является эффективность пользовательских приложений, выполняющихся на суперкомпьютере. При этом вопрос изучения потока выполняющихся приложений можно решать самыми разными способами. Например, в работах [1, 2] авторы используют данные системы мониторинга для выделения как аномалий в работе суперкомпьютера, так и трендов использования ресурсов приложениями с помощью методов интеллектуального анализа данных. В работе [3] авторы с помощью классификатора Random Forest размечают задания как интенсивные с точки зрения потребления энергии, так и по интенсивности использования каналов ввода-вывода. Если интенсивные задания по энергопотреблению или каналу ввод-вывода будут исполняться физически рядом на суперкомпьютере, то будет большая нагрузка на блоки питания или каналы ввода-вывода, из-за чего наблюдается нестабильная работа системы, и решение на основе методов классификация дает им возможность заранее оптимальнее распределить задания по вычислительной системе. Авторы в [4] предлагают использовать методы регрессии для предсказания времени исполнения приложений. Они использовали информацию, доступную при запуске заданий, и обучили регрессор

Decision Tree на заданиях за 8 лет работы суперкомпьютера “Beocat”. Результаты авторы использовали для более оптимального планирования постановки заданий на исполнения в менеджере ресурсов Slurm, что позволило им уменьшить среднее время ожидания заданий в очереди.

Еще одним из способов анализа потока приложений заключается в поиске и анализе схожих приложений. Обнаружение схожих приложений в общем потоке помогает, например, более быстро и полно анализировать свойства различных приложений, а также предсказывать поведение еще выполняющихся заданий. Например, в [5] авторы используют методы кластеризации для нахождения близких к исследуемому заданию групп, которые в дальнейшем используются для предсказания времени исполнения приложения. Но для выделения схожих авторы используют методы сравнения команд запуска заданий, что в нашем случае не является достаточно точной метрикой схожести. В работе [6] авторы анализируют данные системы мониторинга и применяют методы выделения схожих заданий с целью определения интенсивности использования каналов ввода-вывода. Перед сравнением временные ряды из системы мониторинга проходят стадию квантования, и в дальнейшем сравниваются с помощью Евклидова расстояния. Данный подход для нас также не подходит, так как несмотря на то, что процесс квантования позволяет очень быстро сравнивать временные ряды, при этом теряется очень много информации о поведении задания.

Ранее нами были предложены два метода поиска схожих суперкомпьютерных приложений, основанные на технологиях интеллектуального анализа данных [7]. Первый метод предполагает анализ так называемых «статических» данных – тех данных, которые не изменяются во время исполнения программы, как, например, исходный код программы, входные данные, исполняемые и объектные файлы и т.д. В нашем случае мы анализируем исполняемые и объектные файлы. С помощью Linux утилиты nm мы можем извлечь используемые в программе имена функций и переменных, которые в дальнейшем с помощью нейронной сети Doc2Vec [8] преобразуются в вектор фиксированной длины. Полученные вектора сравниваются с помощью косинусного сходства (cosine similarity), и результатом является оценка расстояния между двумя заданиями. Данный метод показал высокое качество работы и применяется на постоянной основе на суперкомпьютере Ломоносов-2 для выявления использования различных программных пакетов в заданиях [9, 10].

Второй метод основан на анализе «динамических» данных – данных, которые меняются во время исполнения задания. В нашем случае производится анализ данных системы мониторинга, которая предоставляет информацию о различных аппаратных датчиках во время исполнения приложения. Примерами датчиков являются загрузка CPU/GPU, интенсивность использования сети Infiniband, количество промахов в кэш-память первого и третьего уровней и т.д. Данные агрегируются каждую минуту, и каждый датчик формирует временной ряд. Сравнение полученных временных рядов дает возможность получить оценку схожести поведения у различных заданий. Стоит отметить, что часто применяемое в подобных случаях Евклидово расстояние не подходит из-за частых временных сдвигов фаз исполнения приложения. Пример схожих заданий по поведению с временными сдвигами показан на рис. 1. Поэтому в нашем случае для получения оценки расстояния мы используем метод Dynamic Time Warping [11], который позволяет учесть эту особенность и получить оптимальное соответствие между точками двух временных рядов. При оценке качества работы динамического метода была получена высокая точность (0.9 на метрике Rand Index) на задаче выявления использования программных пакетов [9].

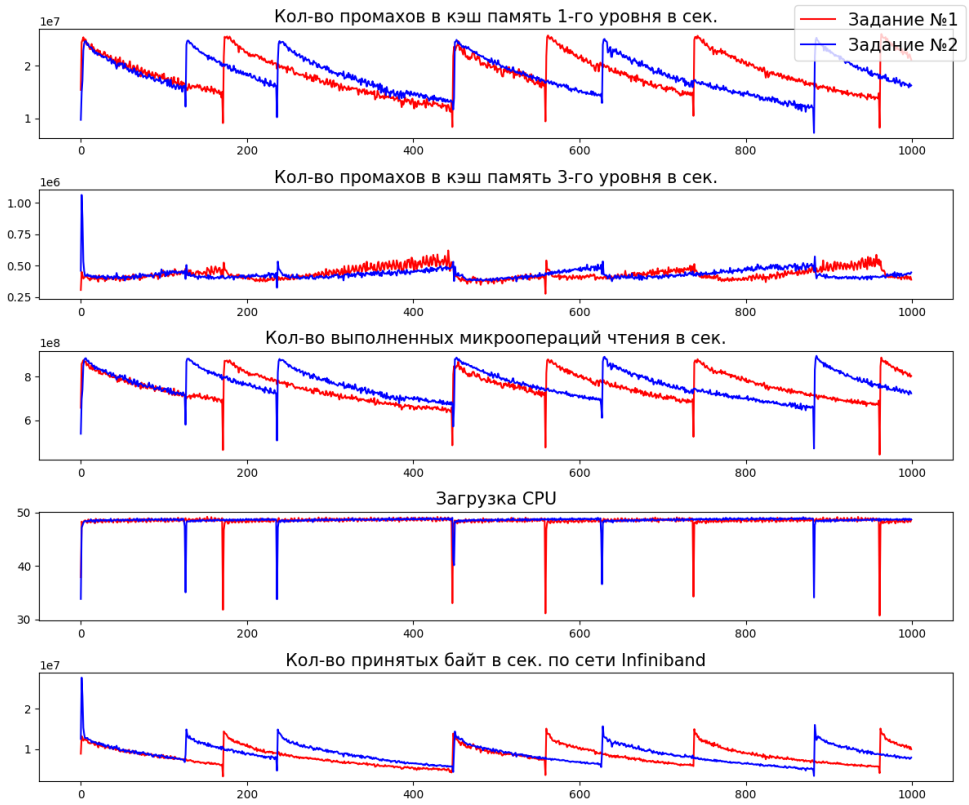

Рис. 1. Показатели динамических характеристик для 2-х схожих заданий во время их исполнения

В данной работе показаны две практически важные задачи, для решения которых предложены подходы с применением указанных методов поиска схожих приложений. Первая задача заключается в кластеризации пользовательских приложений с целью выявления аномальных запусков. Вторая задача направлена на предсказание оценок качества использования суперкомпьютерных ресурсов. Предложенные подходы были апробированы на суперкомпьютере петафлопсного уровня производительности Ломоносов-2, и проведенная апробация показала применимость этих подходов на практике и высокую точность их работы.

Далее статья устроена следующим образом. Раздел 1 описывает предложенный в данной работе метод выявления аномальных запусков на основе кластеризации заданий. В Разделе 2 показано, каким образом методы поиска схожих приложений могут быть применены для предсказания оценок качества использования суперкомпьютерных ресурсов. В Заключении кратко сформулированы основные результаты, представленные в данной работе.

1. Кластеризация суперкомпьютерных заданий.

1.1. Описание подхода к кластеризации заданий. Кластеризация является популярным методом анализа, поскольку ее результаты легко интерпретируются, и она предоставляет информацию о естественной группировке в данных. В нашем случае представляет интерес кластеризация пользовательских заданий, что позволяет лучше понимать общую структуру потока заданий, выполняющихся на суперкомпьютере. Также результаты кластеризации можно использовать в качестве дополнительной информации для других видов анализа, например, для обнаружения аномалий. Ранее нами был предложен алгоритм кластеризации заданий с использованием упомянутых во введении статического и динамического методов выделения схожих приложений [12]. Алгоритм кластеризации устроен следующим образом (схема показана на рис. 2): сначала выполняется кластеризация заданий по исполняемым файлам с помощью иерархической кластеризации, которая использует статический метод оценки расстояния между заданиями. Используется предположение, что задания будут схожи по поведению

только если у них схожи исполняемые файлы. Конечно, есть исключения из такого правила, но на практике они встречаются очень редко. Затем выполняется кластеризация заданий по поведению в рамках каждого полученного кластера по исполняемым файлам с помощью иерархической кластеризации, которая использует динамический метод оценки расстояния между заданиями.

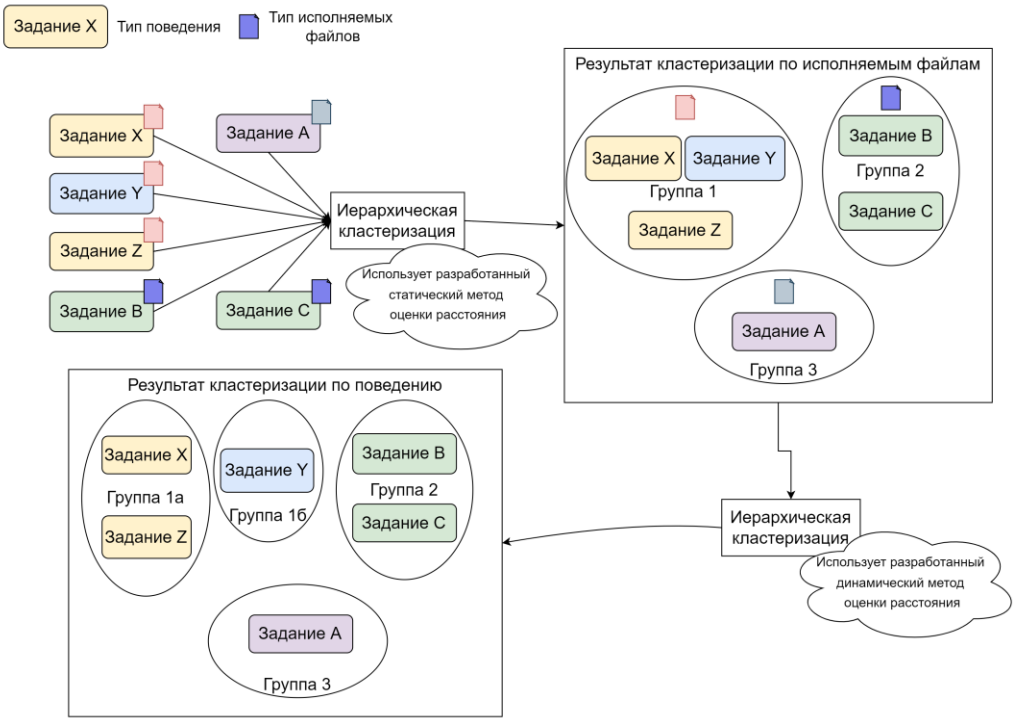

Рис. 2. Предлагаемый алгоритм кластеризации заданий

Предложенный метод уже использовался для выделения групп заданий, которые занимают существенную долю от общего числа использованных CPU-часов суперкомпьютерного центра, а также выделения новых версий используемых программных пакетов [5]. В данной работе предлагается использовать результат кластеризации для другой цели – выделения аномалий в сериях запусков.

1.2. Выделение аномалий в сериях запусков. Аномалиями в сериях запусков будем называть такие задания, которые в последовательности запусков заданий пользователя длиной N сильно отличаются по поведению от всех других заданий в данной последовательности. Число N может варьироваться в зависимости от строгости выделения таких аномалий: если оно большое, то аномалии практически не будут выделяться, при очень маленьком его значении будут выделяться очень много лишних, не аномальных заданий. Для нашей системы эмпирически было выявлено, что при N=9 получаются лучшие результаты.

Такие аномалии предоставляют интерес для пользователей с точки зрения ожидаемой производительности работы их приложения, а для администраторов – с точки зрения эффективного использования ресурсов. Причин таких аномалий может быть много – изменение параметров запуска и/или входных данных, что приводит к другому поведению приложения, аппаратный/программный сбой и т.д. Простые случаи, как, например, аппаратный сбой графической карты или проблемы с сетевой файловой системой Lustre, можно легко отловить с помощью пороговых значений для датчиков из системы мониторинга, но очень тяжело отловить случаи, когда некорректность поведения задания видна только при рассмотрения серии запусков, т.е. ее сложно или даже невозможно обнаружить при анализе одного приложения вне контекста остальных запусков.

Для выделения аномалий в сериях запусков предлагается использовать результат работы вышеописанного метода кластеризации заданий. Идея метода заключается в том, что если в сериях запусков большинство заданий принадлежит одному кластеру, но среди них встречается задание из другого кластера, то его можно считать аномальным. Мы понимаем, что могут быть случаи, когда пользователь запустил то же самое приложение с сильно отличающимися входными параметрами или же совсем другое приложение, что привело к возникновению аномалии, но на практике такое возникает не часто. И такие запуски все равно есть смысл отлавливать, поскольку эта информация может быть также полезна для пользователя в некоторых случаях. Для демонстрации приведем следующий пример запусков заданий (рис. 3), где задания раскрашены по тому, к какому кластеру они принадлежат. Видно, что большая часть последовательности запусков заданий принадлежат зеленому кластеру, но среди них появляется задание из красного кластера – значит считаем задание Job7 аномальным. Если же задания из других кластеров (синий и желтый) находятся на границе последовательности запусков кластера, то их не стоит считать аномальными (поскольку нет достаточной информации о соседних с ними запусках).

$Job1, Job2, Job3, Job4, Job5, Job6, Job7, Job8, Job9, Job10, Job11, Job12$

Рис. 3. Пример последовательности запусков заданий пользователя

1.3. Апробация предложенного подхода. Для проверки работоспособности метода были проанализированы все задания суперкомпьютера Ломоносов-2 за январь-май 2023 года, и с помощью данного метода были выделено 248 аномальных случаев. Это составляет 1.7% от общего числа заданий. Но для подтверждения того, что данные задания действительно аномальные, все выявленные случаи были вручную проверены. В результате проверки было выявлено, что:

- выделенные задания действительно выбиваются по поведению в сериях запусков заданий пользователя;
- команды запуска аномальных заданий и заданий в его окрестности отличаются незначительно – значит запускались однотипные приложения с небольшими изменениями во входных параметрах;
- поведение в большинстве случаев нетривиальное, то есть без учета контекста запусков данное задание не выделяется как подозрительное.

Это показывает, что предложенный метод действительно можно использовать для выявления аномалий в сериях запусков. Но стоит отметить, что далеко не всегда такие выявленные аномальные задания работают неэффективно, ведь даже небольшое изменение параметров запуска может сильно влиять на поведение задания, и это может быть абсолютно нормальным свойством данного типа приложений. Поэтому было принято решение, что, когда данный вид анализа будет запущен для обработки потока суперкомпьютерных заданий в реальном времени, мы не будем оповещать всех пользователей о найденных аномальных случаях. Если пользователю данный вид анализа будет интересен, будет предоставлена возможность включить обнаружение таких аномалий среди их запусков и настроить оповещение. Подобный функционал планируется реализовать в рамках системы Октошелл [13], в котором, среди прочего, пользователям уже доступна детальная информация о производительности их приложений.

2. Предсказание качества использования ресурсов суперкомпьютера. Администраторами Суперкомпьютерного центра МГУ в данный момент разрабатывается система оценок для запускаемых приложений, которая предназначена для быстрого и точного анализа каче-

ства использования ресурсов суперкомпьютера. Эти оценки нужны для первоначального анализа, позволяющего понять в целом, какие задания имеют низкую эффективность и поэтому требуют внимания. Предполагается, что последующий детальный анализ выбранного приложения, необходимый для определения первопричин снижения производительности и способов их устранения, должен выполняться с использованием других существующих средств анализа, таких как профилировщики, отладчики и т.д.

Оценки были разработаны для 6 типов ресурсов: CPU (score_cpu), памяти (score_mem), сети MPI (score_mpi), ввода-вывода (score_io), GPU (score_gpu) и памяти GPU (score_gpu_mem). Каждая из характеристик принимает значение от 0 (в задании нет проблем с данным видом ресурса) до 100 (работа с данным видом ресурса постоянна мешала полезным вычислениям во время работы анализируемого задания). В данной работе мы остановимся на двух оценках – для CPU и памяти. Для вычисления данных оценок требуется собирать достаточно много информации с процессорных датчиков, что требует применения режима мультиплексирования при работе системы мониторинга. Данный режим добавляет дополнительные накладные расходы, что в среднем увеличивает время исполнения на ~2.78% [14]. Такие накладные расходы, хотя и не являются критичными, все же ощутимы, поэтому на данный момент принято решение запускать сбор данных по дополнительным датчикам только для каждого третьего задания, что позволяет в среднем уменьшить накладные расходы до <1%. Но возникает вопрос – как получить данные оценки для остальных 2/3 заданий?

2.1. Описание подхода к предсказанию оценок. Для решения данной задачи было принято решение использовать методы выделения схожих суперкомпьютерных заданий, описанные во введении. Был предложен следующий алгоритм:

1. для 1/3 заданий есть оценки качества использования ресурсов, таким образом, мы их можем использовать как исторические данные в базе знаний;
2. при появлении задания, у которого нет оценок:
   a. производится поиск всех заданий из базы знаний, которые схожи согласно статическому методу;
   b. если находятся задания, схожие по статическим данным, то среди них ведется поиск заданий, схожих по динамическим данным, с менее строгим порогом (при этом выделяется больше схожих заданий);
   c. если заданий, схожих по статическим данным, не находится, то ведется поиск схожих по динамическим данным заданий из базы знаний с более строгим порогом (при этом выделяется меньше схожих заданий);
   d. проводится агрегация оценок найденных схожих по динамическим данным заданий.

Агрегацию оценок можно проводить различными способами. Мы испробовали три различных варианта агрегации: среднее значение, медиана и средневзвешенное значение softmax. В ходе экспериментов было выявлено, что при использовании различных функций агрегации результаты работы метода варьировались несущественно, из-за чего в дальнейшем всегда использовалась медиана.

2.2. Апробация метода предсказания оценок. Апробация проводилась с помощью заданий, у которых есть оценка, поскольку это дает возможность предсказать значение оценки по историческим данным, а потом сравнить результат с реальным значением.

Для оценки качества работы метода использовались метрики Mean Absolute Error, MAE (1) и Mean Squared Error, MSE (2), где j – номер задания, n – количество заданий, $\boldsymbol{y_j}$ – предсказанное значение, $\widehat{\boldsymbol{y_j}}$ **–** истинное значение оценки. Стоит отметить, что отклонение от реальной оценки ±10 в нашем случае является допустимым, так как оценки нужны для первичного качественного анализа производительности, и высокая точность здесь не требуется. Более того,

сами оценки тоже могут меняться в данном промежутке даже при идентичных запусках в силу особенностей их поведения.

$$\mathrm{MAE} = \frac{1}{n}\sum_{j=1}^{n}\left|y_j - \hat{y_j}\right| \quad (1)$$

$$\mathrm{MSE} = \frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y_j})^2 \quad (2)$$

Для апробации были рассмотрены все задания за 6 месяцев работы суперкомпьютера Ломоносов-2. Мы рассматривали только задания, длительность которых была больше получаса, так как при меньшей длительности собирается недостаточно данных мониторинга для оценки поведения. Общее количество заданий с оценками составило ~4600.

При апробации необходимо было рассмотреть 2 варианта использования метода. Первый вариант – для получения наилучших результатов, и в этом случае использовались все доступные данные. Второй вариант – это симуляция онлайн работы алгоритма, то есть при предсказании оценок сразу после завершения задания, из-за чего доступны данные только о предшествующих заданиях. Начнем с первого варианта, так как с его помощью можно получить наиболее точную картину о том, насколько хорошо работает метод. В таблице 1 приведены результаты работы метода при доступности всех данных, где AE – абсолютное значение ошибки. Как можно заметить, в среднем работа метода показывает удовлетворяющую требованиям качество работы: средняя ошибка составляет 2.62. Но не для всех заданий можно предсказать оценки – для ~13% случаев методы выделения схожих заданий не смогли найти близкие задания для предсказаний. Это означает, что данные задания сильно отличаются по поведению от всех заданий, доступных на данный момент. График распределения разности между предсказанными и истинными значениями оценок показан на рис. 4, по которому видно, что у преимущественного числа заданий (>95%) ошибка составляет <10, что полностью удовлетворяет нашим потребностям.

Таблица 1. Результат апробации для случая с использованием всех заданий

	% заданий, для которых удалось предсказать значение	MAE	MSE	% заданий с AE>10
Score_mem	86.94	2.62	34.54	4.71
Score_cpu	86.93	2.19	23.21	3.27

Первый вариант – это идеальный случай, когда время обработки задания не учитывается. Второй вариант является более приближенным к реальности, так как зачастую есть потребность в получении данных о задании сразу после его завершения.

При обработке потока суперкомпьютерных заданий в режиме реального времени искать схожие задания среди всех предыдущих заданий не является возможным, поскольку это может потребовать слишком много времени. В связи с чем возникает вопрос: сколько предыдущих заданий необходимо учитывать? Это количество заданий для сравнения в дальнейшем будем называть окном.

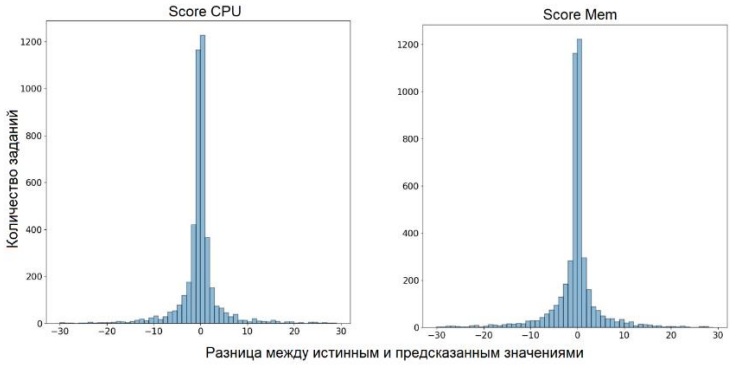

Рис. 4. Гистограмма распределения числа заданий с полученной разницей между истинным и предсказанным значениями для оценок score_cpu и score_mem

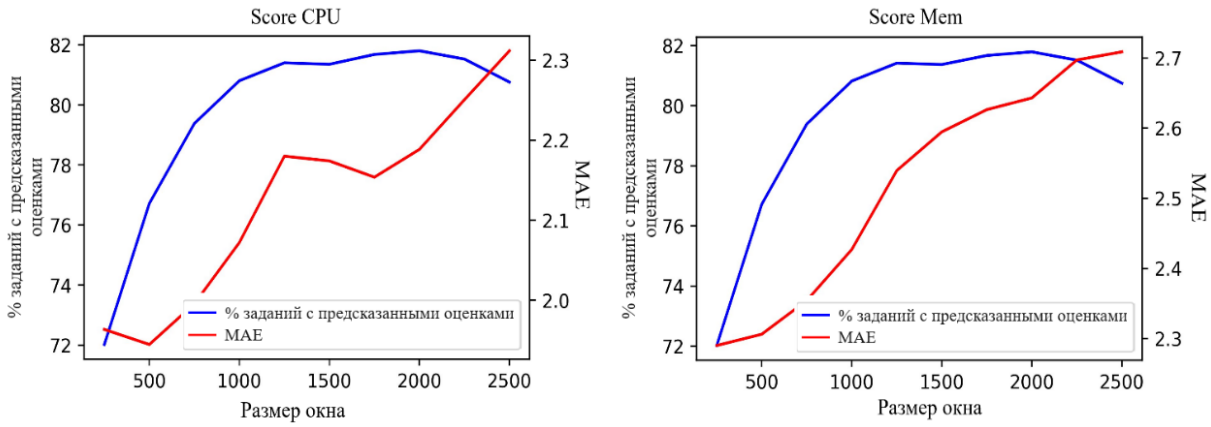

Рис. 5. Графики изменения MAE и процента заданий, для которых удалось предсказать значение оценки, в зависимости от размера окна

Для решения задачи подбора размера окна был проведен эксперимент, в котором оценивалось качество работы метода в зависимости от размера окна. На рис. 5 показаны графики изменения MAE и процента заданий, для которых удалось предсказать значение оценки, в зависимости от размера окна. Как можно заметить, размер окна от 1250 до 1500 кажется лучшим выбором для комбинации метрик, так как последующие размеры практически не дают никакого прироста к качеству работы: количество заданий с предсказаниями не растет, но при этом MAE также немного, но все же увеличивается.

В таблице 2 приведены результаты работы метода при размере окна, равным 1250.

Если сравнивать два варианта апробации, то можно заметить, что учет не только предшествующих, но и будущих заданий не дает никакого прироста к качеству предсказаний, но дает возможность предсказать оценки для еще 5% заданий, из-за чего планируется использовать следующий подход при анализе реального потока суперкомпьютерных заданий:

– сразу после завершения задания выполняется предсказание оценки с использованием второго варианта апробации;
– если предсказать оценку не удалось, через N дней проводится перепроверка с учетом новых завершившихся заданий. Число N может варьироваться.

Таблица 2. Результат апробации для случая с использованием только предшествующих заданий, размер окна=1250

	% заданий, для которых удалось предсказать значение	MAE	MSE	% заданий с AE>10
Score_mem	81.41	2.53	32.59	4.51
Score_cpu	81.39	2.18	24.83	3.68

Заключение. В данной работе показаны две важные практические задачи, для решения которых могут применяться разработанные ранее методы выделения схожих приложений – выявление аномалий в сериях запусков и предсказание оценок качества использования ресурсов суперкомпьютера.

В задаче выявления аномалий применялся подход с использованием методов кластеризации заданий по динамическим данным. Если в некоторой последовательности запусков все задания принадлежат одному кластеру, но среди них присутствует задание из другого кластера, то это задание считается аномальным. С помощью такого подхода были проанализированы задания за 5 месяцев работы суперкомпьютера Ломоносов-2, и среди них обнаружено 1.7% потенциально аномальных заданий. Ручная проверка показала, что действительно эти задания заметно отличаются по поведению от серии запусков заданий пользователей и таким образом могут представлять интерес для обнаружения.

Оценки качества использования ресурсов суперкомпьютера на данный момент собираются только для трети заданий, из-за чего для остальных необходимо проводить предсказание. Предлагаемый метод предсказания оценок основан на поиске схожих заданий, для которых известна оценка. После получения списка близких заданий, их оценки агрегируются, что используется как результат предсказания. Апробация показала, что с помощью данного метода можно предсказать оценки для 87% заданий, и качество предсказаний достаточно высокое. Так, средняя ошибка составляет всего 2.63, а процент заданий, у которых отклонение предсказанной оценки от реальной существенно, не более 5%.

На данный момент программные средства для решения этих задач прошли апробацию и в будущем будут использоваться на суперкомпьютере Ломоносов-2 на постоянной основе.

Благодарности. Результаты, описанные в данной статье, были получены в МГУ имени М.В. Ломоносова при финансовой поддержке Российского научного фонда, договор № 21-71-30003. Работа выполнена с использованием оборудования Центра коллективного пользования сверхвысокопроизводительными вычислительными ресурсами МГУ имени М.В. Ломоносова.

Список источников

1. Joseph 'Joshi' Fullop IV, Brett L. Layman Deriving workload expectations - monitoring and analysis using HPC job profiles. CUG2020 Proceedings, 2020, available at: https://cug.org/proceedings/cug2020_proceedings/includes/files/spec110s1.pdf (accessed: 20/08/2023).
2. Patel T., Liu Z., Kettimuthu R., Rich P., Allcock W, Tiwari D. Job characteristics on large-scale systems: long-term analysis, quantification, and implications. SC20, International conference for high performance computing, networking, storage and analysis, 2020, pp. 1-17.
3. Tsujita Y., Uno A., Sekizawa R., Yamamoto K. Classifying jobs towards power-aware HPC system operation through long-term log analysis, array 15, 2022, pp. 100179, available at: https://www.sciencedirect.com/science/article/pii/S2590005622000376 (accessed: 20/08/2023).
4. Tanash M., Dunn B., Andresen D., Hsu W., Yang H., Okanlawon A., Improving HPC system performance by predicting job resources via supervised machine learning. PEARC19, 2019, available at: https://doi.org/10.1145/3332186.3333041 (accessed: 20/08/2023).
5. Wang H., Dai YQ., Yu J. [et al.] Predicting running time of aerodynamic jobs in HPC system by combining supervised and unsupervised learning method. Advances in Aerodynamics, 3, 22, 2021, available at:https://doi.org/10.1186/s42774-021-00077-8 (accessed: 20/08/2023).
6. Kunkel J., Betke E. Toward a workflow for identifying jobs with similar i/o behavior utilizing time series analysis, high performance computing. ISC high performance 2021, lecture notes in Computer Science, vol 12761.
7. Shaikhislamov D., Voevodin V. Solving the problem of detecting similar supercomputer applications using machine learning methods. Parallel computational technologies 2020, vol. 1263 of communications in computer and information science, New York, Springer International Publishing, 2020, pp. 46–57.
8. Quoc V. Le, Mikolov T. Distributed representations of sentences and documents. ArXiv.org. 2014, available at: DOI:10.48550/arXiv.1405.4053 (accessed: 20/08/2023).
9. Shaikhislamov D., Voevodin V. Development and practical application of methods for detecting similar supercomputer jobs. Parallel computational technologies, PCT 2021, vol. 1437 of communications in computer and information science, Cham, Switzerland, Springer International Publishing AG, 2021, pp. 18-30.
10. Shaikhislamov D., Voevodin V. Analysis of software package usage based on methods for identifying similar HPC applications. Russian supercomputing days, RuSCDays 2021, ed. Voevodin V., Sobolev S., vol. 1510 of communications in computer and information science, Springer International Publishing, Cham, 2021, pp. 310-321.
11. Berndt D.J., Clifford J. Using dynamic time warping to find patterns in time series. Proceedings of the 3rd international conference on knowledge discovery and data mining, AAAIWS'94, AAAI Press, pp. 359-370.
12. Shaikhislamov D., Voevodin V. Smart clustering of hpc applications using similar job detection methods. Parallel processing and applied mathematics, vol. 13826 of lecture notes in computer science, New York, Springer International Publishing, 2023, pp. 209-221.
13. Nikitenko D.A., Voevodin V.V., Zhumatiy S.A. Driving a Petascale HPC center with octoshell management system. Lobachevskii J Math 40, 2019, pp. 1817-1830.
14. Voevodin V., Stefanov K., Zhumatiy S. Overhead analysis for performance monitoring counters multiplexing. 8th Russian supercomputing days, RuSCDays 2022, Moscow, Russia, vol. 13708 of lecture notes in computer science, Cham, Switzerland, Springer International Publishing AG, 2022, pp. 461-474.

Шайхисламов Денис Ильгизович. *Техник научно-исследовательского вычислительного центра МГУ им. М.В.Ломоносова, ведущий инженер в компании Хуавэй. Основные направления научной деятельности: анализ производительности приложений, большие данные, вычислительные системы, системы обработки данных. SPIN: 9153-5619, ORCID: 0000-0002-9279-6397, Scopus AuthorID: 57193691627, sdenis1995@gmail.com.*

Воеводин Вадим Владимирович. *К.ф.м.н., заведующий лабораторией научно-исследовательского вычислительного центра МГУ им. М.В.Ломоносова. Основные направления научной деятельности: параллельные вычисления, суперкомпьютеры, технологии параллельного программирования, системное программное обеспечение, анализ производительности, поиск аномалий, анализ потока заданий, мониторинг, эффективность. ResearcherID: F-1783-2017, SPIN: 3588-3465, ORCID: 0000-0003-1897-1828, Scopus AuthorID: 57188745295.*

UDC 519.68
DOI:10.25729/ESI.2024.33.1.002

Application of data mining methods to study the properties of supercomputing applications

Denis I. Shaikhislamov, Vadim V. Voevodin

Research computing center of Lomonosov Moscow State University,
Russia, Moscow, *sdenis1995@gmail.com*

Abstract. For efficient use of supercomputer resources, it is necessary to constantly analyze various aspects of the quality of modern high-performance systems. One of the most important aspects is the efficiency of execution of parallel applications running on a supercomputer. And in to study this aspect, it is often useful to have information about how different applications are similar to each other. Previously, we proposed two approaches to comparing applications: based on static information about executable files, as well as the behavior during execution. In this paper, we will show two practical methods for applying these approaches: clustering and predicting metrics for assessing the quality of the use of supercomputer resources. Using clustering, we will show how abnormal groups of job launches can be detected, for example, within the entire flow of supercomputing applications or within the launches of a single user. Using the prediction of metrics for assessing the quality of use of supercomputer resources, it will be shown how, while minimizing the impact on running applications, to collect statistics on the efficiency of user applications. These methods were successfully tested on a petaflop supercomputer Lomonosov-2.

Keywords: high performance computing, application efficiency, supercomputing center, abnormal launches, data mining

Acknowledgements: The results described in this article were obtained at Lomonosov Moscow State University with the financial support of the Russian Science Foundation, agreement No. 21-71-30003. The research is carried out using the equipment of the shared research facilities of HPC computing resources at Lomonosov Moscow State University.

References

1. Joseph 'Joshi' Fullop IV, Brett L. Layman Deriving workload expectations - monitoring and analysis using HPC job profiles. CUG2020 Proceedings, 2020, available at: https://cug.org/proceedings/cug2020_proceedings/includes/files/spec110s1.pdf (accessed: 20/08/2023).
2. Patel T., Liu Z., Kettimuthu R., Rich P., Allcock W, Tiwari D. Job characteristics on large-scale systems: long-term analysis, quantification, and implications. SC20, International conference for high performance computing, networking, storage and analysis, 2020, pp. 1-17.
3. Tsujita Y., Uno A., Sekizawa R., Yamamoto K. Classifying jobs towards power-aware HPC system operation through long-term log analysis, array 15, 2022, pp. 100179, available at: https://www.sciencedirect.com/science/article/pii/S2590005622000376 (accessed: 20/08/2023).

4. Tanash M., Dunn B., Andresen D., Hsu W., Yang H., Okanlawon A., Improving HPC system performance by predicting job resources via supervised machine learning. PEARC19, 2019, available at: https://doi.org/10.1145/3332186.3333041 (accessed: 20/08/2023).
5. Wang H., Dai YQ., Yu J. [et al.] Predicting running time of aerodynamic jobs in HPC system by combining supervised and unsupervised learning method. Advances in Aerodynamics, 3, 22, 2021, available at:https://doi.org/10.1186/s42774-021-00077-8 (accessed: 20/08/2023).
6. Kunkel J., Betke E. Toward a workflow for identifying jobs with similar i/o behavior utilizing time series analysis, high performance computing. ISC high performance 2021, lecture notes in Computer Science, vol 12761.
7. Shaikhislamov D., Voevodin V. Solving the problem of detecting similar supercomputer applications using machine learning methods. Parallel computational technologies 2020, vol. 1263 of communications in computer and information science, New York, Springer International Publishing, 2020, pp. 46–57.
8. Quoc V. Le, Mikolov T. Distributed representations of sentences and documents. ArXiv.org. 2014, available at: DOI:10.48550/arXiv.1405.4053 (accessed: 20/08/2023).
9. Shaikhislamov D., Voevodin V. Development and practical application of methods for detecting similar supercomputer jobs. Parallel computational technologies, PCT 2021, vol. 1437 of communications in computer and information science, Cham, Switzerland, Springer International Publishing AG, 2021, pp. 18-30.
10. Shaikhislamov D., Voevodin V. Analysis of software package usage based on methods for identifying similar HPC applications. Russian supercomputing days, RuSCDays 2021, ed. Voevodin V., Sobolev S., vol. 1510 of communications in computer and information science, Springer International Publishing, Cham, 2021, pp. 310-321.
11. Berndt D.J., Clifford J. Using dynamic time warping to find patterns in time series. Proceedings of the 3rd international conference on knowledge discovery and data mining, AAAIWS'94, AAAI Press, pp. 359-370.
12. Shaikhislamov D., Voevodin V. Smart clustering of hpc applications using similar job detection methods. Parallel processing and applied mathematics, vol. 13826 of lecture notes in computer science, New York, Springer International Publishing, 2023, pp. 209-221.
13. Nikitenko D.A., Voevodin V.V., Zhumatiy S.A. Driving a Petascale HPC center with octoshell management system. Lobachevskii J Math 40, 2019, pp. 1817-1830.
14. Voevodin V., Stefanov K., Zhumatiy S. Overhead analysis for performance monitoring counters multiplexing. 8th Russian supercomputing days, RuSCDays 2022, Moscow, Russia, vol. 13708 of lecture notes in computer science, Cham, Switzerland, Springer International Publishing AG, 2022, pp. 461-474.

Shaikhislamov Denis Ilgizovich. *Technician at research computing center of Lomonosov Moscow State University, Senior engineer at Huawei RRI. The main areas of scientific activity: application performance analysis, big data, computing systems, data processing systems. SPIN: 9153-5619, ORCID: 0000-0002-9279-6397, Scopus AuthorID: 57193691627, sdenis1995@gmail.com.*

Voevodin Vadim Vladimirovich. *PhD, head of the laboratory in the research computing center of Lomonosov Moscow State University. The main areas of scientific activity: parallel computing, supercomputers, parallel programming technologies, system software, performance analysis, anomaly search, job flow analysis, monitoring, efficiency. ResearcherID: F-1783-2017, SPIN: 3588-3465, ORCID: 0000-0003-1897-1828, Scopus AuthorID: 57188745295.*

Статья поступила в редакцию 04.09.2023; одобрена после рецензирования 26.10.2023; принята к публикации 26.10.2023.
The article was submitted 09/04/2023; approved after reviewing 10/26/2023; accepted for publication 10/26/2023.