МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
имени М.В.ЛОМОНОСОВА

*На правах рукописи*

**Шайхисламов Денис Ильгизович**

# Исследование и разработка методов для сравнительного анализа суперкомпьютерных приложений на основе технологий интеллектуального анализа данных

Специальность 2.3.5.
Математическое и программное обеспечение вычислительных систем, комплексов и компьютерных сетей

Автореферат
диссертации на соискание учёной степени
кандидата физико-математических наук

Москва — 2025

Диссертация подготовлена в лаборатории анализа суперкомпьютерных систем и приложений научно-исследовательского вычислительного центра Московского государственного университета имени М.В. Ломоносова.

**Научный руководитель:** **Воеводин Вадим Владимирович**
кандидат физико-математических наук

**Официальные оппоненты:**

**Баркалов Константин Александрович,**
доктор технических наук, доцент,
Национальный исследовательский Нижегородский государственный университет им. Н.И. Лобачевского, институт информационных технологий, математики и механики, кафедра математического обеспечения и суперкомпьютерных технологий, заведующий кафедрой

**Ильин Вячеслав Анатольевич,**
доктор физико-математических наук, старший научный сотрудник,
Национальный исследовательский центр "Курчатовский институт", Курчатовский комплекс НБИКС-природоподобные технологии, главный научный сотрудник

**Соколинский Леонид Борисович,**
доктор физико-математических наук, профессор,
Южно-Уральский государственный университет (национальный исследовательский университет), институт естественных и точных наук, кафедра системного программирования, заведующий кафедрой

Защита диссертации состоится «05» марта 2026 г. в 11 часов 00 минут на заседании диссертационного совета МГУ.012.2 при Московском государственном университете имени М.В. Ломоносова по адресу: 119991, Москва, ГСП-1, Ленинские горы, МГУ, д. 1 строение 52, факультет Вычислительной математики и кибернетики, аудитория №238.
Email: ds012.2@cs.msu.ru.

С диссертацией можно ознакомиться в отделе диссертаций научной библиотеки МГУ имени М.В. Ломоносова (Ломоносовский проспект, д.27) и на портале: https://dissovet.msu.ru/dissertation/3735.

Автореферат разослан «____» _____ _____ года.

Ученый секретарь
диссертационного совета МГУ.012.2,
кандидат физико-математических наук Антонов А.С.

## Общая характеристика работы

**Актуальность темы.** Суперкомпьютеры являются важным инструментом для решения научных и прикладных задач в различных областях, таких как физика, геология, метеорология, химия, медицина и многих других. Использование суперкомпьютеров необходимо из-за вычислительной сложности некоторых задач, решение которых на обычных компьютерах может занять дни, месяцы или даже годы. Однако эффективно использовать суперкомпьютерные системы очень непросто как из-за чрезвычайно сложной архитектуры этих систем, так и из-за сложности создания высокопроизводительных параллельных программ. Для достижения высокой производительности требуются специализированные знания и опыт в разработке эффективных приложений для суперкомпьютеров, которыми зачастую не обладают пользователи суперкомпьютера из других областей науки.

Обеспечение высокой эффективности работы суперкомпьютера требует постоянного контроля его состояния и качества функционирования. Для этого используется специализированное системное программное обеспечение, которое собирает и анализирует разнообразные данные как о работе отдельных программных и аппаратных компонент суперкомпьютера (вычислительных узлов, служебных серверов, файловой системы, менеджера ресурсов и т.д.), так и о выполнении пользовательских приложений. При этом анализ производительности выполняемых на суперкомпьютере приложений очень важен, поскольку именно производительность приложений является одним из ключевых факторов, влияющих на эффективность работы суперкомпьютерной системы в целом.

В качестве входных данных для изучения производительности пользовательских приложений чаще всего используется информация о загрузке и поведении вычислительных узлов, собираемая во время выполнения этих приложений. Для получения подобной информации используются различные системы мониторинга (такие как Zabbix, Nagios, Collectd или DiMMon), которые могут собирать данные от самых разных источников на вычислительном узле — операционная система, аппаратные процессорные датчики, сетевая карта, графический ускоритель и т.д. Также важно получать от менеджера ресурсов информацию о запуске приложений. Широкий спектр собираемых данных и их источников предоставляет системным администраторам обширный объем разнообразной полезной информации, который, однако, невозможно в полной мере обработать вручную.

Поэтому в настоящее время все более важным становится вопрос разработки методов на основе технологий интеллектуального анализа данных для оценки состояния суперкомпьютеров. Эти методы позволят администраторам проще и эффективнее выявлять и устранять проблемы, которые

могут снижать качество функционирования суперкомпьютеров. Примерами подобных проблем являются низкая производительность пользовательских заданий, неоптимальная конфигурация системного или прикладного ПО или неэффективная работа менеджера ресурсов, что приводит как к замедлению выполнения пользовательских расчетов, так и к простою вычислительных ресурсов.

За последние десятилетия наблюдается значительный рост применения методов интеллектуального анализа данных в различных областях, поскольку они позволяют выявлять закономерности в больших объемах данных и использовать их для решения практических задач с высокой точностью. Примерами таких областей являются машинное зрение (распознавание объектов, сегментация изображений), медицина (диагностика заболеваний, разработка лекарственных средств), геология (прогнозирование месторождений полезных ископаемых, моделирование качества воды), естественная обработка языка (распознавание речи, перевод и генерация текста) и многие другие. Активное развитие методов интеллектуального анализа данных повышает их применимость и в сфере высокопроизводительных вычислений и суперкомпьютеров. Современные суперкомпьютеры собирают огромное количество информации о работе своих компонент и запускаемых приложений, которая может быть использована для анализа качества их функционирования, однако ручной анализ этих данных зачастую затруднителен или даже невозможен из-за их большого объема. Методы интеллектуального анализа данных помогают решать эту проблему; в частности, они уже применяются для предсказания времени выполнения приложений или анализа состояния инженерной инфраструктуры.

Еще одной задачей, для решения которой можно применять методы интеллектуального анализа данных, является поиск схожих приложений, выполняющихся на суперкомпьютере, на что и направлена данная работа. В данном случае схожими будем называть приложения, которые обладают одними и теми же важными свойствами, связанными с поведением приложения во время его выполнения. Практика показывает, что если приложение А обладает некоторыми свойствами (например, использует определенный программный пакет), и при этом оно схоже с приложением Б, то в ряде случаев можно считать, что приложение Б также обладает этими свойствами. Само определение схожести меняется в зависимости от того, какие свойства или особенности приложений рассматриваются, и это будет влиять на то, как в данной работе будет определяться оценка расстояния и построенные на ее основе методы выявления схожих приложений. Разработка методов обнаружения схожих приложений позволит анализировать новые задания, опираясь на результаты анализа изученных ранее приложений: проводить кластеризацию заданий, прогнозировать их

поведение, выявлять аномальные запуски, выявлять проблемы с производительностью и т.д. Это значительно упрощает оценку поведения и эффективности выполнения приложений для пользователей и администраторов суперкомпьютеров.

**Целью** данной диссертационной работы является исследование и разработка методов на основе технологий интеллектуального анализа данных для решения задачи выявления схожих суперкомпьютерных приложений, а также реализация и применение разработанных методов на практике для решения актуальных для администраторов и пользователей задач, таких как определение используемых программных пакетов в приложениях или кластеризация приложений по их поведению.

Для достижения поставленной цели было необходимо решить следующие **задачи**:

1. Разработать и экспериментально исследовать методы выявления схожих суперкомпьютерных приложений с использованием технологий интеллектуального анализа данных.
2. Разработать алгоритмы для исследования потока суперкомпьютерных заданий на основе предложенных методов выявления схожих приложений.
3. Реализовать и апробировать программные решения для предложенных алгоритмов с целью исследования реального потока суперкомпьютерных заданий.

**Научная новизна:** В представленном исследовании предложены новые методы выявления схожих суперкомпьютерных приложений на основе технологий интеллектуального анализа данных, демонстрирующие высокую точность и эффективность. Эти методы основаны на анализе как статических данных (исполняемых файлов приложений), так и динамического поведения заданий во время их выполнения. На основе этих методов предложены новые подходы к решению различных актуальных задач: обнаружение используемых программных пакетов и их версий в приложениях; кластеризация пользовательских заданий для определения групп пользователей, решающих похожие задачи; выявление “сбойных” групп заданий для их последующего детального анализа с целью выявления проблем с производительностью; предсказание оценок качества использования суперкомпьютерных ресурсов; обнаружение аномального поведения заданий на основе исторических данных. Предложенные методы могут быть также использованы для решения других важных задач, таких как прогнозирование времени выполнения заданий или создание рекомендательных систем, помогающих пользователям оптимизировать производительность своих параллельных программ.

**Практическая значимость** работы состоит в создании программного решения, позволяющего с высокой точностью выявлять схожие

суперкомпьютерные приложения, а также в реализации вышеуказанных способов применения методов поиска схожих приложений с целью анализа потока заданий. Данное программное решение активно используется на суперкомпьютере петафлопсного уровня производительности Ломоносов-2, где показало свою применимость на практике и позволило администраторам данной вычислительной системы существенно продвинуться в исследовании качества ее работы. Данное решение может применяться и на других вычислительных системах.

**Методология и методы исследования.** При получении результатов диссертационной работы использовались элементы теории вероятностей, методы интеллектуального анализа данных и математической статистики. При выполнении программной реализации методов выявления схожих суперкомпьютерных приложений использовались методы объектно–ориентированного анализа и проектирования, классические алгоритмы и структуры данных.

**Основные положения, выносимые на защиту:**

1. Два разработанных подхода, основанные на анализе статической информации (исполняемых файлов приложения) и динамических данных о поведении задания (данных от системы мониторинга), позволяют решать задачу выявления схожих суперкомпьютерных приложений. По результатам апробации данные методы показывают высокое качество работы.
2. Подходы к выявлению схожих суперкомпьютерных приложений позволяют создать методы для решения задач обнаружения используемых программных пакетов и их версий в приложениях, кластеризации приложений, а также предсказания оценок качества использования суперкомпьютерных ресурсов. Экспериментальное исследование полученных решений на потоке заданий, запускаемых на суперкомпьютере Ломоносов-2, демонстрирует их работоспособность и применимость на практике.

**Достоверность** полученных результатов обеспечивается масштабной апробацией разработанных и реализованных методов анализа приложений на суперкомпьютере «Ломоносов-2», обоснованием принятых решений по их разработке и внедрению.

**Апробация работы.** Представленные в работе результаты докладывались на следующих международных и российских конференциях:

1. XXVIII Байкальская Всероссийская конференция с международным участием, г. Иркутск, Россия, 29 июня - 8 июля 2023
2. Международная конференция «Parallel Processing and Applied Mathematics 2022», г. Гданьск, Польша, 8-11 сентября 2022
3. Международная конференция «Международная научная конференция студентов, аспирантов и молодых учёных «Ломоносов-2022», г. Москва, Россия, 20,21 апреля 2022

4. Международная конференция «Суперкомпьютерные дни в России 2021», г. Москва, Россия, 28 сентября 2021
5. Международная конференция «Параллельные вычислительные технологии (ПаВТ) 2021», Онлайн, Россия, 30 марта 2021
6. Международная конференция «Международная научная конференция студентов, аспирантов и молодых учёных «Ломоносов-2020», г. Москва, Россия, 10-27 ноября 2020
7. Международная школа-конференция «Интеллектуальный анализ Больших данных и распределенные системы», г. Будва, Hotel Splendid, Черногория, 1 октября 2019

**Личный вклад.** Все представленные в диссертации результаты получены лично автором. В работах [A1-A3] подготовка части материалов к публикации проводилась совместно с соавтором, причем вклад диссертанта был определяющим. В работах [A1-A3] Воеводину Вад.В. принадлежит постановка задачи, обсуждение способов и результатов ее решения, а также совместная верификация результатов. В работе [A4] автором был самостоятельно разработан метод предсказания оценок качества использования суперкомпьютерных ресурсов приложениями, который основан на сравнительном анализе суперкомпьютерных приложений с использованием статических и динамических данных, а также проведено экспериментальное исследование предложенного метода.

**Публикации.** Основные результаты по теме диссертации в полной мере изложены в 4 научных работах, в том числе в 4 публикациях в рецензируемых научных изданиях, определенных п. 2.3 «Положения о присуждении ученых степеней в Московском государственном университете имени М.В.Ломоносова».

**Объем и структура работы.** Диссертация состоит из введения, четырех глав и заключения. Полный объем диссертации составляет 99 страниц текста с 26 рисунками и 10 таблицами. Список литературы содержит 45 наименований.

## Основное содержание работы

Во **введении** описывается область исследования, показывается актуальность работы, раскрываются ее цели и задачи, обосновываются научная новизна и практическая значимость работы.

**Первая глава** посвящена обзору существующих подходов к обнаружению схожих приложений с помощью методов интеллектуального анализа данных. В задаче выделения схожих приложений выделяются два главных направления исследования: анализ статических и динамических данных.

В **разделе 1.1** рассматриваются методы анализа статических данных. Под статическими данными понимаются те данные, которые доступны до запуска приложения на суперкомпьютере. Примерами являются: данные планировщика (например, запрашиваемое количество вычислительных узлов или ядер, пути до исполняемых файлов, максимальное время исполнения), исходный код, исполняемые и объектные файлы приложений и т.д. Основным преимуществом методов анализа статических данных является возможность работы с данными до запуска приложения, что исключает зависимость от конкретного запуска. Однако, это же преимущество является и главным недостатком, поскольку поведение приложения может значительно изменяться в зависимости от входных данных, параметров запуска, используемого оборудования и других факторов, которые невозможно отследить с помощью статических данных.

В **разделе 1.2** рассматриваются методы анализа динамических данных. Под динамическими данными понимаются те данные, которые собираются во время работы приложения. В текущей работе представляют интерес данные о производительности заданий, собираемые системой мониторинга на вычислительных узлах. Поэтому особое внимание уделяется обзору методов анализа временных рядов, получаемых из данных систем мониторинга.

В **разделе 1.3** проводится анализ представленных ранее методов и их применимость для решения поставленной задачи.

**Вторая глава** посвящена исследованию и разработке методов для выявления схожих суперкомпьютерных приложений на основе статической информации. В данному случае, приложения будут считаться схожими, если их исполняемые файлы используют схожие имена функций и переменных.

В **разделе 2.1** рассматривается предлагаемый метод на основе применения модели нейронной сети Doc2Vec. Этот метод используется для анализа исполняемых и объектных файлов, что требует получения доступа к полным путям к этим файлам. После получения путей к файлам, с помощью Linux утилиты ‘nm‘ извлекаются имена используемых функций и переменных. Эти имена служат источником данных для модели Doc2Vec, которая может преобразовать этот набор данных в вектор фиксированной длины. Данные вектора сравниваются с помощью косинусного расстояния, которое определяется как единица минус косинус угла между векторами большой размерности. Чем ближе значение косинусного расстояния к нулю, тем больше близость между векторами.

Информация о полных путях к исполняемым и объектным файлам может быть получена непосредственно от планировщика заданий (например, Slurm). Однако пользователи могут использовать скрипты запуска, в которых запускаются несколько разных исполняемых файлов, или могут

запускать приложения из другого приложения (например, из интерпретатора Python), о которых планировщик заданий не имеет информации. Существуют различные способы решения данной проблемы, одним из которых является замена команд компоновщика (ld) и запуска (mpirun). Это позволяет вставить вспомогательную систему, которая при попытке запуска приложения будет иметь возможность сохранить информацию о местоположении всех запускаемых исполняемых и связанных файлов, количестве запущенных исполняемых файлов, информации о компиляции приложения и т.д.

В **разделе 2.2** представлен предлагаемый алгоритм работы метода. Он включает следующие шаги:

- Извлечение данных об исполняемых файлах заданий. Источниками такой информации могут выступать планировщики заданий (Slurm, PBS), сторонние системы (XALT).
- Извлечение термов — используемых имен функций и переменных — из исполняемых файлов с помощью утилиты ‘nm‘.
- Предобработка термов — удаление наиболее часто встречающихся имен функций и переменных.
- Применение модели Doc2Vec на полученных термах для получения вектора фиксированной длины для данного исполняемого файла. Данный вектор и будет использоваться в дальнейшем для выявления схожести приложений.

**Раздел 2.3** подробно описывает процесс обучения модели Doc2Vec и подбора оптимальных параметров модели с целью улучшения ее производительности. Для тестирования работы модели и оптимизации ее параметров вручную были размечены 130 заданий различных пользователей. В процессе разметки задания объединялись в группы таким образом, чтобы задания внутри каждой группы имели в основном пересекающиеся имена функций и переменных. Данные задания были использованы только для первичной оценки качества работы модели.

Оценка качества работы метода проводилась следующим образом:

- У каждого из размеченных заданий были извлечены используемые имена функций и переменных.
- С помощью модели Doc2Vec имена функций и переменных были преобразованы в вектора фиксированной длины.
- К полученным векторам было попарно применено косинусное сходство, что дает матрицу расстояний между размеченными заданиями.
- Матрица расстояния подается на вход методу иерархической кластеризации (использовался метод агломеративной кластеризации), что в результате дает автоматическую группировку (разметку) заданий, которая сравнивается с ручной.

Для оценки качества работы модели использовался Adjusted Rand Index, который выявляет степень сходства между ручной и автоматической разметкой элементов.

Проводилась оптимизация параметров модели Doc2Vec, таких как размерность вектора фиксированной длины, а также предобработка используемых имен функций и переменных. Предобработка заключалась в исключении часто используемых имен функций и переменных, так как данные имена не добавляют новой информации об исполняемом и объектном файле. После процесса оптимизации параметров, была получена оценка Adjusted Rand Index, составляющая 0.79, что показывает хорошее качество работы.

**Третья глава** посвящена исследованию и разработке динамических методов поиска схожих приложений.

**Раздел 3.1** начинается с описания данных о поведении приложения, которые используются в качестве входных данных для разработанных методов. Они предоставляются существующей системой мониторинга, которая собирает усредненные данные датчиков со всех вычислительных узлов суперкомпьютера. Эти данные описывают загрузку центральных процессоров и графических ускорителей, интенсивность работы с памятью (количество промахов в L1 и L3 кэш-память в секунду) и сетью (количество отправленных и принятых по коммуникационной сети байт данных в секунду) и т.д., и в дальнейшем они будут называться динамическими характеристиками. Значения каждой динамической характеристики во время выполнения суперкомпьютерного задания формируют временной ряд, в котором каждая точка представляет собой усредненное значение соответствующего датчика за выбранный период работы приложения. В текущей системе мониторинга суперкомпьютера Ломоносов-2 данные агрегируются за период в 1 минуту. В результате каждое задание характеризуется многомерным временным рядом, который и является необходимыми входными данными для работы разработанного метода. Таким образом, в случае динамических данных, приложения будут считаться схожими, если многомерные временные ряды, описывающие поведение приложений во время выполнения, являются схожими.

В **разделе 3.2** описывается предлагаемый метод анализа полученных временных рядов. Данный метод использует алгоритм Dynamic Time Warping (DTW), который ищет оптимальное соответствие между точками временных рядов. Это соответствие может быть использовано для получения оценки расстояния между временными рядами, что позволит определить схожесть заданий, представленных этими рядами. Для вычисления расстояния между отдельным точками временных рядов используется Евклидово расстояние. Основными преимуществами DTW являются его устойчивость как к глобальным, так и к локальным

временным сдвигам, которые с точки зрения приложений могут характеризоваться изменением длительности выполнения определенных этапов расчетов (как за счет вариаций внутри приложения, так и из-за изменений в программно-аппаратной среде суперкомпьютера), а также возможность расчета оценки расстояния для временных рядов различной длины.

**Раздел 3.3** рассматривает один из важнейших этапов динамического метода — предобработку данных и оценку точности работы метода. Динамические характеристики существенно отличаются по своим абсолютным значениям, из-за чего необходимо преобразовать каждый параметр таким образом, чтобы влияние характеристик на итоговое расстояние было одинаковым. Основными методами являются стандартизация (данные преобразуются таким образом, что их среднее значение становится равным нулю, а стандартное отклонение равно единице), и нормализация (позволяющая привести значения в диапазон от 0 до 1). Другим преобразованием является логарифмирование, в котором каждое значение x заменяется на значение log(1+x). Такое преобразование имеет смысл для данных, имеющих сильное смещение к значению 0, что характерно для некоторых динамических характеристик, таких как количество промахов в кэш память первого уровня и интенсивность использования коммуникационной сети. Даже после стандартизации, у таких данных большинство значений будет иметь смещение в сторону 0, и логарифмирование позволяет уменьшить такое смещение и лучше различать низкие значения показателей датчиков.

На следующем шаге исследования требовалось оценить влияние различных видов преобразований на качество работы метода. Для оценки точности метода была проведена ручная разметка 300 пар приложений, где для каждой пары было отмечено, являются ли эти задания схожими или нет.

Точность в данном случае определялась как отношение количества пар заданий, правильно определенных как схожие или несхожие, к количеству общих пар заданий. На размеченных 300 пар приложений точность составила 0.7. Это значение можно улучшить путем анализа случаев, в которых метод совершил ошибку, и последующим поиском и исправлением причин возникновения этих ошибок. Проведенный анализ показал, что для некоторых характеристик требуется дополнительное снижение их влияния. Это было сделано с помощью применения дополнительных весов, которые умножаются на значения характеристик после предобработки данных. Таким образом, было необходимо не только подбирать методы преобразования для каждой характеристики, но и значения весов. В результате подбора методов и значений весов удалось достичь точности 0.9. Для оценки применимости метода предобработки данных к новым заданиям, не использовавшимся при оптимизации параметров, было размечено

еще 425 пар заданий. Это позволило исключить возможность переобучения метода на данных, использованных для оптимизации, и обеспечить его эффективную работу на реальных данных. Точность на этой выборке составила 0.85. Незначительное снижение точности при оценке качества работы на тестовой выборке является ожидаемым и показывает отсутствие переобучения параметров метода предобработки данных.

В **четвертой главе** приведены различные применения разработанных методов обнаружения схожих приложений в практике работы суперкомпьютерного центра.

В **разделе 4.1** приведено описание решения задачи выявления программных пакетов в суперкомпьютерных приложениях. Вначале приводится описание существующей системы выделения пакетов, использующейся на суперкомпьютере Ломоносов-2, основанной на ПО XALT. ПО XALT собирает полные пути ко всем исполняемым и объектным файлам, используемым в пользовательских приложениях. Текущая система выделения пакетов основывается на поиске ключевых слов в этих путях, и при обнаружении таких слов считает, что используется определенный пакет. У данной системы есть множество недостатков, главный из которых — невозможность определить использование программного пакета при изменении имени исполняемого файла или каталога. Как будет показано далее, эти недостатки приводят к тому, что ПО XALT не может обнаружить использование пакетов в половине пользовательских заданий. В дальнейшем в данной главе термин "XALT" будет использоваться для обозначения текущей системы выделения пакетов.

Стоит отдельно отметить, что если XALT обнаружил использование пакета, то можно с большой долей уверенности сказать, что он это сделал корректно. При выборочной проверке заданий как при повседневном использовании ПО, так и при апробации предлагаемого статического метода выделения пакетов были выявлены единицы случаев, когда XALT некорректно выделял использование пакета. Данный факт позволяет использовать результат работы XALT как хороший источник разметки данных для предлагаемого метода.

Далее предоставляется алгоритм адаптации статического метода выделения схожих приложений, описанного в разделе 2, для решения задачи выделения программных пакетов. Основная суть подхода заключается в использовании исторических данных для определения свойств в новых заданиях, которые еще не были исследованы. С помощью исторических данных строится база знаний, которая включает в себя информацию об изученных ранее заданиях и используемых в них пакетах. Новые задания сравниваются с уже прошедшими заданиями из базы знаний. Если в базе знаний есть схожее задание — считается, что в новом задании используются все те же пакеты, что и в задании из базы. Данный подход применим

как к статическому, так и к динамическому методам. В итоге предлагается использовать следующий алгоритм обнаружения пакетов:

1. Построение базы знаний
    а) Извлечение из данных системы XALT всех заданий за выбранный прошедший период, для которых имеется информация о том, какие программные пакеты были использованы. Длительность периода подбирается эмпирически администраторами.
    б) Построение уникального набора заданий из предыдущего пункта. Уникальным набором заданий будет считаться такой набор, задания в которых не схожи друг с другом. Функция схожести будет использоваться как в статическом методе — косинусное сходство для вывода модели Doc2Vec.
    в) Сохранение в базе знаний полученного набора заданий как массив из пар {задание : используемые программные пакеты}.
2. После получения данных о новом задании:
    а) Сравнение нового задания со всеми заданиями из базы знаний.
    б) Если находятся задания из базы знаний, расстояние до которых меньше порогового значения, то считаем, что в новом задании используются аналогичные программные пакеты.
    в) (Опционально) Если такие задания не находятся, но другая система выделения пакетов (например, XALT) обнаружила, что программные пакеты присутствуют, то необходимо добавить данное задание в базу знаний.

База знаний статического метода основывается на использовании данных XALT — среди исторических данных выбираются все задания, в которых XALT обнаружил использование пакетов. С помощью статического метода выделения схожих заданий мы можем исключить те задания, которые практически идентичны друг другу, что позволит получить только наиболее репрезентативные задания для определенных пакетов, которые будут включены в базу знаний.

Также в данном разделе описывается процесс апробации показанного метода на реальных заданиях. Для этого были проанализированы задания за 11 месяцев 2022 года, и результаты работы предложенного метода сравниваются с XALT.

Во время апробации статического метода анализировалась примерно 71 тысяча заданий, среди которых было выделено около 34 тысяч заданий с пакетами. Результат сравнения с XALT показана на таблице **??**, который показывает, что XALT и статический метод схожи на ∼92%. Среди всех

2985 заданий, в которых статический метод обнаружил наличие пакетов, а XALT — нет, все эти задания действительно использовали найденные пакеты. В 10% случаев XALT обнаружил пакеты, которые не смог выявить статический метод. В этих случаях возможно 2 варианта: или XALT некорректно сработал, или статический метод не смог выделить использование пакета. Первый вариант возможен из-за того, как XALT выделяет использование пакетов: любое переименование имени исполняемого файла может привести к некорректному определению пакета. На практике такие случаи находились, но редко. Анализ случаев второго варианта позволяет понять, почему предложенный метод не справился с поставленной задачей. Это может быть вызвано несколькими факторами, такими как недостаток прав доступа на чтение исполняемых файлов, их модификацию до начала анализа или отсутствие подобного задания в базе знаний. Последнюю проблему можно решить регулярным обновлением базы знаний.

Таблица 1 — Результат сравнения статического метода с XALT

| | Пакет обнаружен статическим методом | Пакет не обнаружен статическим методом |
|---|---|---|
| Пакет обнаружен с помощью XALT | 27667 | 2857 |
| Пакет не обнаружен с помощью XALT | 2985 | 37491 |

Следующим шагом является адаптация динамического метода выделения схожих приложений, описанного в разделе 3, для решения задачи выделения программных пакетов. Алгоритм, описанный выше для статического метода, применим и для динамического, за исключением метода построения базы знаний.

Построение базы знаний для динамического метода аналогично тому, как это было сделано для статического метода, затруднительно. Этому препятствуют несколько причин:

- Количество вариантов поведения одного и того же исполняемого файла может быть велико, как минимум из-за различных входных данных и параметров запуска, поэтому для одного приложения недостаточно иметь только один запуск в базе знаний.
- Из-за вычислительной сложности динамического метода не предоставляется возможным сравнение заданий за очень длительный период.

По опыту администраторов, пользователи часто последовательно запускают задания, которые имеют схожее поведение. Учитывая данный факт и причины, указанные ранее, можно использовать следующий алгоритм для построения базы знаний:

– Рассмотрение заданий только за небольшой временной период, например, за последние 1.5 месяца.
– Использование совместных данных от XALT и статического метода для получения соответствия {поведение задания : используемый программный пакет}.
– С помощью динамического метода нахождение заданий со схожим поведением, что исключит в базе знаний дубликаты по поведению.
– Использование полученных заданий как базу знаний для выявления пакетов.

В ходе экспериментов использовались задания за предыдущие 1.5 месяца, из-за чего в базе знаний в среднем находились 1.5 тысяч заданий.

Для апробации динамического метода были проанализированы задания за 2 месяца работы суперкомпьютера Ломоносов-2. Всего было проанализировано 3300 заданий. Такое небольшое количество за этот период объясняется следующими причинами: 1) не рассматривались задания, которые выполнялись меньше часа, так как это не дает достаточное количество динамических данных для корректного анализа временных рядов; 2) данные могут отсутствовать из-за сбоев в работе системы мониторинга. Так как и XALT, и статический метод выделения пакетов основываются на анализе статических данных задания, сравнение результатов работы динамического метода было произведено с комбинированными результатами работы XALT и статического метода, т.е. если хотя бы один из методов выделения пакета обнаружил использование пакета, то считается, что он присутствует. Результат сравнения показал, что динамический метод не смог выделить пакеты в 15% заданий, из-за чего использование только динамического метода в боевом режиме не представляется целесообразным. Результат сравнения динамического метода со статическим методом и XALT представлена в таблице **??**. Дополнительная ручная проверка заданий, в которых динамический метод обнаружил пакет, в то время как другие методы этого не сделали, показала, что как минимум в трети случаев действительно использовался найденный пакет (в остальных случаях однозначного вывода сделать не удалось).

Таблица 2 — Результат сравнения динамического метода с XALT и статическим методом

| | Пакет обнаружен динамическим методом | Пакет не обнаружен динамическим методом |
|---|---|---|
| Пакет обнаружен XALT и статическим методом | 1783 | 250 |
| Пакет не обнаружен XALT и статическим методом | 358 | 903 |

Особое внимание уделяется внедрению разработанной системы обнаружения пакетов на основе статического метода для постоянного использования на суперкомпьютерной системе Ломоносов-2. Несмотря на возможность использования динамического метода совместно со статическим, было принято решение не применять его для постоянной обработки суперкомпьютерного потока заданий. Такое решение было обусловлено тем, что дополнительное использование динамического метода лишь немного увеличивает точность работы, однако заметно увеличивает вычислительную сложность предложенного решения.

Дайджест по пакетам (Lomonosov-2) Inbox ×

Stat2 <info@stat2.parallel.com> May 25, 2022, 4:02 AM
to me, vadim, vadim_voevodin

Дайджест по пакетам за 24-05-2022

Найдено пакетов 87/204

| Job id | Username | unique | doc2vec | xalt | dynamic | Versions | Command | Workdir | Comment |
|---|---|---|---|---|---|---|---|---|---|
| | | | lammps | lammps | | | | | |
| | | lammps | lammps | | | | | | srun only \| [Errno 13] Permission denied: |
| | | | cp2k | cp2k | | | | | nm: File format not recognized |
| | | | cp2k | | | | | | nm: : File format not recognized |
| | | | quantum espresso,gamess | | | | | | nm: : File format not recognized |
| | | lammps | lammps | lammps | | | | | [Errno 2] No such file or directory: |
| | | | gromacs | gromacs | | | | | [Errno 2] No such file or directory: \| srun only |
| | | | | gromacs | | | | | [Errno 13] Permission denied: |
| | | | | vasp | | | | | : File format not recognized |

Рисунок 1 — Пример отчета по заданиям с обнаруженными пакетами

Анализ суперкомпьютерного потока заданий проводится статическим методом периодически (по умолчанию каждые 30 минут). В процессе анализа обнаруженные программные пакеты сохраняются в локальную базу данных. Ежедневно администраторам предоставляется отчет с указанием того, какие пакеты были обнаружены в заданиях с помощью статического метода и XALT. Это обеспечивает более удобное сравнение результатов работы обоих методов. Пример данного отчета показан на рисунке **??**.

Результаты работы методов выделения программных пакетов за 2021 и 2022 годы показаны в таблице **??**. Применение статического метода

позволило выявить в 1.5 раза больше заданий (в 2 раза больше процессорочасов) по сравнению с использованием только XALT. А для пакетов, как, например, Espresso, статический метод выделил в 4 раза больше заданий.

На текущий момент приложения, в которых было обнаружено использование программных пакетов, составляют 47% от общего количества используемых процессоро-часов. Такой низкий процент обусловлен тем, что не все пользователи используют программные пакеты для решения своих прикладных задач, а также тем, что администраторам неизвестны некоторые пакеты, которые могут использовать пользователи.

Таблица 3 — Количество заданий, в которых выявлено использование программных пакетов, результаты для XALT и статического метода

| Пакет | Статический метод или XALT | И статический метод, и XALT | Только XALT | Только статический метод |
|---|---|---|---|---|
| LAMMPS | 16 493 | 14 332 | 719 | 1 442 |
| VASP | 5 647 | 3 253 | 1 438 | 956 |
| Amber | 455 | 278 | 91 | 86 |
| Cabaret | 849 | 196 | 60 | 593 |
| Espresso | 1 198 | 226 | 9 | 963 |
| Orca | 444 | 280 | 107 | 57 |

В **разделе 4.2** описывается решение задачи кластеризации суперкомпьютерных заданий. Даются базовые определения задачи кластеризации приложений, а также описывается, какие методы могут быть применены для решения данной задачи. Кластеризация заданий в основном полезна администраторам, например, для выявления наиболее влиятельных заданий с точки зрения используемых ресурсов и аномальных запусков, но может быть полезна и пользователям для упрощения процесса анализа своих приложений при большом числе их запусков. Особое внимание уделяется способам оценки качества работы методов кластеризации. Всего существует два способа: первый применяется при наличии ручной разметки групп элементов, второй — в случае ее отсутствия. В нашем случае наиболее подходит метод Adjusted Rand Index, принадлежащий к первому типу.

Далее описывается предложенный метод кластеризации, который основан на совместном применении статического и динамического методов выделения схожих приложений. Метод основан на предположении о том, что задания могут быть похожи по поведению только в том случае, если их исполняемые файлы схожи. Данное предположение подтверждается наблюдениями администраторов о реальных запускаемых заданиях. В связи с этим алгоритм состоит из двух этапов. Первый этап заключается в использовании статического метода для получения оценки расстояния между

заданиями на основе анализа исполняемых файлов. Полученные значения расстояний могут быть кластеризованы для формирования больших групп заданий, которые схожи статически. Вторым этапом является применение динамического метода выделения схожих приложений для определения расстояния между заданиями внутри каждого статического кластера. Полученные значения расстояний также могут быть кластеризованы, что позволит формировать группы заданий, которые схожи не только статически, но и по своему поведению. Полученная на этом этапе разметка становится финальным результатом работы алгоритма.

Часто в методах кластеризации применяется Евклидово расстояние в качестве метрики для измерения расстояния между элементами, но в таком случае нельзя использовать результаты работы статического и динамического методов, так как в данных методах кластеризации используются свойства Евклидова расстояния, которые отсутствуют в оценках расстояния у статического и динамического методов. Для кластеризации элементов с нестандартными функциями расстояния часто применяются иерархические методы, в которых происходит последовательное объединение меньших кластеров в большие или разделение больших кластеров на меньшие. В данной работе использовался метод агломеративной кластеризации, в котором малые кластера объединяются в один в зависимости от расстояния между элементами.

Следующим логическим шагом является подбор параметров алгоритма с целью повышения точности его работы. В данном случае, поскольку мы используем методы оценки расстояния между заданиями, эти методы уже настроены, и поэтому необходимо подобрать только параметры методов кластеризации. Для оценки качества работы было принято решение также использовать Adjusted Rand Index, однако для его работы требуется ручная разметка приложений. В связи с этим была выполнена ручная разметка примерно 600 приложений, в которых 300 заданий использовались для оптимизации параметров (обучающая выборка), а остальные 300 — для проверки того, что выбранный набор параметров не переобучен и будет хорошо переноситься и на другую выборку приложений (тестовая выборка). В результате оптимизации на обучающей выборке, значение оценки на тестовой выборке изменилось с 0.7 до 0.8, что свидетельствует об улучшении качества работы метода.

При периодической обработке потока, возникает потребность в кластеризации новых заданий, завершившихся за прошедший период. Поскольку иерархическая кластеризация не может добавлять новые задания к уже существующей разметке, каждый раз требуется проводить кластеризацию заново, что является вычислительно затратным при выполнении частого анализа интенсивного потока заданий. Для решения этой проблемы был предложен упрощенный метод, основанный на кластеризации с

использованием метода ближайшего соседа. Идея метода схожа с предложенной ранее идеей метода выделения использования программных пакетов. Имеется база знаний, с заданиями из которой сравнивается новое приложение. В данном случае базой знаний выступает результат работы ранее описанного метода кластеризации, то есть соответствие {задание : ID группы, к которому принадлежит задание}. Если находится близкое задание в базе знаний, считается, что новое задание тоже относится к данной группе. Этот алгоритм значительно упрощает работу метода кластеризации, однако требуется проверить, настолько при этом снижается качество работы. Для проверки этого было предложено сравнить результат работы предложенного упрощенного метода с результатом исходной версии алгоритма кластеризации за фиксированный промежуток времени с помощью оценки Adjusted Rand Index. В проверке участвовало около 1.5 тысяч заданий. Значение оценки Adjusted Rand Index составило 0.99, что указывает на практическую идентичность разметки заданий у обоих методов.

После общего описания метода кластеризации приложений, приводятся примеры его применения на практике. Один из таких примеров — выделение аномальных запусков приложений в сериях запусков заданий пользователя. Зачастую пользователи запускают большое количество однотипных экспериментов с небольшими изменениями во входных данных или параметрах запуска, и данные запуски обычно имеют схожее поведение во время их исполнения. Метод кластеризации может объединить данные задания в одну группу, и можно использовать данное свойство для выделения аномалий. В данном контексте аномалией является задание, которое существенно отличается от предыдущих и последующих заданий по своему поведению во время исполнения (контекст задания). Причинами такого существенного отличия могут быть запуск другого приложения, изменение параметров запуска, влияющее на поведение программы (например, изменение алгоритма обработки данных/расчета), специфические особенности реализации программы, при которых для определенного набора входных данных/параметров наблюдается нетипичное поведение, ошибка в реализации приложения, что приводит к некорректному поведению, или аппаратные/системные сбои на вычислительных узлах суперкомпьютера.

Существуют ситуации, когда обнаружить аномалии в работе задания невозможно без учета контекста и анализа серии запусков. Например, чрезмерная общая нагрузка на сетевую файловую систему может снизить производительность задания, что выделяет его среди типичных запусков. Сбои на коммутаторах могут привести к задержкам в сети, что влияет на использование CPU и интенсивность сетевой нагрузки, и без учета серии запусков такие аномалии могут быть расценены как особенности программной реализации. Для выявления таких аномалий предлагается использовать следующий подход: если среди последовательных запусков приложений одной группы попадается задание другой группы, это может

указывать на аномальное поведение. Пример такого запуска (Job 7) показан ниже, где разные группы приложений обозначены разными цветами.

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

При анализе заданий суперкомпьютера за 5 месяцев его работы, с помощью предложенного метода выделения аномалий было выявлено 248 случаев. Все эти аномалии были подвергнуты ручной проверке, и действительно данные задания выделялись по поведению среди других запусков пользователей. В дальнейшем планируется предоставлять пользователям возможность получать уведомления о подобных найденных случаях.

Метод кластеризации может быть также использован для улучшения эффективности методов выделения программных пакетов. Одно из улучшений заключается в выделении запусков с использованием программных пакетов, которые не были обнаружены ни статическими или динамическими методами выделения схожих приложений, ни с помощью XALT. В случае применения этого решения на суперкомпьютере Ломоносов-2, процент таких запусков составил около 2%. Кроме того, кластеризация помогает обнаруживать новые программные пакеты, о которых ранее не было известно. Если образуется большая группа запусков, в которых присутствуют задания нескольких пользователей, то такая группа представляет интерес: есть вероятность того, что в них используется еще не идентифицированные программный пакет или библиотека. Например, в результате анализа таких больших групп был обнаружен ранее неизвестный администраторам суперкомпьютера программный пакет XAMG, который представляет собой пакет решения СЛАУ с несколькими правыми частями.

В **разделе 4.3** описывается решение задачи предсказания оценок качества использования суперкомпьютерных ресурсов. Ранее в МГУ имени М.В. Ломоносова была разработана система оценок, которая позволяет анализировать и сравнивать задания согласно качеству использования различных суперкомпьютерных ресурсов. В настоящее время система работает на суперкомпьютере Ломоносов-2 и вычисляет четыре оценки, которые определяют качество использования центральных процессоров (`cpu score`), памяти (`mem score`), коммуникационной сети (`ib score`) и файловой системы (`fs score`). Значение каждой оценки варьируется от 0 до 100, где 0 указывает на то, что работа с данным типом ресурсов никак не мешала полезным вычислениям в рамках выбранного пользовательского задания, а 100 — что работа с данным типом ресурсов постоянно мешала полезным вычислениям. Для вычисления оценок `score ib` и `score fs` можно использовать данные, которые собираются штатной системой мониторинга, поэтому данные оценки могут быть рассчитаны для всех заданий. Однако для оценок `cpu score` и `score mem` этих данных недостаточно, поэтому необходимо собирать дополнительные метрики производительности во время выполнения приложений. Сбор большего числа метрик вносит

дополнительные накладные расходы, из-за чего данные метрики (а, следовательно, и сами оценки) сейчас собираются только для каждого третьего задания. Соответственно, возникает задача предсказания значений оценок для всех остальных заданий.

Для решения задачи предсказания оценок `score cpu` и `score mem` было принято решение применениить статический и динамический методы выделения схожих приложений. Предлагается использовать следующий алгоритм:

- Собирается база знаний, которая включает в себя историческую информацию о значениях оценок для выполненных ранее заданий (для которых сработал существующий метод вычисления оценок).
- При появлении нового задания, которое не удалось оценить существующим методом, происходит его следующая обработка:
  - С помощью статического анализа в базе знаний находятся все задания, схожие с новым.
  - При нахождении статически схожих заданий, среди них проводится поиск схожих заданий с помощью динамического метода с менее строгим порогом (шаг A), что позволит получить больше заданий для анализа.
  - При отсутствии статически похожих заданий, проводится поиск похожих заданий во всей базе знаний с помощью динамического метода, используя гораздо более строгий порог (шаг B).

Суть шагов A и B состоит в том, что если задания имеют схожие статические данные, вероятность значительного отличия их поведения невелика. Следовательно, для получения большего количества заданий для сравнения можно использовать менее строгий порог.

После получения списка схожих заданий с известными оценками, необходимо предсказать оценку для нового задания путем агрегирования полученных значений. Были рассмотрены три различных способа агрегации: среднее значение, медиана и взвешенное среднее значение с использованием функции softmax. Средневзвешенное значение softmax вычисляется по формуле **??**, где $distance_i$ — расстояние (близость) между новым заданием и i-м заданием из базы знаний, а $score_i$ — оценка для i-го задания из базы знаний.

$$softmax = \frac{\sum_i e^{-distance_i} * score_i}{\sum_i e^{-distance_i}} \tag{1}$$

В целях определения качества работы метода было отобрано примерно 4600 заданий с оценками за 6 месяцев работы суперкомпьютера Ломоносов-2. Для каждого задания была предсказана оценка, которая сравнивалась с реальным значением. Для оценки качества работы метода были выбраны Mean Absolute Error (MAE, формула **??**) и Mean Squared

Error (MSE, формула **??**), где $predicted_i$ — предсказанное значение оценки для задания $i$, $real_i$ — реальное значение оценки, а $N$ — общее количество заданий.

$$MAE = \frac{\sum_i |predicted_i - real_i|}{N} \tag{2}$$

$$MSE = \frac{\sum_i (predicted_i - real_i)^2}{N} \tag{3}$$

В ходе проверки качества работы метода было обнаружено, что метод агрегации влияет на него несущественно, поэтому было решено использовать функцию softmax во всех последующих случаях.

После применения данного метода удалось предсказать оценки (найти схожее задание в базе знаний) для 89% заданий, при этом средняя абсолютная ошибка (MAE) составила 2.11, а среднеквадратичная ошибка (MSE) — 10.42. Это показывает, что в среднем наше предсказание ошибается на 2.11, что в данном случае совсем немного, т.к. при анализе оценок было выявлено, что при запуске одного и того же приложения оценки могут варьироваться на ±5 (что является приемлемым отклонением для данных оценок). Причинами такой вариации могут являться недетерминированность суперкомпьютера или использование разных вычислительных узлов (из-за чего меняется выполнение самого приложения от запуска к запуску), а также качество собираемых данных мониторинга. Также, процент заданий с ошибкой более 10 составил всего 2.5%, что означает, что значительная погрешность при предсказании с помощью разработанного метода возникает редко. На рисунке **??** показана гистограмма распределения числа заданий на основе разницы между предсказанным и реальным значениями оценок. Видно, что большинство заданий имеют ошибку в диапазоне от -5 до 5, что является приемлемым для данной задачи.

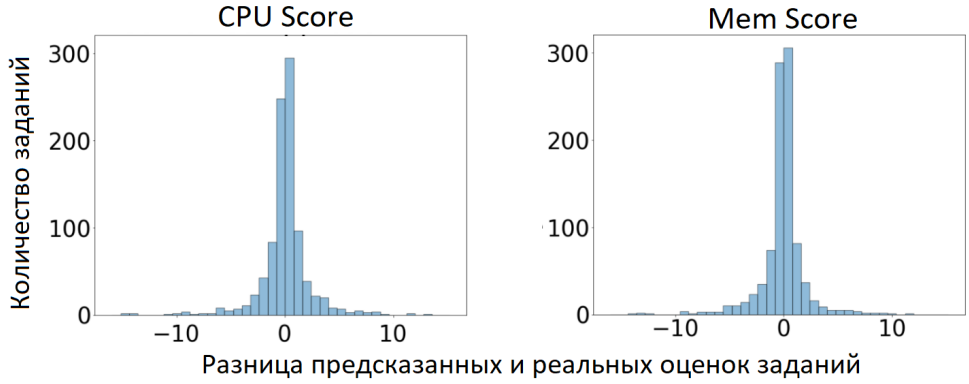


Рисунок 2 — Гистограмма распределения числа заданий с полученной разницей между истинным и предсказанным значениями оценок для центральных процессоров (`score cpu`) и памяти (`score mem`)

В **разделе 4.4** описано реализованное программное решение, регулярно анализирующее поток заданий, выполняющихся на суперкомпьютере Ломоносов-2. Вначале описываются основные источники данных о заданиях, которыми являются: система мониторинга, предоставляющая данные о динамике работы приложений во время их исполнения; система XALT, позволяющая получить информацию о запускаемых исполняемых файлах.

Сценарий работы программного решения для анализа потока заданий устроен следующим образом. Процесс начинается с отбора новых заданий, которые были запущены или завершили свое исполнение за предыдущий период, а далее начинается их анализ. При рассмотрении различных методов анализа суперкомпьютерного потока заданий стало понятно, что первые шаги алгоритма зачастую схожи: выбор заданий для анализа, извлечение данных и их предобработка и т.д., зачастую не зависят от того, какой метод анализа используется. Из-за этого вводится понятие очереди заданий. В данном случае очередью заданий будем называть упорядоченный набор заданий, для которых необходимо провести анализ с помощью нескольких методов, имеющих общие шаги и входные данные. На практике, это выливается в образование двух очередей: для обработки статическими и динамическими методами. В частности, к статической очереди привязана реализация статического метода выделения пакетов, а к динамической очереди — реализации методов оценки качества использования суперкомпьютерных ресурсов, динамического метода выделения пакетов и кластеризации заданий.

Далее описывается архитектура реализованного программного решения, анализирующего суперкомпьютерный поток заданий на основе методов обнаружения схожих приложений. Это решение состоит из следующих слоев, каждый из которых представляет собой этап обработки и анализа заданий: инициализация, извлечение данных и предобработка, анализ и завершение анализа.

В слое инициализации производится инициализация всех необходимых моделей и структур для обработки. Примерами являются модель Doc2Vec, базы знаний, вспомогательные динамические данных о заданиях и т.д.

Следующий слой — извлечение данных и предобработка — отвечает за извлечение данных о задании из внешних источников и их предобработку. Например, на данном этапе извлекаются данные о динамике работы задания из системы мониторинга и проводится предобработка динамических характеристик.

На следующем слое проводится непосредственно анализ данных о задании. После получения данных о задании, проводится его анализ с помощью методов выделения схожих приложений. Например, на данном этапе

проводится предсказание оценки качества использования суперкомпьютерных ресурсов.

Последний слой отвечает за завершение анализа. На данном этапе идет обработка результата анализа с предыдущего слоя, и данный этап привязан к определенному слою анализа. Зачастую, этот слой отвечает за сохранение результатов, как, например, в базу данных/файловую систему.

## Заключение

В данной работе представлены два разных метода поиска схожих приложений, работающих на суперкомпьютере. Первый метод основан на анализе статической информации о приложениях: именах используемых функций и переменных, извлеченных из исполняемых и объектных файлов. Метод показывает хорошие результаты в выявлении схожих приложений (значение ARI на тестовой выборке равно 0.79). Второй метод основан на анализе динамических характеристик, предоставляемых системой мониторинга и описывающих производительность приложений во время выполнения. Метод основывается на алгоритме Dynamic Time Warping. Результаты экспериментов показывают хорошие результаты в обнаружении похожих по поведению приложений: точность 0.85 на тестовой выборке, где точность — отношение количества пар заданий с корректно определенной схожестью к общему количеству пар заданий в выборке.

На основе предложенных методов оценки схожести заданий были предложены варианты решения важных практических задач, которые встают перед администраторами суперкомпьютерных центров. Одной из таких задач является решение проблемы обнаружения пакетов программ (таких как GROMACS, NAMD и др.), используемых в заданиях, которые запускаются на суперкомпьютере. Эта задача очень важна, так как информацию об используемых пакетах можно применять не только для получения новой статистики о функционировании суперкомпьютера или предоставления администраторам новой информации о потребностях пользователей, но и в качестве составляющей рекомендательной системы. Например, если известны свойства пакета (масштабируемость, интенсивность использования процессоров, графических ускорителей или памяти), можно подсказать пользователям, как использовать пакет более эффективно.

Первый предложенный подход, основанный на статическом анализе, показал точность 0.9 на тестовой выборке. Во время тестирования было обнаружено более 40% новых заданий, в которых использовались различные пакеты и что не было обнаружено ранее другими методами. Также было показано, что данный подход может не только обнаруживать известные пакеты, но и помогать выявлять новые, неизвестные пакеты (например,

в нашем случае был обнаружен пакет XAMG для решения систем линейных алгебраических уравнений). Статический метод уже используется на суперкомпьютере “Ломоносов-2” на постоянной основе.

Второй подход, основанный на динамическом анализе, также показал высокую точность на тестовой выборке — 0.85. В ходе его тестирования мы обнаружили, что при рассмотрении заданий с уникальным поведением этот метод выявлял до 20% новых заданий с пакетами, но из-за сложности верификации результатов было принято решение отказаться от применения динамического метода для выделения программных пакетов.

Предложенные в работе методы выделения схожих приложений также были применены для решения задачи предсказания оценок качества использования суперкомпьютерных ресурсов. Данные оценки качества были разработаны ранее администраторами суперкомпьютера “Ломоносов-2”, и их расчет уже запущен в реальном времени. Но данные оценки рассчитываются только для трети запускаемых заданий, из-за чего есть потребность в предсказании этих оценок для оставшихся заданий. Предложенный подход на основе выделения схожих приложений показал высокую эффективность: с помощью него можно предсказать оценки заданий для 87% заданий со средним отклонением в 2.5%, что считается допустимым качеством работы для данной прикладной задачи.

Также была показана эффективность применения методов выделения схожих приложений для решения задачи кластеризации заданий. Кластеризацию предлагается проводить в два этапа. На первом этапе выделяются кластера на основе статических данных, что позволяет сгруппировать задания со схожими исполняемыми файлами. На втором этапе каждый такой кластер разбивается на более маленькие кластера на основе динамических данных. Это позволяет выделять кластера заданий, схожих как по поведению во время их исполнения, так и по исполняемым файлам. На обоих этапах применяется метод агломеративной кластеризации. На тестовой выборке, предложенный метод получил оценку ARI равную 0.75, что говорит о хорошем качестве работы метода. Данный метод кластеризации может быть использован для решения множества практических задач. В качестве примера были приведены задача выделения крупных кластеров заданий для их дальнейшего анализа, а также задача выделения аномальных заданий в сериях запусков отдельных пользователей.

## Публикации автора по теме диссертации

**Научные статьи, опубликованные в рецензируемых научных изданиях, рекомендованных для защиты в диссертационном совете МГУ по специальности и отрасли наук:**

1. Shaikhislamov, D. Solving the problem of detecting similar supercomputer applications using machine learning methods /

D. Shaikhislamov, V. Voevodin // Communications in Computer and Information Science. — 2020. — vol. 1263. — pp. 46-57. — EDN OPGJNI. — (Scopus Q4, Импакт-фактор 0.182 (SJR))[1.125/1 п.л.]
Автором были самостоятельно разработаны два метода для сравнительного анализа суперкомпьютерных приложений на основе статических и динамических данных. Автором также было проведено экспериментальное исследование предложенных методов на суперкомпьютере «Ломоносов-2».

2. Shaikhislamov, D. Development and Practical Application of Methods for Detecting Similar Supercomputer Jobs / D. Shaikhislamov, V. Voevodin // Communications in Computer and Information Science. — 2021. — vol. 1437. — pp. 18-30. — EDN HZXGMO. — (Scopus Q4, Импакт-фактор 0.182 (SJR))[1.1875/1.0675 п.л.]
Автором был самостоятельно разработан метод для обнаружения использования программных пакетов суперкомпьютерными приложениями, который основан на сравнительном анализе приложений с использованием статических данных. Автором также было проведено экспериментальное исследование предложенного метода.
3. Shaikhislamov, D. Analysis of Software Package Usage Based on Methods for Identifying Similar HPC Applications / D. Shaikhislamov, V. Voevodin // Communications in Computer and Information Science. — 2021. — vol. 1510. — pp. 310-321. — EDN DGFJFF. — (Scopus Q4, Импакт-фактор 0.182 (SJR))[1.125/1 п.л.]
Автором был самостоятельно разработан метод для обнаружения версий программных пакетов, которые используются приложениями. Метод основан на сравнительном анализе суперкомпьютерных приложений с использованием статических данных. Автором также было проведено экспериментальное исследование предложенного метода.
4. Voevodin, V. V. How to Assess the Quality of Supercomputer Resource Usage / V. V. Voevodin, D. I. Shaikhislamov, D. A. Nikitenko // Supercomputing Frontiers and Innovations. — 2022. — vol. 9, No. 3. — pp. 4-18. — EDN VKETHG. — (Scopus Q4, Импакт-фактор 0.138 (SJR))[2.0625/0.55 п.л.]
Автором был самостоятельно разработан метод предсказания оценок качества использования суперкомпьютерных ресурсов приложениями, который основан на сравнительном анализе суперкомпьютерных приложений с использованием статических и динамических данных. Автором также было проведено экспериментальное исследование предложенного метода.

*Шайхисламов Денис Ильгизович*

Исследование и разработка методов для сравнительного анализа суперкомпьютерных приложений на основе технологий интеллектуального анализа данных

Автореф. дис. на соискание ученой степени канд. физ.-мат. наук

Подписано в печать ___.___._____. Заказ № _____
Формат 60×90/16. Усл. печ. л. 1. Тираж 100 экз.
Типография _____