На правах рукописи

Баранов Антон Викторович

МЕТОДЫ И СРЕДСТВА УПРАВЛЕНИЯ ВЫЧИСЛИТЕЛЬНЫМИ РЕСУРСАМИ В СУПЕРКОМПЬЮТЕРНЫХ СИСТЕМАХ КОЛЛЕКТИВНОГО ПОЛЬЗОВАНИЯ

2.3.5 – Математическое и программное обеспечение вычислительных систем, комплексов и компьютерных сетей

АВТОРЕФЕРАТ
диссертации на соискание ученой степени
доктора технических наук

Москва – 2025

Работа выполнена в Федеральном государственном бюджетном учреждении «Национальный исследовательский центр «Курчатовский институт»

Официальные оппоненты:

Марченко Михаил Александрович
доктор физико-математических наук, профессор РАН,
Институт вычислительной математики и математической геофизики Сибирского отделения РАН
(ИВМиМГ СО РАН), директор

Топорков Виктор Васильевич
доктор технических наук, профессор,
ФГБОУ ВО «НИУ «МЭИ»,
заведующий кафедрой вычислительных технологий

Фельдман Владимир Марткович
доктор технических наук, профессор,
АО «МЦСТ»,
главный научный сотрудник

Ведущая организация: Акционерное общество «Институт точной механики и вычислительной техники имени С.А. Лебедева Российской академии наук» (АО «ИТМиВТ»)

Защита диссертации состоится «17» декабря 2025 г. в 10 ч. 00 мин. на заседании диссертационного совета 24.1.224.04 при Федеральном исследовательском центре «Информатика и управление» Российской академии наук по адресу: 119333, г. Москва, ул. Вавилова, д.44, кор.2.

С диссертацией можно ознакомиться в библиотеке ФИУ ИУ РАН по адресу: 119333, г. Москва, ул. Вавилова, д. 44, корп. 2 и на официальном сайту ФИЦ ИУ РАН http://www.frccsc.ru.

Отзывы на автореферат в двух экземплярах, заверенные печатью учреждения, высылать по адресу: 119333, г. Москва, ул. Вавилова, д. 44, корп. 2, ученому секретарю диссертационного совета 24.1.224.04.

Автореферат разослан «_____» _____ 2025 г.

Ученый секретарь
диссертационного совета 24.1.224.04 Р.В. Разумчик

Общая характеристика работы

Актуальность темы исследования

Суперкомпьютеры в 21 веке прочно вошли в практику научных исследований, как неотъемлемый инструмент сверхточного моделирования сложных природных явлений и общественных процессов. Примерами могут служить задачи гидродинамики, физики высоких энергий, астрофизические, биологические, медицинские и фармакологические исследования, наноэлектроника и синтез новых материалов, климатология, науки о Земле и океане, энергетика, а также задачи машинного обучения и искусственного интеллекта. С каждым годом сфера применения суперкомпьютерных технологий расширяется, растет сложность фундаментальных и прикладных вычислительных задач, увеличивается число пользователей суперкомпьютерных систем.

Уровень развития суперкомпьютерных технологий является для государства одним из факторов стратегического значения. Мировые лидеры в области науки, образования, промышленности и бизнеса – США, КНР, Европейский Союз, Япония – не первое десятилетие реализуют национальные и наднациональные проекты развития суперкомпьютерных технологий. Целями подобных проектов являются получение конкурентных преимуществ за счет сокращения сроков научных исследований и разработок промышленных изделий, а также достижение технологического лидерства в таких областях, как создание новых материалов, энергетика, транспорт, медицина, обеспечение безопасности государства.

Создание инфраструктуры и условий для проведения научных исследований и разработок, внедрения наукоемких технологий, отвечающих современным принципам организации научной, научно-технической и инновационной деятельности, на основе лучших российских и мировых практик является одной из основных задач Стратегии научно-технологического развития Российской Федерации. Современная научная инфраструктура немыслима без применения высокопроизводительных вычислительных систем, на основе которых реализуется такой приоритет научно-технологического развития, как переход к передовым технологиям проектирования и создания высокотехнологичной продукции. Примерами развития и использования научной инфраструктуры мирового уровня являются суперкомпьютерные центры коллективного пользования МГУ им. М.В. Ломоносова, НИЦ «Курчатовский институт», Российской академии наук, Объединенного института ядерных исследований.

В суперкомпьютерных центрах высокопроизводительные вычислительные системы, как дорогостоящее научное оборудование, используются главным образом в режиме коллективного пользования. Для производства высокопроизводительных расчетов пользователи формируют задания, каждое из которых включает расчетную программу, входные данные и требования к ресурсам. Входной поток заданий последовательно проходит технологические этапы обработки, первоначально поступая в очередь одной из территориально распределенных суперкомпьютерных систем. После прохождения очереди каждому заданию выделяются вычислительные узлы суперкомпьютера, внутри которых запускаются процессы расчетной программы. В каждом узле эти процессы распределяются по вычислительным ядрам, а внутри ядра распараллеливаются выполняемые процессом отдельные вычислительные операции.

Таким образом, распараллеливание входного потока заданий осуществляется одновременно на нескольких иерархических уровнях, соответствующих этапам единой технологической цепочки обработки. Общая эффективность использования су-

перкомпьютерных систем очевидным образом зависит от эффективности управления вычислительными ресурсами на каждом технологическом этапе обработки (уровне распараллеливания) входного потока заданий. Постоянное усложнение архитектуры суперкомпьютеров, применение для их построения новейших технических решений, увеличение числа процессорных узлов и ядер, повышение степени их разнородности обуславливают актуальность развития существующих и создания новых методов и средств управления вычислительными ресурсами суперкомпьютерных систем коллективного пользования на разных уровнях распараллеливания входного потока заданий. При этом важно не только достигнуть эффективного распределения вычислительной работы по имеющимся ресурсам, но и обеспечить надежность и отказоустойчивость параллельных вычислений, поскольку рост степени распараллеливания ведет к соответствующему росту числа сбоев и отказов.

Поскольку доступ пользователей к вычислительным ресурсам суперкомпьютера обеспечивается посредством заданий, многие методы и алгоритмы управления этими ресурсами реализуются в виде специальных программных систем управления заданиями. Как отдельный вид системного программного обеспечения системы управления заданиями начали формироваться в середине 1990-х годов, к настоящему времени лидирующие позиции заняли такие системы, как SLURM, IBM Platform LSF, Portable Batch System (PBS), Grid Engine. Перечисленные системы-лидеры активно развиваются, пополняя свой арсенал новыми функциональными возможностями. Несмотря на то, что часть ведущих систем, подобно SLURM, свободно распространяются в открытых исходных кодах, разработка и развитие этих систем осуществляются западными компаниями. Необходимость сохранения компетенций и научно-технологического паритета в области управления суперкомпьютерными ресурсами обуславливает актуальность исследований и разработок по созданию и развитию отечественных систем управления заданиями, обладающих набором функциональных возможностей, соответствующим сложившейся мировой практике.

Центральной функцией любой системы управления заданиями является их планирование, которое заключается в организации одной или нескольких очередей поступивших заданий и формировании расписаний их запусков. Качество планирования оценивается рядом показателей, таких как загрузка вычислительных ресурсов, среднее время ожидания задания в очереди, средний коэффициент замедления заданий и другими. Показатели качества планирования являются взаимосвязанными и противоречивыми, оптимизация системы по одному показателю часто приводит к ухудшению значений другого. Разнообразие решаемых вычислительных задач, расширение применения облачных вычислений и связанных с ними технологий виртуализации и контейнеризации приводит к появлению разнородных потоков заданий на входе суперкомпьютерной системы, что обуславливает актуальность исследований и разработок методов и средств повышения качества планирования заданий путем эффективного совмещения разнородных потоков заданий и поиска баланса между противоречивыми показателями качества.

Не менее важной функцией системы управления является распределение процессов расчетной программы по вычислительным узлам суперкомпьютера, осуществляемое для каждого задания. Для того чтобы сократить время выполнения задания, необходимо найти оптимальное отображение информационного графа расчетной программы на граф вычислительных узлов. В условиях режима коллективного пользования оба графа, как правило, неизвестны заранее, и задача поиска оптимального отоб-

ражения должна решаться системой управления при каждом запуске задания и за ограниченное время. Эта задача в общем случае является NP-полной, ее сложность многократно возрастает в связи с непрерывным ростом числа процессорных ядер и вычислительных узлов в суперкомпьютерных системах, что обуславливает актуальность исследований и разработок методов и алгоритмов решения задачи поиска оптимального отображения средствами системы управления заданиями за приемлемое время.

Повышение доступности и универсальности суперкомпьютеров расширяет круг пользователей. При этом с одной стороны, в распоряжении пользователя оказывается суперкомпьютерная система со сложной иерархической структурой, эффективное использование которой требует от него специализированных знаний и навыков в области организации параллельных вычислений. С другой стороны, возрастает доля пользователей, для которых организация параллельных вычислений не является основной профессиональной специализацией и требует существенных дополнительных трудозатрат. Наиболее ярко это противоречие проявляется для вычислительных задач с распараллеливанием по данным, где подготовка заданий и организация параллельных вычислений носят рутинный характер. Автоматизация распараллеливания по данным сопряжена с высокими накладными расходами, обуславливаемыми главным образом необходимостью учета обработанных порций данных. Исследование и разработка методов и средств, обеспечивающих распараллеливание по данным с меньшими по сравнению с известными решениями накладными расходами, также является актуальной научной задачей.

Таким образом, каждый этап обработки входного потока заданий требует поиска и внедрения новых научно обоснованных архитектурных, технических и технологических решений по повышению эффективности использования суперкомпьютерных систем.

Существенное влияние на развитие суперкомпьютерных технологий и их применение, включая создание, эксплуатацию и развитие суперкомпьютерных систем и центров коллективного пользования, создание математических методов, алгоритмов, системного, инструментального и прикладного программного обеспечения для эффективного управления суперкомпьютерными ресурсами и организации высокопроизводительных вычислений, оказали работы известных советских и российских ученых Г.И. Савина, В.К. Левина, И.А. Соколова, Б.Н. Четверушкина, А.И. Аветисяна, Б.М. Шабанова, Вл.В. Воеводина, М.В. Якобовского, В.Ф. Тюрина, А.Н. Томилина, В.В. Корнеева, А.О. Лациса, В.В. Топоркова, Н.Н. Миренкова, В.В. Коренькова.

Цель диссертационной работы состоит в повышении эффективности использования суперкомпьютерных систем коллективного пользования за счет разработки комплекса архитектурных, технических и технологических решений для управления вычислительными ресурсами.

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

− разработать иерархическую модель управления вычислительными ресурсами в суперкомпьютерной системе коллективного пользования, построить архитектуру и создать систему управления заданиями, обладающую качественными и количественными характеристиками на уровне мировых систем-лидеров;

− разработать методы и средства планирования заданий, обеспечивающие распределение по вычислительным ресурсам суперкомпьютера разнородных потоков пользовательских заданий различных классов;

− разработать решения по повышению быстродействия прикладных программ за счет оптимизации их отображения на структуру вычислительных узлов суперкомпьютера;

− разработать методы и технические решения организации параллельных вычислений с распараллеливанием по данным.

Методы исследования

Результаты диссертации были получены с привлечением моделей и методов, используемых при поиске архитектурных и системных решений. Математическую основу исследования составляют методы теории алгоритмов, математической логики, теории графов.

Основные положения, выносимые на защиту

1. Метод планирования, совмещающий обработку классов ординарных, отладочных и фоновых заданий, позволяет снизить средний коэффициент замедления для отладочных и фоновых заданий при сохранении высокой загрузки вычислительных ресурсов;

2. Метод постпланирования, совмещающий поток заданий с фиксированными параметрами и поток адаптивных заданий, повышает загрузку суперкомпьютерной системы и минимизирует коэффициент замедления адаптивных заданий;

3. Двухэтапный метод и алгоритмы отображения параллельной программы на вычислительные узлы суперкомпьютера позволяют ускорить высокопроизводительные расчеты, повышают точность и быстродействие поиска отображения по сравнению с известными методами и алгоритмами;

4. Метод иерархического деления данных позволяет организовать параллельные вычисления с распараллеливанием по данным с меньшими накладными расходами на распараллеливание по сравнению с известными решениями;

5. Система управления прохождением параллельных заданий составляет основу информационно-вычислительной среды (цифровой экосистемы) проведения высокопроизводительных научных расчетов в суперкомпьютерных центрах коллективного пользования, обеспечивает качество решения задач управления вычислительными ресурсами, соответствующее мировому уровню.

Научная новизна

− Разработаны новые методы планирования, совмещающие обработку классов ординарных, отладочных, фоновых и адаптивных заданий, в том числе представленных в виде виртуальных машин и контейнеров;

− Разработан новый двухэтапный метод отображения параллельной программы на вычислительные узлы суперкомпьютера: на первом этапе производится выделение вычислительных узлов для параллельной программы и тем самым сокращается размер задачи отображения, на втором этапе выделенные узлы используются для выполнения параллельного алгоритма поиска отображения;

− Разработан новый эвристический алгоритм выделения вычислительных узлов для задания, основанный на разрезании графа свободных вычислительных узлов на два минимально связанных друг с другом подграфа при помощи имитации отжига;

− Разработан новый параллельный алгоритм поиска отображения программного графа на граф вычислительных узлов с использованием циклической смены фаз имитации отжига и генетического отбора;

− Разработан новый метод иерархического деления данных для организации параллельных вычислений с распараллеливанием по данным, за счет разделения входного пула данных на наборы упорядоченных элементарных вычислительных работ обеспечивающий компактное представление состояния вычислений и низкие накладные расходы на распараллеливание;

− Разработана новая архитектура системы управления заданиями пользователей, основанная на иерархической модели управления вычислительными ресурсами суперкомпьютерной системы коллективного пользования.

Диссертационное исследование **соответствует следующим пунктам паспорта специальности** 2.3.5 «Математическое и программное обеспечение вычислительных систем, комплексов и компьютерных сетей»:

− модели, методы, архитектуры, алгоритмы, языки и программные инструменты организации взаимодействия программ и программных систем;

− модели и методы создания программ и программных систем для параллельной и распределенной обработки данных, языки и инструментальные средства параллельного программирования;

− модели, методы, алгоритмы, облачные технологии и программная инфраструктура организации глобально распределенной обработки данных.

Теоретическая и практическая значимость

Практическая значимость диссертации определяется тем, что разработанный и реализованный комплекс решений по управлению вычислительными ресурсами суперкомпьютерных систем коллективного пользования в течение десятилетий обеспечивает проведение высокопроизводительных расчетов в суперкомпьютерных центрах коллективного пользования МСЦ РАН – НИЦ «Курчатовский институт», ИПМ им. М.В. Келдыша РАН для пользователей-исследователей из 135 научных и образовательных организаций. Разработанные эвристические алгоритмы разрезания графа свободных узлов и поиска отображения программного графа на граф вычислительных узлов суперкомпьютера вносят вклад в решение фундаментальной квадратичной задачи о назначениях. Теоретические положения и накопленный практический опыт, представленные в диссертации, могут служить основой дальнейшего развития методов и средств построения систем управления вычислительными ресурсами и организации параллельных вычислений в суперкомпьютерных центрах.

Достоверность полученных результатов подтверждается реализацией разработанных решений в составе системы управления прохождением параллельных заданий и положительным опытом ее практического применения для управления вычислительными ресурсами отечественных суперкомпьютерных систем МВС-1000, МВС-1000/16, МВС-1000/32, МВС-1000М, МВС-15000ВМ, МВС-6000IM, МВС-100К, МВС-10П, МВС-Экспресс, К-100, К-60. Эффективность предложенных автором методов и средств подтверждается данными статистики использования указанных высокопроизводительных систем, результатами сравнительных вычислительных экспериментов и имитационного моделирования.

Апробация результатов работы осуществлена на 30 международных и всероссийских научных конференциях, среди которых:

- Всероссийская научная конференция «Высокопроизводительные вычисления и их приложения», 30 октября - 2 ноября 2000 года, Черноголовка;
- Всероссийская научная конференция «Научный сервис в сети Интернет: суперкомпьютерные центры и задачи», 20-25 сентября 2010 г., Новороссийск;
- Всероссийская научная конференция «Научный сервис в сети Интернет: поиск новых решений», 17-22 сентября 2012 г., Новороссийск;
- Всероссийская научная конференция «Научный сервис в сети Интернет: все грани параллелизма», 23-28 сентября 2013 г., Новороссийск;
- Всероссийская научная конференция «Научный сервис в сети Интернет: многообразие суперкомпьютерных миров», 22-27 сентября 2014 г., Новороссийск;
- Всероссийская научно-техническая конференция «Суперкомпьютерные технологии СКТ-2014», 29 сентября - 4 октября 2014 г., Дивноморское, Геленджик;
- Национальный Суперкомпьютерный Форум (НСКФ-2015), 24-27 ноября 2015 г., Переславль-Залесский;
- Международная конференция «Суперкомпьютерные дни в России», 26-27 сентября 2016 г., Москва;
- 14th International Conference on Parallel Computing Technologies, September 4-8, 2017, Nizhni Novgorod, Russia;
- 2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus), 29 января - 1 февраля 2018 г., Москва;
- Национальный Суперкомпьютерный Форум (НСКФ-2018), 27-30 ноября 2018 г., Переславль-Залесский;
- 2019 Federated Conference on Computer Science and Information Systems, FedCSIS 2019, 01-04 сентября 2019 г., Leipzig;
- 5-я Международная научно-практическая конференция «Информационные технологии и высокопроизводительные вычисления», 16-19 сентября 2019 г., Хабаровск;
- 13th International Scientific Conference «Parallel computational technologies (PCT) 2020», March 31 - April 2, 2020, Perm, Russia;
- 16th International Conference «Parallel Computing Technologies (PaCT 2021)», September 13–18, 2021, Kaliningrad, Russia;
- Международная конференция «Суперкомпьютерные дни в России», 23-24 сентября 2024 г., Москва.

Личный вклад

Все выносимые на защиту результаты получены автором лично. В основных публикациях по теме диссертации автору принадлежат: архитектурные и технические решения по построению и развитию системы управления прохождением параллельных заданий [15, 22, 30, 32-34, 41, 44, 45], методика проведения сравнительного анализа с ведущими системами управления заданиями [36, 37], двухэтапный метод и эвристические параллельные алгоритмы поиска отображения программного графа на граф вычислительных узлов суперкомпьютера, а также методика проведения вычислительных экспериментов по исследованию характеристик предложенных метода и алгоритмов отображения [3, 5, 6], двухуровневая архитектура управления вычислительными ресурсами в территориально распределенной сети суперкомпьютерных

центров как составная часть иерархической модели управления вычислительными ресурсами суперкомпьютерных систем коллективного пользования [17-19, 24], метод иерархического деления данных в системах распараллеливания по данным и методика экспериментального сравнения программных комплексов распараллеливания по данным [7, 12, 35, 38, 39], методы представления заданий в виде виртуальных машин и контейнеров, а также методы совмещения потоков стандартных заданий и заданий, представленных в виде виртуальных машин и контейнеров [9, 14, 20, 23, 28, 40, 42, 43], метод постпланирования адаптивных заданий и методика имитационного моделирования для исследования характеристик этого метода [4, 11, 13, 21, 25, 27], постановка научной задачи прогнозирования времени выполнения заданий [10, 16, 26].

Публикации

По теме диссертации автором опубликовано 45 печатных работ [1-45], из них 27 работ [1-27] опубликованы в журналах, рекомендованных ВАК, в том числе 20 работ [1, 2, 4-6, 8-11, 13-15, 17, 19-21, 24-27] – в журналах, входящих в индекс RSCI, 4 работы [1, 5, 10, 13] – в журналах, входящих в список К1 перечня ВАК, 15 работ [2-4, 6, 7, 12, 14-16, 18, 22, 23, 25-27] – в журналах, индексируемых в международных базах научных изданий Web of Science и Scopus. По теме диссертации получено 6 свидетельств о государственной регистрации программ для ЭВМ и баз данных.

Структура и объем диссертации

Диссертация состоит из введения, пяти глав, заключения и списка литературы. Общий объем диссертации – 295 страниц, в том числе 79 рисунков и 22 таблицы. Список литературы состоит из 242 наименований.

ОСНОВНОЕ СОДЕРЖАНИЕ РАБОТЫ

Во введении обоснована актуальность работы, сформулированы цель и задачи работы, приведены научная новизна, практическая значимость полученных результатов и защищаемые положения, рассмотрена структура диссертации.

Первая глава посвящена построению архитектуры системы управления заданиями (СУЗ), основанной на иерархической модели вычислительными ресурсами суперкомпьютерных систем коллективного пользования.

В п. 1.1 представлена типовая архитектура суперкомпьютерной системы коллективного пользования, включающая решающее поле (вычислитель) как совокупность вычислительных узлов (ВУ), объединенных коммуникационной средой. Каждый ВУ представляет собой автономный компьютер (многопроцессорный сервер), управляемый собственной операционной системой, имеющий уникальное сетевое имя и взаимодействующий с остальными ВУ через коммуникационную среду. В типовой архитектуре выделяются также сервер доступа и управляющая ЭВМ, обеспечивающие удаленный доступ пользователей к системе и функции управления множеством ВУ соответственно. Как правило, в суперкомпьютерном центре организуется единая система хранения данных, одинаково доступная по сети из сервера доступа, управляющей ЭВМ и любого ВУ.

В п. 1.2 введено понятие «задание» (англ. – job) как специальный информационный объект, включающий:

– параллельную программу, реализующую прикладной алгоритм и состоящую из нескольких взаимодействующих процессов или потоков, которые могут одновременно выполняться на нескольких ВУ или процессорных ядрах;

– требования к параллельному ресурсу, под которым понимается подмножество ВУ, выделяемое заданию на определенное время, при этом величина параллельного ресурса (число ВУ и время выполнения) называется размером задания или параллельного ресурса;

– входные данные параллельной программы.

Жизненный цикл задания включает этапы создания задания, направления задания в очередь СУЗ, нахождения в очереди, инициализации требуемых вычислительных ресурсов, запуска и выполнения параллельной программы, завершения задания.

В п. 1.3 рассмотрены следующие основные задачи управления вычислительными ресурсами суперкомпьютерной системы.

1. Подготовка программ и данных, оформление соответствующего им задания в виде паспорта.

2. Прием входного потока заданий, их планирование в одной или нескольких очередях.

3. Выделение и конфигурирование требуемых для задания вычислительных ресурсов.

4. Запуск задания, поддержка и контроль его выполнения.

5. Освобождение занятых ресурсов по завершении задания.

6. Непрерывный мониторинг состояния вычислительных ресурсов суперкомпьютера и учет потребления пользовательскими заданиями вычислительных ресурсов суперкомпьютера.

Первая задача, как правило, возлагается на пользователя, решением остальных занимается система управления заданиями.

В п. 1.4 сформулированы основные требования к СУЗ [30]:

− универсальность, как отсутствие ограничений на выполняемые пользователями прикладные программы, а также как возможность функционирования СУЗ на произвольной вычислительной системе, соответствующей типовой архитектуре суперкомпьютера;

− с универсальностью связаны требования надежности и императивности управления, как техническая невозможность совершения пользователем неразрешенных в системе действий; во многом эти требования выполняются за счет изоляции заданий, сводящей к минимуму потенциальное влияние одних заданий на выполнение других;

− круглосуточная доступность, автоматическое функционирование с организацией режимов профилактики и обслуживания;

− прозрачность алгоритмов планирования и справедливое распределение вычислительных ресурсов, под которым понимается невозможность длительного захвата одним пользователем большого объема вычислительных ресурсов;

− автоматизация формирования типового программного окружения, под которым понимается автоматическая настройка заданий пользователя на выбранный типовой стек системного, инструментального и прикладного программного обеспечения;

− модульность архитектуры СУЗ, позволяющая изменять функционал системы за счет добавления или замены программных модулей; в совокупности с универсальностью модульность архитектуры обеспечивает в том числе переносимость СУЗ между различными суперкомпьютерами типовой архитектуры;

− возможность реконфигурации в процессе функционирования, включающая динамическое изменение состава вычислительных узлов суперкомпьютера и связанную с этим виртуализацию представления параллельного ресурса для пользователя.

В п. 1.5 приведен краткий обзор ведущих мировых СУЗ: PBS (Portable Batch System), IBM Spectrum LSF, Moab HPC Suite, Grid Engine, Windows HPC Server и SLURM. Отмечены отечественные разработки в этой области: системы Cleo и Slurm-ВНИИТФ.

Параграф 1.6 посвящен построению иерархической модели управления вычислительными ресурсами суперкомпьютерной системы коллективного пользования. В основу положена представленная в монографии Н.Н. Миренкова[1] четырехуровневая модель управления многопроцессорной вычислительной системой. Модель Миренкова в диссертации была адаптирована по отношению к современным суперкомпьютерным системам: в новой модели учтены наличие множества ядер в процессорах, автономность вычислительных узлов, появление распределенных вычислительных сред.

Первый уровень адаптированной модели – уровень пользователя, который выбирает алгоритмы для своих задач, средства для записи алгоритмов, их анализа, отладки и т.п. В современных суперкомпьютерах это уровень процесса (потока) пользовательской программы, выполняющейся на отдельном процессорном ядре вычислительного узла. На втором уровне принимаются локальные, автономные решения по распределению ресурсов в рамках одного или нескольких вычислительных узлов. На третьем уровне принимаются решения, которые связаны с управлением подсистемой вычислительных узлов. На четвёртом уровне принимаются системные решения, изменяющие состояние вычислений отдельной суперкомпьютерной системы. Эти решения определяются приоритетами режимов функционирования, очередью заданий, состоянием ресурсов системы, директивами оператора и т.п.

В диссертации исходная модель дополнена пятым уровнем иерархии, на котором представлены средства управления несколькими параллельными компьютерами, объединенными в единую распределенную вычислительную систему (РВС). На этом уровне принимаются глобальные решения по распределению вычислительных работ по очередям суперкомпьютерных систем, входящих в РВС.

На основе иерархической модели в п.1.7 предложена архитектура Системы управления прохождением параллельных заданий (СУППЗ) [1], представленная на рисунке 1. Архитектурно СУППЗ можно разделить на ядро и надстройки. Ядро включает планировщик (сервер очереди) и служебные процессы: клиентское приложение, серверы запросов и запуска и менеджеры заданий. Надстройками являются подсистемы сценариев пользовательских команд, команд и утилит оператора и системного программиста, сценариев управления заданием, сценариев диагностики, выделения и освобождения узлов.

В п. 1.8 показано, что предложенная архитектура СУППЗ соответствует сформулированным требованиям.

Требование универсальности выполняется за счет выделения в архитектуре СУППЗ ядра системы, для которого задание представляет собой абстрактный информационный объект, запрашивающий некоторый виртуальный параллельный ресурс. Задача ядра – обеспечить выделение реального физического параллельного ресурса

[1] Миренков Н.Н. Параллельное программирование для многомодульных вычислительных систем. М.: Радио и связь, 1989. 320 с.

для нужд задания. Вид и характер пользовательских прикладных программ, которые будут выполнены на выделенном параллельном ресурсе, полностью определяются соответствующими надстройками подготовки заданий. За счет надстроек, реализованных в виде переносимых командных сценариев, также достигается возможность функционирования СУППЗ на широком спектре суперкомпьютеров, соответствующих типовой архитектуре.

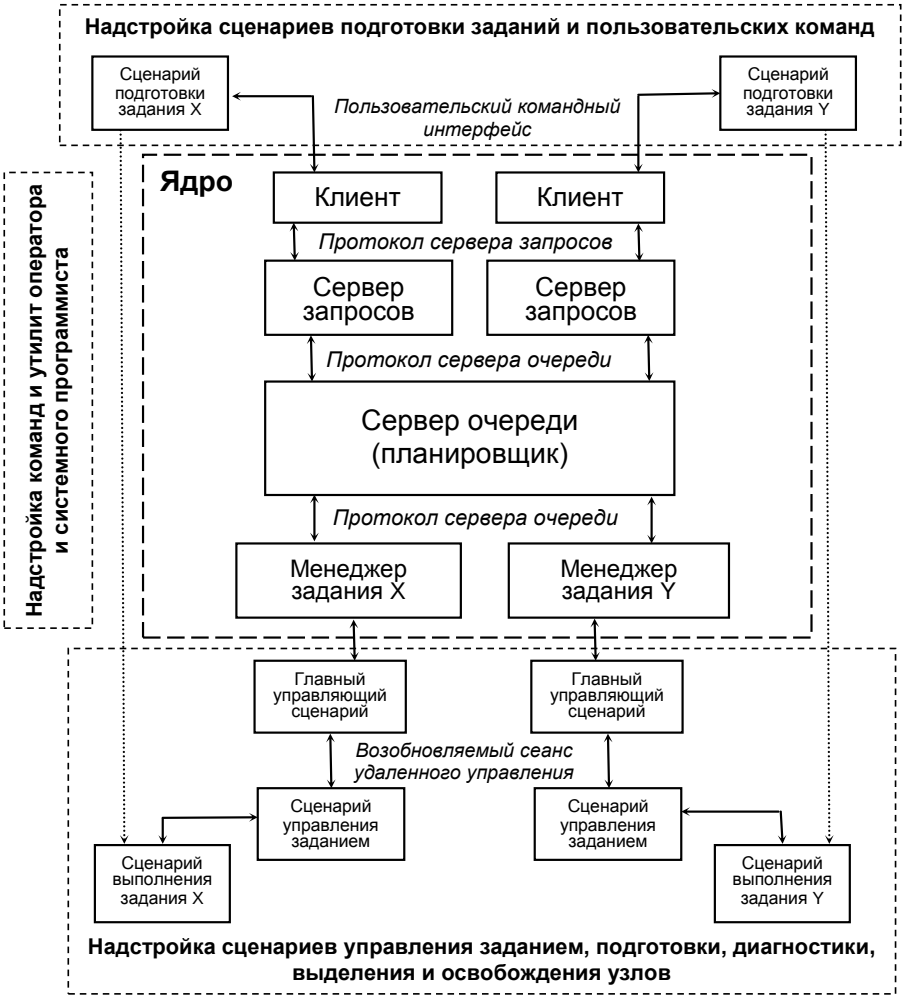

Рис. 1. Архитектура СУППЗ

Подтверждением является множество функционировавших под управлением СУППЗ вычислительных систем разной архитектуры, в том числе:

- массивно-параллельная система МВС-1000 на базе процессоров DEC Alpha;
- кластерная система МВС-1000М на базе процессоров DEC Alpha;
- кластерная система МВС-15000 на базе процессоров Power;
- кластерная система МВС-100К на базе процессоров x86;
- кластерная система МВС-6000 на базе процессоров Itanium;
- гибридная система МВС-10П с ускорителями Intel Xeon Phi KNC;
- кластерная система МВС-10П ОП KNL на базе процессоров Intel Xeon Phi KNL;
- гибридная система К100 с графическими ускорителями Nvidia.

ВУ перечисленных систем объединялись различными коммуникационными средами, в том числе сетями линков на базе сигнальных процессоров, интерконнектами Myrinet, Infiniband, OmniPath, а также специализированной сетью МВС-Экспресс.

Надёжность в СУППЗ обеспечивается за счет принципа делегирования функций управления ресурсами высоконадежным компонентам и утилитам операционной системы с сохранением контроля за жизненным циклом заданий. В СУППЗ в качестве таких компонентов выступают сетевая файловая система (NFS) и ОС Linux на узлах. За счет сетевой файловой системы в СУППЗ реализован ряд критических функций управления, в том числе авторизация пользователей и ведение оперативной базы данных статусов ВУ. Делегирование функций управления позволяет, в отличие от SLURM, LSF и PBS, отказаться от наличия активных компонентов СУППЗ на ВУ, что повышает надежность и открывает возможность применения сторонних систем для управления ресурсами.

Императивность управления обеспечивается за счет выделенной в архитектуре надстройки сценариев подготовки и освобождения ВУ, ориентированных на обеспечение максимальной взаимной изоляции заданий. Доступ пользователю и его программе предоставляется только к выделенным для его задания ВУ и только на время выполнения задания, при этом один и тот же ВУ не может быть одновременно предоставлен двум разным заданиям. После завершения задания производится принудительная очистка каждого ВУ от процессов пользователя, а в случае невозможности очистки ВУ помечается как неисправный и выводится из состава решающего поля.

Автоматическое функционирование и круглосуточная доступность обеспечиваются за счет выделения надстройки сценариев управления заданием, подготовки, диагностики, выделения и освобождения ВУ. Сценарии диагностики автоматически выявляют возникающие неисправности ВУ и выводят отказавшие узлы из состава решающего поля, очередь заданий при этом автоматически перепланируется.

Автоматическая организация режимов профилактики, а также прозрачность алгоритмов планирования и справедливое распределение вычислительных ресурсов обеспечиваются методом планирования заданий за счет применения шкал доступа, определяющих возможности пользователей и системных администраторов по доступу к решающему полю в определенное время.

Автоматизация формирования стандартного программного окружения обеспечивается наличием в архитектуре СУППЗ надстройки сценариев подготовки заданий, которая позволяет автоматизировать подготовку паспорта задания для некоторого типового программного окружения. Одной из таких групп сценариев, изначально входившей в состав СУППЗ, являются сценарии поддержки MPI-программ, которые, как показала практика, оказались наиболее востребованными пользователями и стали характерной особенностью СУППЗ.

Модульность архитектуры СУППЗ обеспечивается:

- соответствием архитектуры иерархической модели управления ресурсами;
- разделением на ядро, неизменяемые и изменяемые системным программистом надстройки;
- применением специфицированных протоколов взаимодействия компонентов.

Ядро СУППЗ, реализованное на языке высокого уровня, изменяется разработчиками сравнительно редко. Достаточно большое число функций управления и мониторинга реализовано в виде командных сценариев, что позволяет разработчикам при необходимости оперативно вносить изменения в систему, не затрагивая ядро. Нако-

нец, разграничение функций компонентов по уровням иерархии управления и применение специфицированных протоколов, таких как протоколы сервера запросов и сервера очереди, дало возможность подключения к СУППЗ ключевых сторонних компонентов. Примерами служат подключение планировщика Maui, а также организация совместной работы СУППЗ и таких систем, как SLURM и Sun Grid Engine.

Параграф 1.9 демонстрирует развитие архитектуры СУППЗ, начавшегося от первого варианта, разработанного в 1999 г. для массивно-параллельных суперкомпьютеров серии МВС-1000.

Во второй главе представлены технические и технологические решения, предложенные автором для управления вычислительными ресурсами суперкомпьютерных систем коллективного пользования на каждом уровне иерархической модели. Решения, составившие основу выносимых на защиту положений диссертации, подробно рассмотрены в главах 3-5. Параграф 2.1 посвящен обзору остальных решений, составляющих в совокупности комплекс функциональных возможностей СУППЗ.

Для первого уровня иерархии, на котором решения принимаются пользователем в его прикладной программе, рассмотрены результаты исследований в области автоматизации создания контрольных точек параллельного приложения [15]. В ходе исследований были сформулированы требования к средствам автоматического создания контрольных точек, разработана методика проведения эксперимента и проведено экспериментальное сравнение таких средств.

Для второго уровня иерархии, уровня управления отдельным ВУ, отражены результаты исследований и экспериментов в области виртуализации и контейнеризации пользовательских заданий [9, 14, 23, 28, 43]. За счет представления задания в виде контейнера возможно разрешение противоречий между исторически сложившейся цифровой экосистемой суперкомпьютерного центра и особым порядком работы тех категорий пользователей, для которых требуется индивидуальный стек программного обеспечения. Рассмотрены два метода контейнеризации заданий – с применением системного репозитория и с возможностью запуска контейнеров с произвольным содержимым.

На третьем уровне иерархии осуществляется управление подмножеством ВУ как подсистемой суперкомпьютера, выделенной определенному заданию. К предложенным техническим решениям для этого уровня иерархии относятся средства сопряжения СУППЗ и сторонних систем управления заданиями, позволившие в СУППЗ организовать совместное управление вычислительными ресурсами с Sun Grid Engine и SLURM. Сопряжение с Sun Grid Engine [43] стало возможно за счет применения метода контейнеризации заданий, основанного на системном репозитории контейнеров. Сопряжение со SLURM [44] имело целью использовать уникальные возможности средств управления уровней иерархии 1-3, реализованных в ядре SLURM, в частности, – средства управления микропроцессором Intel Xeon Phi KNL. Предложенный способ заключается в передаче прошедшего через очередь СУППЗ задания в пустую очередь SLURM, после чего задание запускается средствами SLURM на заданных со стороны СУППЗ узлах. При этом автоматически становятся доступны все средства SLURM для управления вычислительными ресурсами суперкомпьютера.

Четвертый уровень иерархии – уровень распределения подсистем суперкомпьютера между заданиями. К предложенным техническим и технологическим решениям на этом уровне относятся следующие:

− механизм логических подсистем СУППЗ, позволяющий организовать разбиение суперкомпьютера на разделы с отдельными очередями или стеками программного обеспечения;

− механизм многоресурсного планирования [32], позволяющий обслуживать гетерогенные суперкомпьютерные системы;

− средства обработки заданий с заданным уровнем обслуживания [45], позволяющие обеспечить выполнение заданий к заданному пользователем сроку;

− средства интеграции в состав СУППЗ сторонних планировщиков [33, 34], в частности, известного планировщика Maui.

Для пятого, высшего уровня иерархии проведены исследования по созданию территориально распределенной среды для высокопроизводительных вычислений [18, 19, 24, 31, 40]. К этому же уровню следует отнести разработки по созданию и развитию подсистемы сбора и обработки статистики СУППЗ [41] и POSIX-совместимого пользовательского интерфейса [32].

В п. 2.2 представлен сравнительный анализ качественных характеристик СУППЗ [36] и ведущих мировых систем: SLURM, PBS, OpenLava, Sun Grid Engine и IBM Spectrum LSF. Методика сравнения и данные о характеристиках других систем были взяты из фундаментального исследования коллектива Массачусетского технологического института[2]. Сравнение производилось по следующим категориям:

− общие характеристики: способ распространения, поддерживаемые ОС и языки программирования и другие;

− виды поддерживаемых заданий: параллельные задания, массивы заданий, ведение нескольких очередей и метапланирование;

− методы планирования заданий: поддержка обратного заполнения, пакетирования и упаковки заданий, группового планирования и другие;

− управление ресурсами: поддержка гетерогенных систем, отображения заданий на структуру ВУ, наличие политики выделения ресурсов и другие;

− управление очередью заданий: наличие механизма динамических приоритетов, возможности резервирования ресурсов и учета их энергопотребления и другие;

− управление выполнением заданий: поддержка сценариев пролога/эпилога, контрольных точек, перезапуска и вытеснения заданий и другие.

Проведенный сравнительный анализ показал, что СУППЗ обладает подавляющим большинством качественных характеристик ведущих систем, что позволяет говорить о соответствии СУППЗ мировому уровню развития систем управления заданиями.

В п. 2.3 введены основные количественные показатели эффективности СУЗ, к которым отнесены следующие.

1. Загрузка U вычислительных ресурсов суперкомпьютера за определённый интервал времени T определяется как

$$U(T) = \frac{M_{all}(T)}{N_{all}(T)}, \quad (1)$$

где $M_{all}(T)$ – размер параллельного ресурса, занятого заданиями за время T, а $N_{all}(T)$ – размер параллельного ресурса, доступного для заданий за время T.

[2] Reuther A. et al. Scalable system scheduling for HPC and big data // Journal of Parallel and Distributed Computing, 2018, vol. 111, pp. 76–92. doi: 10.1016/j.jpdc.2017.06.009

2. Среднее время ожидания задания в очереди *Q(T)* за интервал *T*:

$$Q(T) = \frac{\sum_{i=0}^{n} Q_i}{n}, \quad (2)$$

где *n* – число заданий, выполненных за интервал времени *T*, а Q_i – время ожидания *i*-го задания в очереди.

3. Средний коэффициент замедления *n* заданий за интервал времени *T*:

$$S(T) = \frac{1}{n}\sum_{i=1}^{n} \max(S_i, 1)\,,\ S_i = \frac{R_i + Q_i}{\max(R_i, \tau)}, \quad (3)$$

где R_i – время выполнения *i*-го задания. Параметр τ применяется для исключения из расчетов аварийно завершившихся заданий с аномально коротким временем выполнения, не превышающим τ.

Коэффициент замедления отражает время пребывания задания в системе относительно времени его выполнения, поскольку само по себе среднее время ожидания задания в очереди не является информативным показателем и может служить лишь для сравнения качества разных СУЗ на одном и том же входном потоке заданий. Поясним этот факт на следующем примере. Пусть два разных задания ожидали в очереди одно и то же время, например, 1 час. При этом первое задание выполнялось 10 часа, а второе – 10 минут. Очевидно, что время ожидания является достаточно малым для первого задания, поскольку составляет только 0,1 от времени его выполнения, и неприемлемым для второго задания, поскольку превышает время его выполнения в 6 раз. Минимальное значение коэффициента замедления равно 1, что соответствует немедленному запуску задания после его поступления без ожидания в очереди.

Руководство и администрация суперкомпьютерного центра обычно стремятся к максимизации загрузки вычислительных ресурсов, а пользователи заинтересованы в минимизации среднего коэффициента замедления.

4. Среднее значение *V(T)* времени нахождения задания в очереди относительно заказанного времени выполнения. Пусть B_i – заказанное пользователем время выполнения *i*-го задания. Тогда

$$V(T) = \frac{1}{n}\sum_{i=1}^{n} \frac{Q_i}{B_i}. \quad (4)$$

5. Относительная доля своевременно обработанных заданий $C_{своевр}$, расчет которой производится в соответствии с определенным в ГОСТ Р 59341 2021 критерием по среднему времени реакции. Для расчета показателя $C_{своевр}$ для каждого типа заданий должны быть известны интенсивности λ_i поступления в систему *i*-го типа заданий и пороговое значение $T_{зад\,i}$. Условие своевременности выполняется, если среднее время обработки заданий *i*-го типа T_i не более задаваемого $T_{зад\,i}$ порогового значения.

При постановке задания в очередь пользователи, как правило, не задают пороговых значений $T_{зад\,i}$ и указывают только максимальное время выполнения задания, на основании которого планировщик составляет расписание и прогнозирует время запуска каждого задания. О своевременности обработки задания можно сделать вывод по следующему признаку. Если пользователя не устраивает прогнозируемое время запуска его задания, то он удаляет это задание из очереди до начала его выполнения.

Пусть *i*-й пользователь за период времени *T* направил в систему n_i заданий. Тогда интенсивность поступления его заданий можно оценить как $\lambda_i = \frac{n_i}{T}$. Пусть d_i –

число заданий, который i-й пользователь за время T удалил из очереди до начала выполнения, т.е. то число заданий, для которых не было выполнено условие своевременности. Тогда вероятность $P_{\text{св }i}(T_{\text{зад }i})$ своевременного выполнения заданий i-го пользователя можно оценить как $P_{\text{св }i}(T_{\text{зад }i}) = 1 - \frac{d_i}{n_i}$.

Пусть I пользователей за время T направили в систему $n = \sum_{i=1}^{I} n_i$ заданий. Будем считать, что у каждого пользователя свои требования по своевременности обработки. В этом случае выполнение условия своевременности по среднему времени реакции для каждого пользователя будет определяться неявно подразумевающимся этим пользователем временем $T_{зад\,i}$. Определим величину d удаленных из очереди заданий за время T как $d = \sum_{i=1}^{I} d_i$. Тогда в соответствии с ГОСТ Р 59341 2021показатель $C_{своевр}$ приобретает следующий вид:

$$C_{\text{своевр}} = \frac{\sum_{i=1}^{I} \lambda_i P_{\text{св }i}(T_{\text{зад }i})}{\sum_{i=1}^{I} \lambda_i} = \frac{\sum_{i=1}^{I} \frac{n_i}{T}\left(1 - \frac{d_i}{n_i}\right)}{\sum_{i=1}^{I} \frac{n_i}{T}} = 1 - \frac{d}{n}. \quad (5)$$

Показатель $C_{своевр}$, рассчитанный в соответствии с (5), является частным случаем показателя доли своевременно обработанных запросов, определенного в ГОСТ Р 59341 2021.

В п. 2.4 приведены результаты сравнения количественных показателей СУППЗ и SLURM [37], которое производилось на статистических данных установленного в МСЦ РАН суперкомпьютера МВС-100К за период с 6 по 12 декабря 2012 года. В этот период под планирование заданий в системе было отведено 728 8-процессорных ВУ. Для определения показателей SLURM был использован симулятор, на основе которого был собран экспериментальный стенд с виртуальным вычислителем с параметрами, аналогичными кластеру МВС-100К. Для симулятора SLURM был сформирован модельный поток заданий по статистической информации СУППЗ за рассматриваемый период. Результаты сравнения приведены в таблице 1 и свидетельствуют о примерном паритете по количественным показателям между СУППЗ и SLURM.

Табл. 1. Показатели качества СУЗ SLURM и СУППЗ

Показатель	SLURM	СУППЗ
1	2	3
Загрузка ресурсов за исследуемый период в соответствии с (1)	0,94	0,97
Средний коэффициент замедления заданий за исследуемый период в соответствии с (3)	9,67	9,05
Среднее значение времени нахождения задания в очереди относительно заказанного времени счёта за исследуемый период в соответствии с (4)	0,34	0,23
Доля своевременно обработанных заданий за исследуемый период в соответствии с (5)	–	0,978
Число запущенных заданий за исследуемый период	13 500	13 748

В п. 2.5 рассмотрены результаты эксплуатации СУППЗ [1], которая осуществляется с 1999 года по настоящее время на множестве суперкомпьютерных систем МСЦ РАН [15] (ныне – отделения МСЦ Курчатовского высокопроизводительного вычислительного комплекса НИЦ «Курчатовский институт»), ИПМ им. М.В. Келдыша РАН. Кроме этого, СУППЗ применялась на ряде суперкомпьютерных систем, поставлявшихся НИИ «Квант» институтам РАН.

В диссертации подробно рассмотрены состав и характеристики суперкомпьютерных систем под управлением СУППЗ, среди которых следует выделить:

1. Первый российский суперкомпьютер МВС-1000М (МСЦ РАН) сферы науки и образования, перешагнувший терафлопсный рубеж производительности и занявший 64-е место в рейтинге Топ-500 самых производительных суперкомпьютеров мира.

2. Серия суперкомпьютеров МВС-1000/16 и МВС-1000/32 производства НИИ «Квант», поставленных в ИПМ им. М.В. Келдыша РАН, ИММ УрО РАН, МФТИ, Ивановский государственный энергетический университет им. В.И. Ленина, ИВМиМГ СО РАН, ИВМ СО РАН, ИАПУ ДВО РАН.

3. Суперкомпьютер МВС-15000 (МСЦ РАН) с пиковой производительностью 10,1 Тфлопс, занявший 56-е место в рейтинге Топ-500.

4. Суперкомпьютер МВС-100К (МСЦ РАН), занявший 38-е место в рейтинге Топ-500 с пиковой производительностью 227,94 Тфлопс.

5. Суперкомпьютер К100 был создан и установлен в ИПМ им. М.В. Келдыша РАН в 2010 году. Является первым отечественным суперкомпьютером гибридной архитектуры, содержащей графические ускорители в составе ВУ.

6. Суперкомпьютер МВС-10П Торнадо (МСЦ РАН) с пиковой производительностью 524 ТФлопс, занявший 59-е место в списке Топ-500.

7. Линейка суперкомпьютеров МВС-10П ОП, установленных в МСЦ РАН в виде основных разделов Broadwell, KNL, Skylake, Cascade Lake и Optane.

8. Суперкомпьютер К60, установленный в ИПМ им. М.В. Келдыша РАН.

Множество суперкомпьютеров под управлением СУППЗ охватывает широкий спектр архитектур, в том числе на базе: универсальных процессоров, ВУ с графическими ускорителями, ВУ с ускорителями линейки Intel Xeon Phi обоих поколений.

Диаграмма рассчитанной по формуле (1) загрузки некоторых перечисленных суперкомпьютеров за периоды их активной эксплуатации приведена на рисунке 2. Как видно, загрузка суперкомпьютеров превышает 90%.

Общее число выполненных под управлением СУППЗ заданий составляет более 3,7 млн. Общее число исследователей, воспользовавшихся услугами СУППЗ, составляет в МСЦ РАН свыше 1240 человек из 135 научных и образовательных организаций, а в ИПМ им. М.В. Келдыша РАН – свыше 280 человек. Общее число научных проектов, реализованных при помощи суперкомпьютеров МСЦ РАН, превышает 530, а реализованных при помощи суперкомпьютеров ИПМ им. М.В. Келдыша РАН – превышает 32.

Сложившаяся практика работы суперкомпьютеров под управлением СУППЗ позволяет говорить о формировании цифровой экосистемы научного суперкомпьютерного центра как цифрового пространства, построенного на базе СУППЗ и включающего в себя совокупность сервисов высокопроизводительных вычислений, которые позволяют пользователям удовлетворять потребности в рамках реализации процесса научных исследований. Экосистема суперкомпьютерного центра представляет собой динамичную совокупность научного оборудования

центра, его персонала, пользователей и отношений между ними. Во многом эти отношения определяются системой управления заданиями, на базе которой пользователи выстраивают инструменты и среды разработки для проведения своих исследований и через которую получают доступ к высокопроизводительным вычислениям.

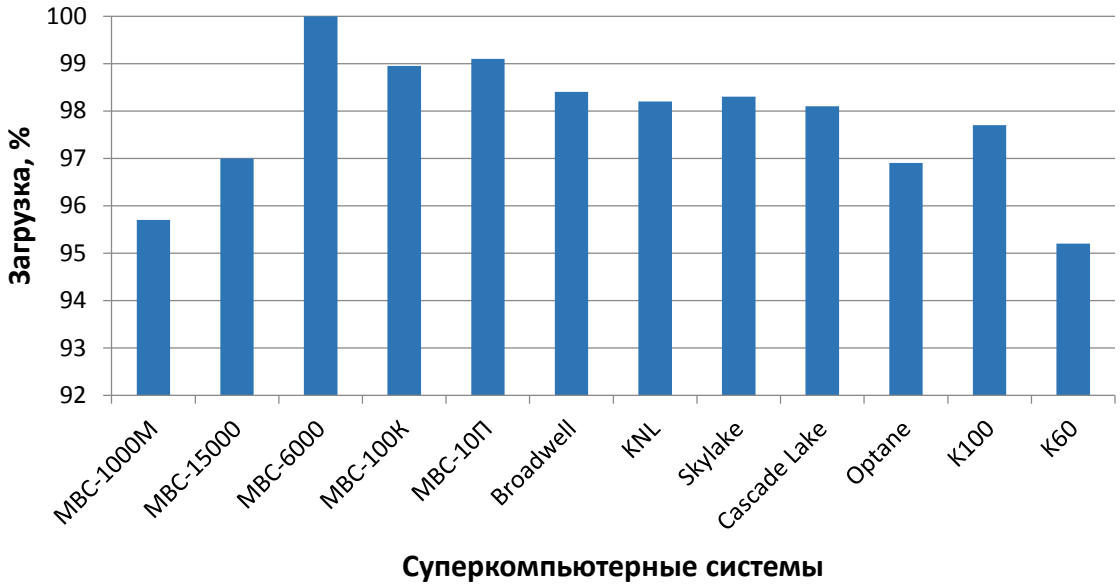

Рис. 2. Загрузка суперкомпьютеров под управлением СУППЗ

Третья глава посвящена методам планирования заданий в суперкомпьютерных системах коллективного пользования. Аналитический обзор существующих методов, алгоритмов и средств планирования приведен в п. 3.1.

В большинстве систем управления применяются методы планирования заданий с фиксированными параметрами. В таких методах планирование основывается на предоставляемых пользователем оценках необходимых для задания числа ВУ и времени выполнения, которые не изменяются на протяжении жизненного цикла задания. Для методов планирования заданий с фиксированными параметрами характерна невытесняющая приоритетная дисциплина обслуживания с применением принципов справедливого распределения ресурсов и обратного заполнения. Справедливое распределение ресурсов означает обратную зависимость приоритета пользователя от объема потребленных его заданиями ресурсов. Обратное заполнение призвано решить следующую проблему.

При планировании заданий с фиксированными параметрами в расписании запусков неизбежно образуются окна свободных вычислительных ресурсов, что снижает загрузку суперкомпьютера. Для борьбы с окнами планировщики применяют различные стратегии (алгоритмы) обратного заполнения, позволяющие заполнять окна менее приоритетными заданиями подходящего размера. Принцип обратного заполнения разрешает запуск вне очереди некоторых заданий, если этот запуск не повлияет на время старта заданий, стоящих в очереди выше. Такое возможно, например, в случае, если для задания А, стоящего в очереди выше, недостаточно ресурсов, и при этом задание Б, стоящее в очереди ниже, успеет завершиться до момента, когда освободится достаточное количество ресурсов для запуска задания А.

П. 3.2 посвящен исследованиям планировщиков заданий как систем массового обслуживания, преследующим цель построения и анализа математических моделей суперкомпьютерных систем коллективного пользования. По этому направлению отмечены успехи отечественной школы теории массового обслуживания, представленные в работах Б.В. Гнеденко, Г.П. Климова, В.Ф. Матвеева, В.А. Балыбердина, А.И. Костогрызова, В.М. Вишневского, А.С. Румянцева, Р.В. Разумчика и других ученых, которые постоянно совершенствуют математические модели, методы и средства исследования усложняющихся вычислительных машин и систем, в том числе суперкомпьютеров.

В работах ведущих специалистов отмечаются особенности, затрудняющие анализ суперкомпьютерных систем коллективного пользования при помощи методов теории массового обслуживания, такие как «тяжелые хвосты распределений», ограничивающие использование экспоненциальных распределений для моделирования процессов в современных компьютерных системах[3], и неконсервативность процесса нагрузки (узлы или процессоры суперкомпьютера могут простаивать при непустой очереди). Отмечается[4], что нахождение явных решений для характеристик производительности таких систем затруднено даже для систем малого размера.

Заявки в виде заданий в суперкомпьютерную систему, как правило, направляют люди – пользователи системы. Пользователи отслеживают текущие параметры планирования заданий и адаптируют требования своих заданий к изменениям этих параметров. Таким образом, мы получаем систему массового обслуживания с обратной связью, обусловленной в том числе человеческим фактором. Это обстоятельство еще сильнее усложняет анализ суперкомпьютерных систем коллективного пользования. В монографии[5] отмечается, что аналитическое или численное моделирование столь сложных систем либо затруднено, либо невозможно традиционными методами теории массового обслуживания, что заставляет исследователей прибегать к методам имитационного моделирования и машинного обучения.

В п. 3.3 предложен метод планирования заданий с фиксированными параметрами [2]. В одной очереди часто оказываются задания со значительно различающимися параметрами по числу ВУ и времени выполнения, что делает входной поток заданий существенно разнородным. Возникает необходимость выделения во входном потоке различных классов заданий и учета класса задания при планировании.

Идея метода, определяющая его новизну, состоит в разделении входного потока на классы ординарных (пакетных), отладочных и фоновых заданий. Ординарное задание может быть запущено не более одного раза на ограниченное время. Отладочное задание предполагает малое время выполнения на небольшом числе ВУ. Фоновое задание может выполняться произвольное время, но при этом прерываться системой управления с возвратом в очередь. Метод использует приоритетную невытесняющую дисциплину обслуживания с реализацией принципа справедливого распределения и применением консервативной стратегии обратного заполнения.

---

[3] Морозов Е.В., Румянцев А.С. Вероятностные модели многопроцессорных систем: стационарность и моментные свойства // Информатика и ее применения, 2012, Т. 6, № 3, с. 99-106.

[4] Разумчик Р.В., Румянцев А.С., Гаримелла Р.М. Вероятностная модель для оценки основных характеристик производительности марковской модели суперкомпьютера // Информатика и ее применения, 2023, Т.17, № 2, с. 62-70. doi: 10.14357/19922264230209

[5] Вишневский В.М., Ефросинин Д.В. Теория очередей и машинное обучение. М.: ИНФРА-М, 2025, 370 с.

Размер отладочных заданий ограничивается сравнительно небольшим числом ядер $P_{отл}$ и коротким временем выполнения $T_{отл}$. Задания, чьи требования не превышают этих значений, категорируются как отладочные. Ординарными считаются однократно запускаемые задания, время исполнения которых сравнительно велико и ограничивается сверху параметром $T_{орд}$. Число ядер для ординарного задания ограничивается только общим числом ядер в суперкомпьютерной системе, однако все ординарные задания в сумме не могут на время, превышающее $T_{отл}$, занимать ядер больше, чем разность между общим числом ядер и значением $P_{отл}$. Фактически ядра числом $P_{отл}$ «резервируются» для отладочных заданий, но это «резервирование» не означает выделение конкретных ядер или узлов. Планировщик гарантирует, что $P_{отл}$ процессоров не будет использовано для ординарных заданий на время, не превышающее $T_{отл}$, а какие конкретно узлы будут «зарезервированы» для этой цели, зависит от текущей ситуации. Подход с «плавающим» резервом отладочных ядер позволяет сократить простои ресурсов при отсутствии в системе отладочных заданий.

Фоновые задания могут выполняться произвольное время, но при этом периодически прерываться системой управления. Для фонового задания пользователь явно указывает квант – минимальное время выполнения. СУППЗ гарантирует, что если фоновое задание было запущено, то ему будет предоставлено время, не меньшее указанного кванта. Если по истечении кванта будут обнаружены задания, претендующие на занятые фоновым заданием ресурсы, это задание будет снято с выполнения и заново поставлено в очередь.

Приоритет пользователя напрямую зависит от суммарного времени T_{user} выполнения всех его заданий за некоторый учетный период. В зависимости от величины T_{user} заданию пользователя присваивается один из k приоритетов, $0 \le k \le 6$. Планировщик пытается выделить ресурсы из числа свободных сначала для заданий приоритета 0, потом – приоритета 1 и т.д. Внутри одного приоритета ресурсы выделяются в порядке поступления заданий (принцип FCFS). Никакое менее приоритетное задание не может занять ресурсы так, чтобы это отодвинуло запуск более приоритетного задания. При отсутствии конфликта по ресурсам менее приоритетное задание может стартовать раньше более приоритетного в соответствии с консервативным алгоритмом обратного заполнения.

Метод был реализован в составе планировщика СУППЗ и применяется на суперкомпьютерах, установленных в МСЦ РАН и ИПМ им. М.В. Келдыша РАН. Анализ статистики позволяет утверждать, что определяемый в соответствии с (3) средний коэффициент замедления отладочных и фоновых заданий в подавляющем большинстве случаев кратно ниже, чем общий коэффициент замедления, а темп обработки заданий этих классов соответственно выше. Примером может служить приведенная на рисунке 3 статистика суперкомпьютера МВС-100К (МСЦ РАН), который с 2009 по 2013 гг. входил в рейтинг Топ-500 самых производительных суперкомпьютеров в мире (38-е место в 2009 г.).

Механизм фоновых заданий положительно сказывается на загрузке, что можно подтвердить результатами сравнения из таблицы 1. Поскольку SLURM не поддерживает отладочные задания, при формировании входного потока для симулятора SLURM каждый запуск каждого отладочного задания был представлен в виде отдельного задания. Как итог, SLURM обеспечил загрузку на 3% меньше, чем СУППЗ.

На примере раздела Broadwell суперкомпьютера МВС-10П ОП (МСЦ РАН) показаны преимущества механизма отладочных заданий, который в этой системе был

задействован с 5 августа 2021 года по 30 мая 2023 года и за этот период продемонстрировал кратное снижение коэффициента замедления отладочных заданий.

«Плавающий резерв» ВУ при отсутствии отладочных заданий автоматически задействовался заданиями других классов. Это позволило увеличить загрузку на разделе Broadwell до 4,5 тыс. узло-часов в год, а на суперкомпьютере МВС-100К – от 9 до 14 тыс. узло-часов в год.

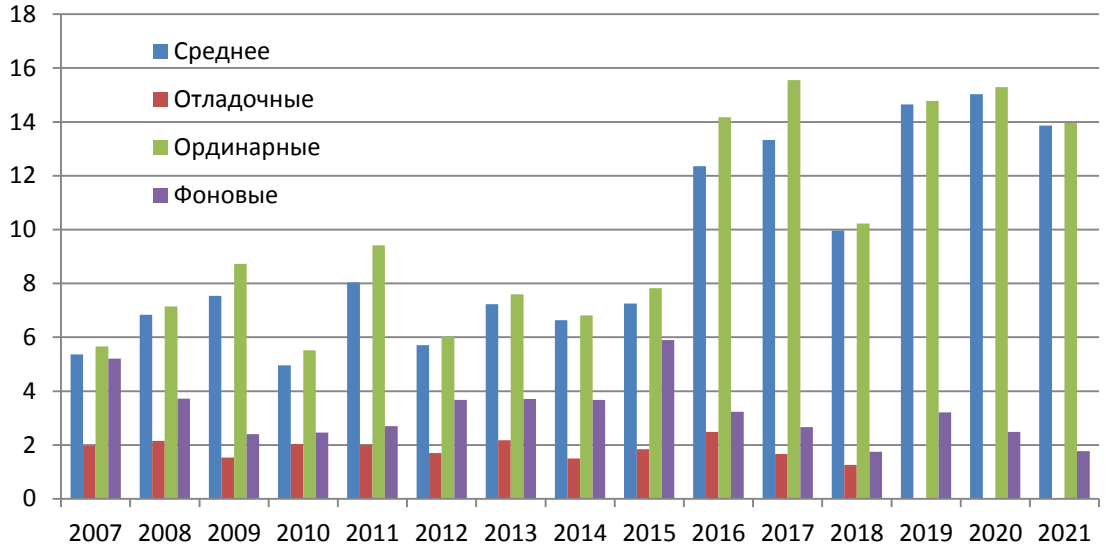

Рис. 3. Коэффициент замедления разных типов заданий
для суперкомпьютера МВС-100К

Доля своевременно обработанных заданий $C_{своевр}$, рассчитанная в соответствии с (5), составила от 0,944 до 0,989 на разных суперкомпьютерах. При этом в большинстве исследуемых систем значение $C_{своевр}$ для отладочных заданий оказалось выше по сравнению с ординарными и фоновыми заданиями.

П. 3.4 посвящен планированию адаптивных (эластичных) заданий, которые могут быть выполнены на параллельном ресурсе произвольного размера в некотором заданном диапазоне. Как правило, для эластичных заданий неизменным остается площадь требуемого параллельного ресурса. Условно, такое задание может быть выполнено на 100 ВУ за 10 часов или на 10 ВУ за 100 часов.

Пусть задания с фиксированными параметрами образуют основной поток, а адаптивные задания – дополнительный поток, который обрабатывается специальным компонентом системы управления – постпланировщиком. Автором предложен метод постпланирования [4, 11], суть которого в том, что постпланировщик при обнаружении окна в расписании запусков заданий основного потока помещает очередное адаптивное задание в основную очередь. Время выполнения и число узлов адаптивного задания подбираются в соответствии с размером обнаруженного окна, что за счет принципа обратного заполнения влечёт практически немедленный запуск адаптивного задания основным планировщиком.

Для исследования характеристик метода постпланирования и определения границ его применимости было проведено имитационное моделирование [4, 13] на 17 входных модельных потоках с различной интенсивностью, обеспечивавших загрузку суперкомпьютера от 59,6% до 98,4%. Постпланирование повышает загрузку суперкомпьютера для входных потоков любой интенсивности, как показано на рисунке 4. Платой за это является незначительное ухудшение таких показателей, как среднее

время ожидания и средний коэффициент замедления заданий основной очереди. Отрицательное влияние постпланирования сокращается при уменьшении интенсивности входного потока, что продемонстрировано на рисунке 5.

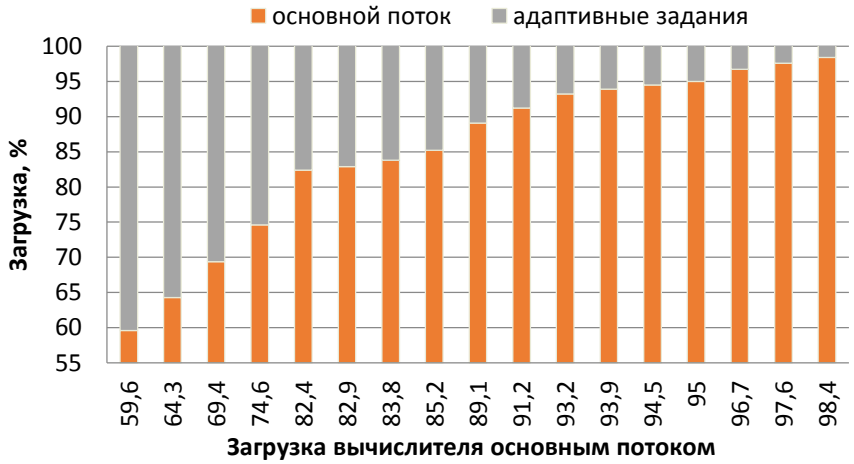

Рис. 4. Загрузка для входных модельных потоков различной интенсивности

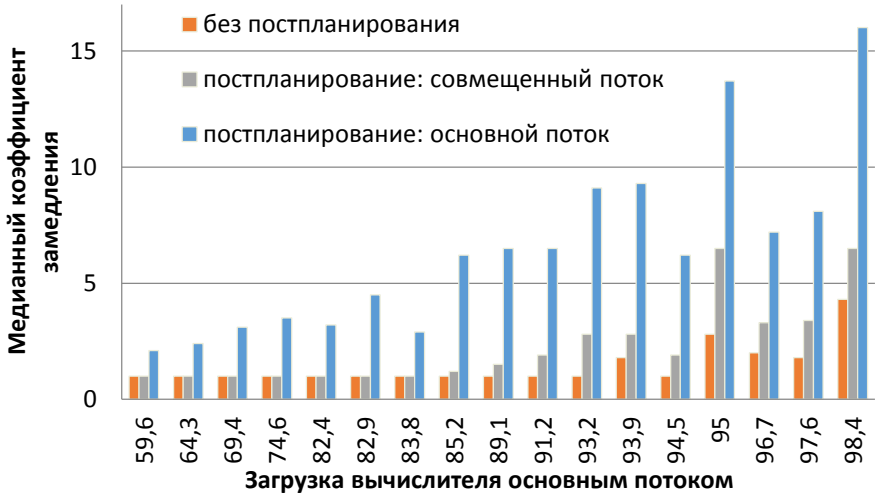

Рис. 5. Медианный коэффициент замедления для входных модельных потоков различной интенсивности

На диаграмме рисунка 5 видно, что коэффициенты замедления заданий совмещенного в результате постпланирования потока и заданий основного потока без постпланирования практически не отличаются, если загрузка вычислителя основным потоком не превосходит 85%. Это позволяет говорить о границе применимости метода постпланирования: если загрузка вычислителя менее 85%, за счет потока адаптивных заданий метод позволит максимизировать загрузку без существенного замедления заданий в очереди. Результаты имитационного моделирования, показывающие влияние метода на показатели качества планирования заданий при входных потоках различной интенсивности и определяющие границы применимости метода, получены впервые.

В п. 3.5 рассмотрены методы совмещения потоков стандартных и нестандартных заданий. Под стандартными заданиями понимаются задания с фиксированными параметрами, использующие для своего выполнения типовой стек программного обеспечения. Нестандартные задания могут поступать в СУЗ суперкомпьютера из об-

лачных платформ, либо представлять собой набор виртуальных машин для некоторого облачного сервиса высокопроизводительных вычислений. Автором предложено два метода совмещения потоков стандартных и нестандартных заданий.

Первый метод [42] заключается в представлении системы управления заданиями в виде гипервизора. Основным средством реализации метода явилось создание собственного драйвера для библиотеки управления виртуальными машинами libvirt, на которую опирается значительное число существующих облачных платформ. Создание драйвера позволило осуществить прозрачное включение СУППЗ в состав облачной платформы OpenStack. Предложенный метод был реализован в виде облачного сервиса для высокопроизводительных вычислений, развёрнутого на ресурсах суперкомпьютера МВС-100К.

Второй метод [8, 40] основан на использовании облачной платформы, как инструмента автоматического развертывания на вычислительных узлах суперкомпьютера виртуальных машин и контейнеров при старте нестандартного задания. При этом вычислительные узлы для запуска набора виртуальных машин динамически выделяются через СУППЗ. Проведённые эксперименты показали, что применённые при реализации метода средства виртуализации KVM и Proxmox VE не оказывают значительного влияния на производительность стандартных заданий, что позволяет успешно совмещать обработку стандартных и нестандартных заданий в одном входном потоке. Результаты исследований позволили предложить трехуровневую архитектуру облачной среды для научных исследований, предоставляющей пользователям Iaas-, PaaS- и SaaS-сервисы для высокопроизводительных вычислений.

Четвертая глава посвящена методам и алгоритмам отображения параллельной программы на вычислительные узлы суперкомпьютера. В п. 4.1 производится формальная постановка задачи как задачи управления вычислительными ресурсами третьего уровня иерархии, преследующей цель распределить по узлам процессы расчетной программы из состава задания таким образом, чтобы минимизировать время выполнения этой программы.

В каждый момент времени существует некоторый подграф $G_N = (V_N, E_N)$, где V_N – множество вершин, представляющих свободные ВУ, E_N – множество дуг, представляющих линии связи между свободными ВУ. Обозначим число свободных ВУ как $N = | V_N |$. Дугам графа G_N приписываются веса m_{ij}, отражающие пропускную способность (либо латентность) линии связи между i-м и j-м узлами. Отметим, что состав и структура подграфа G_N постоянно изменяются в зависимости от текущей мультипрограммной ситуации. Пусть в некоторый момент из очереди поступает задание, содержащее параллельную программу из M взаимодействующих процессов. Программа может быть представлена графом $G_M = (A_M, E_M)$, где A_M – множество вершин, соответствующее процессам программы, E_M – множество дуг, представляющих информационные связи между этими процессами. Дугам графа G_M приписываются веса c_{ij}, отражающие интенсивность информационного обмена между i-м и j-м процессами. Граф G_M называется программным или информационным графом.

Задача отображения сводится в такой постановке к поиску отображения информационного графа G_M параллельной программы на структуру свободных ВУ, заданную графом G_N. Это отображение обозначается $\varphi : A_M \rightarrow V_N$ и представляется матрицей $X = \{ X_{ij} : i \in A_M, j \in V_N \}$, где $X_{ij} = 1$, если $\varphi(i) = j$, и $X_{ij} = 0$, если $\varphi(i) \neq j$.

Критерием оптимальности отображения для случая однородных ВУ служит целевая функция $F(X)$[6]:

$$F(X) = \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{p=1}^{M} \sum_{k=1}^{M} m_{ij} c_{kp} X_{ki} X_{pj} \quad \longrightarrow \quad \min. \qquad (6)$$

Если в качестве весов m_{ij} используются величины, обратные пропускной способности линии связи между i-м и j-м ВУ, то целевая функция (6) будет определять время информационных обменов между процессами параллельной программы.

Нахождение минимума (6) требуется обеспечить в системе управления заданиями суперкомпьютера в условиях обработки потока пользовательских заданий. При запуске очередного задания узлы для него выделяются из числа свободных на момент запуска. Поскольку состав графа G_N динамичен, какие конкретно узлы будут свободны при поступлении из очереди задания, заранее неизвестно. Для каждого задания требуется найти оптимальное в соответствии с (6) отображение программного графа на неизвестный заранее граф подмножества свободных узлов. Отображение должно строиться за приемлемое время, то есть быть существенно меньше времени выполнения задания (в среднем, несколько часов) и не превышать при этом времени системных таймаутов (до 5-15 минут).

В п. 4.2 приведен анализ научных публикаций по теме решения задачи отображения как частного случая NP-полной квадратичной задачи о назначениях. Большинство авторов отмечают экспоненциальную вычислительную сложность точных алгоритмов, которая делает последние неприменимыми для оперативного поиска отображения. Среди эвристических алгоритмов наиболее широкое распространение получили алгоритмы имитации отжига и генетического отбора. В большинстве публикаций отмечается, что генетический алгоритм превосходит остальные эвристики по точности отображения, в то время как алгоритм имитации отжига является одним из самых быстрых. Результаты многих исследований подтверждают целесообразность применения параллельных версий алгоритмов поиска отображения, при этом исследователи для повышения точности и скорости поиска отображения сосредотачивают свои усилия на комбинировании различных базовых эвристик.

В п. 4.3 рассмотрены предложенные автором метод и алгоритмы отображения параллельной программы на вычислительные узлы суперкомпьютера [29].

Сформулируем две подзадачи поиска отображения:

- выбор и выделение требуемых для выполнения параллельной программы ВУ из числа свободных;
- назначение процессов (ветвей) параллельной программы на ВУ.

При одновременном решении этих подзадач фактически решается сразу подзадача 2, т.е. выделенными для параллельной программы считаются все свободные ВУ системы. В результате отображения происходит назначение ветвей программы на определённые ВУ. Эти ВУ становятся занятыми, а свободными

[6] Монахов О.Г., Гросбейн Е.Б., Мусин А.Р. Алгоритмы отображения для параллельных систем на основе моделирования эволюции и моделирования отжига // Труды 6-го международного семинара «Распределенная обработка информации». Под ред. В.Г. Хорошевского. Новосибирск: СО РАН, 1998, с.142.

остаются те узлы, на которые не была назначена ни одна ветвь. Такой подход обладает рядом недостатков.

Во-первых, размерность задачи поиска отображения будет определяться числом свободных ВУ, которых почти всегда больше, чем число требуемых для задания ВУ. Учитывая экспоненциальную сложность поиска отображения, увеличение размера задачи крайне нежелательно. В некоторой мере компенсировать возросший размер задачи возможно за счет применения параллельных эвристических алгоритмов поиска отображения, для выполнения которых можно использовать свободные ВУ. Однако, во время работы параллельного алгоритма поиска отображения будет невозможен запуск других заданий, поскольку в это время все свободные ВУ будут заняты выполнением алгоритма.

Во-вторых, нахождение оптимального решения (6) для одной отдельно взятой программы не означает, что для программ из состава следующих заданий удастся произвести хорошее отображение на оставшиеся ВУ. При поиске отображения необходимо обеспечить сокращение как времени выполнения отдельно взятого задания, так и суммарного времени выполнения всего потока задания.

Альтернативой является последовательное решение подзадач 1-2, при котором сначала производится выбор требуемого параллельной программе количества ВУ из числа свободных, а затем решается задача назначения ветвей программы на ВУ из выделенного подмножества. При этом для решения подзадачи 2 становится возможным применить параллельные алгоритмы, задействовав для их выполнения выделенные ВУ. В этом, собственно, и заключается суть предлагаемого метода.

Выбор M вычислительных узлов для очередного задания сведем к разбиению множества N свободных ВУ на два подмножества из M и из N-M ВУ с минимальным количеством связей между этими подмножествами. Это эквивалентно разрезанию графа $G_N = (V_N, V_N)$ свободных ВУ на два подграфа, $G_1 = (V_1, E_1)$ и $G_2 = (V_2, E_2)$, $|V_1| = M$, $|V_2| = N - M$, причем

$$V_1 \bigcup V_2 = V_N, V_1 \bigcap V_2 = \varnothing,$$

таким образом, чтобы сумма весов ребер, попавших в разрез графа G_N, была минимальной. Сумма весов ребер, связывающих вершины из различных подграфов, называется связностью разрезания q. В общем случае задача разрезания графа на два подграфа с минимальной связностью разрезания также является NP-полной.

Среди эвристических алгоритмов наиболее быстрым, простым и удобным для реализации является алгоритм[7], в начале которого множества V_1 и V_2 выбираются произвольным образом, и вычисляется связность начального разрезания q_0. На каждом шаге алгоритма для всех вершин, составляющих подграфы G_1 и G_2, вычисляются значения внутренней и внешней связности. В каждом подграфе выделяется вершина с наибольшей разностью между внутренней и внешней связностью. Далее осуществляется обмен выделенными вершинами между подграфами, и вычисляется связность q_k полученного нового разрезания, где k – номер шага. Переход к следующему шагу осуществляется, если значение q_k не больше значения q_{k-1}, полученного на предыдущем шаге алгоритма, или в подграфах

[7] Фещенко В.П., Матюшков Л.П. Итерационный алгоритм разрезания графа на К подграфов. В сб.: Автоматизация проектирования сложных систем (Вычислительная техника в машиностроении). Минск. 1976. Вып. 2. С. 74-77

G_1 и G_2 еще остаются нерассмотренные вершины. В противном случае алгоритм завершает работу.

Рассмотренный алгоритм достаточно быстр и прост в реализации, однако не обеспечивает поиска оптимального решения даже при небольших размерах графов из-за сильной зависимости от начального разрезания. Для улучшения характеристик алгоритма автором было предложено применить метод имитации отжига. Модифицированный алгоритм получил название РГО (Разрезание Графа Отжигом) и так же, как и исходный алгоритм, заключается в последовательности взаимных перестановок случайных вершин из V_1 и V_2. Если связность разрезания уменьшается, то перестановка фиксируется, а если нет – то фиксируется с вероятностью $e^{-\Delta q/T}$, где Δq – приращение связности, а T – текущая температура отжига. Отметим, что в методе имитации отжига температура носит характер условной безразмерной переменной величины, уменьшающейся в ходе работы алгоритма. Для преодоления эффекта «застревания в локальном минимуме» на последних стадиях при достижении температуры 1° осуществляется переход к исходному алгоритму, который за несколько итераций находит оптимальное решение.

Для решения подзадачи распределения процессов параллельной программы по выделенным ВУ автором был предложен параллельный алгоритм имитации отжига с генетическими операциями (ПОГ – Параллельный Отжиг и Генетика). Алгоритм ПОГ выполняется по схеме «мастер-рабочие». Начальная популяция заполняется случайно выбранными особями, которые распределяются между процессами-рабочими, каждый из которых осуществляет над своей частью популяции алгоритм имитации отжига. Результаты отжига собираются в процессе-мастере, который осуществляет над популяцией операции селекции и отбраковывает худшие особи (решения процессов-рабочих). Далее мастер путем скрещивания формирует новые особи, которые вместе с отобранными в результате селекции особями образуют новую популяцию. Новая популяция распределяется по процессам-рабочим, и начинается очередной шаг алгоритма. Алгоритм останавливается, если по истечении заданного числа шагов не произошло улучшения целевой функции (6).

Алгоритмы РГО и ПОГ были реализованы в составе СУППЗ для суперкомпьютеров серии МВС-1000. Апробация алгоритмов [29] была произведена при помощи известного стандартного набора тестов NPB (NAS Parallel Benchmark) версии 2.3, которые выполнялись на 32-процессорной системе МВС-1000, установленной в 1999 году в ИПМ им. М.В. Келдыша РАН. Каждый тест NPB помимо времени своей работы выдает оценку производительности P суперкомпьютера в миллионах операций в секунду. Каждый тест NPB выполнялся без применения алгоритмов РГО и ПОГ и с применением этих алгоритмов. Если обозначить производительность, оцененную тестами без применения алгоритмов отображения, как P_1, а производительность, оцененную с применением алгоритмов, как P_2, то можно определить прирост производительности W

$$W = \frac{P_2 - P_1}{P_1} \times 100\%. \qquad (7)$$

Усредненный по всем тестам прирост производительности от применения алгоритма РГО в суперкомпьютере МВС-1000 представлен в таблице 2, а прирост производительности от применения алгоритма ПОГ – в таблице 3.

Апробация алгоритма ПОГ производилась также и на реальных задачах. При решении задачи моделирования процессов структурообразования рибонуклеиновых кислот программным комплексом ГЕН-2[8] прирост производительности составил от 37,1% до 47,5% в зависимости от числа процессоров.

Табл. 2. Средний по всем тестам NPB прирост производительности при применении алгоритма РГО в суперкомпьютере МВС-1000

Число процессоров (ВУ)	4	8 (9)	16
Прирост производительности *W*	10,1%	22,3%	23,3%

Табл. 3. Средний по всем тестам NPB прирост производительности при применении алгоритма ПОГ в суперкомпьютере МВС-1000

Число процессоров (ВУ)	4	8 (9)	16	25	32
Прирост производительности *W*	0%	23,4%	32,4%	44,3%	36%

В п. 4.4 рассмотрены предложенные автором решения по развитию алгоритма ПОГ в виде параллельного комбинированного алгоритма PGSA (Parallel Genetic Simulated Annealing) [6] и его циклической модификации CPGSA (Cyclic PGSA) [3]. Алгоритм PGSA состоит из двух фаз: параллельных алгоритмов имитации отжига и генетического отбора, в алгоритме CPGSA эти фазы циклически повторяются. Указанные алгоритмы были реализованы в составе программного средства GraphHunter [5], с помощью которого было произведено экспериментальное исследование их характеристик.

Эксперименты [6] показали, что алгоритм PGSA всегда даёт среднее лучшее решение, чем генетический алгоритм и имитация отжига, однако его быстродействие не превышает быстродействия генетического алгоритма. Циклическое повторение фаз отжига и генетического отбора в алгоритме CPGSA [3] позволило добиться итерационного повышения точности найденного отображения и скорости его поиска. В результате удалось достичь более высокой точности отображения графа программы на граф ВУ по сравнению с известными алгоритмами при времени поиска отображения, сравнимом с системными таймаутами. Точность разных алгоритмов в виде значения целевой функции (6) представлена на рисунке 6, а время их работы в секундах показано на рисунке 7. При сравнении использовался известный набор тестовых графов для решения квадратичной задачи о назначениях[9] с заранее известным оптимальным значением целевой функции (6).

[8] Козлов Н.Н., Кугушев Е.И., Энеев Т.М. Параллельные вычисления при решении некоторых задач астрофизики и молекулярной биологии // Матем. моделирование, 2000, №12:7, с. 65–70.

[9] Drezner Z., Hahn P.M., Taillard E.D. Recent advances for the quadratic assignment problem with special emphasis on instances that are difficult for meta-heuristic methods // Ann. Oper. Res., 2005, vol. 139, pp. 65–94. doi: 10.1007/s10479-005-3444-z

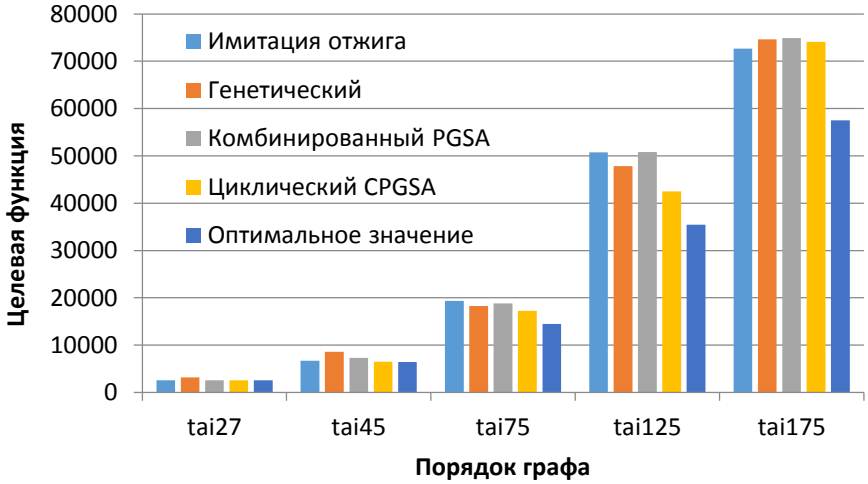

Рис. 6. Зависимость значения целевой функции (6) от порядка графа для различных алгоритмов отображения

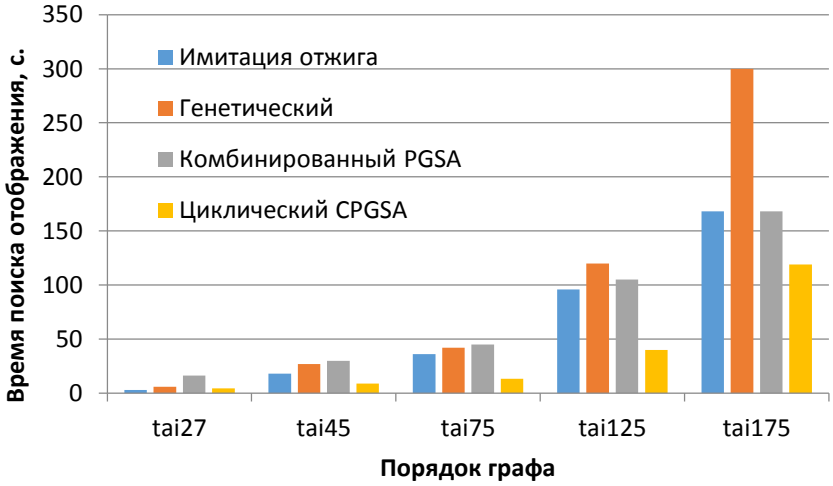

Рис. 7. Зависимость времени поиска отображения различными алгоритмами от порядка графа

Пятая глава посвящена вопросам организации параллельных вычислений с распараллеливанием по данным, в которых одна и та же последовательность вычислений (прикладной алгоритм) выполняется над каждым элементом множества (пула) входных данных. Между процессами параллельной программы отсутствуют информационные обмены. К типичным задачам распараллеливания по данным относятся виртуальный скрининг, поиск простых чисел, проверка стойкости паролей и др.

В п. 5.1 вводится модель одной последовательной программы (ОПП), реализующей прикладной алгоритм. Порция входных данных для ОПП определяется значениями одного или нескольких параметров. Из таких порций вычислительной работы складывается пул входных данных – множество всех возможных значений параметров ОПП во всех их комбинациях.

При распараллеливании по данным на каждом узле суперкомпьютера выполняются один или несколько экземпляров ОПП с различными значениями входных параметров. Организация параллельных вычислений заключается в осуществлении запусков экземпляров ОПП на всем множестве доступных узлов. Определим техноло-

гию распараллеливания по данным как совокупность методов, алгоритмов и средств, осуществляющих распределение экземпляров ОПП и порций данных по вычислительным узлам суперкомпьютера.

Пусть ОПП обрабатывает весь пул входных данных в течение некоторого времени. Если входные данные равномерно распределить по p вычислительным узлам и запустить на этих узлах экземпляры ОПП, то при отсутствии накладных расходов обработка ускорится в p раз. Но в действительности имеют место накладные расходы на организацию распараллеливания: затраты времени на передачу служебных данных между ВУ, на обращения к сервисам (база данных, веб сервер, планировщик), задержки между получением данных и началом их обработки и другие.

Помимо накладных расходов, определяющих быстродействие вычислений, важным показателем качества является отказоустойчивость, подразумевающая способность системы распараллеливания по данным как продолжать вычисления в случае выхода из строя части ВУ, так и возобновлять вычисления с автоматически сохраняемой контрольной точки в случае отказа всех ВУ или управляющей ЭВМ.

В п. 5.2 приведен обзор известных технологий распараллеливания по данным: MapReduce, BOINC и программного комплекса X-COM. Рассмотренные технологии, хотя и могут быть применены на отдельном суперкомпьютере, но создавались для распределенной обработки данных. Пользователь, применяющий эти технологии, освобождается от множества рутинных задач организации параллельных вычислений, таких как распределение вычислений и данных по узлам суперкомпьютера, учет обработанных порций данных, обеспечение отказоустойчивости вычислений, сбор результатов. Тем не менее, все рассмотренные технологии предполагают разработку специальных служебных программ, от эффективности которых во многом зависит итоговое быстродействие расчетов.

В п. 5.3 рассмотрен предложенный автором диссертации метод иерархического деления данных [38].

Представим пул входных данных как множество D всевозможных комбинаций значений, задаваемых параметрами ОПП. Назовем элемент множества D слайсом, задающим элементарную работу (элементарную порцию данных), выполняемую экземпляром ОПП. Сопоставим каждому слайсу $S \in D$ номер – натуральное число $i, 1 \le i \le N$, где $N = |D|$, т.е. пронумеруем слайсы множества D от *1* до N. Определим подмножество

$$D_i^j \subseteq D,\ i \in \overline{1,N},\ j \in \overline{1,N},\ i < j,\ \left|D_i^j\right| = j - i + 1,$$

как множество слайсов с последовательными номерами от i до j.

Пул работ D_m^n полностью определяется номерами m, n первого и последнего слайсов соответственно. Определим функцию $Chip(D_m^n, k)$ деления множества D_m^n на два подмножества D_m^{m+k-1} и D_{m+k}^n. Функцию деления можно рекурсивно применить к подмножеству D_m^{m+k-1} для его дальнейшего деления на подмножества меньшей мощности.

Введем иерархическую систему объектов, обрабатывающих множество D, с числом уровней иерархии r. Для каждого уровня иерархии (рекурсии) $l \in \overline{1,r}$ может быть определена мощность n_l пула работ. Объект, обрабатывающий пул работ на уровне l, назовем менеджером M_l^i, где $i \in \overline{1,c_l}$ – это номер менеджера, а c_l – число менеджеров на уровне l. В подчинении менеджера M_l^i находятся менеджеры следующего уровня иерархии M_{l+1}^j, $j \in \overline{1,c_{l+1}}$.

Пусть в качестве пула входных данных для некоторого менеджера M_l^m выступает подмножество D_1^n, где $n = |D_1^n| = n_l$. Менеджер M_l^m отделяет от своего пула работ D_1^n очередную порцию D_i

$$D_i = D_{1+i\cdot n_{l+1}}^{(i+1)n_{l+1}},\ i \in \overline{1, k_{l+1}},\ k_{l+1} = \left\lceil \frac{n_l}{n_{l+1}} \right\rceil,$$

которая передается для выполнения первому свободному менеджеру нижестоящего уровня M_{l+1}^j, не занятому в этот момент вычислительной работой. Если все менеджеры нижестоящего уровня заняты работой, менеджер M_l^m переходит в состояние ожидания результатов.

Для каждого менеджера нижестоящего уровня M_{l+1}^j, выполнившего переданную ему порцию работ, вышестоящий менеджер M_l^m сохраняет результат обработки порции, отделяет от остатка пула входных данных очередную порцию D_i и передает её для выполнения освободившемуся менеджеру M_{l+1}^j. Если очередную порцию работы отделить нельзя по причине исчерпанного пула D_1^n, то запускается фаза перераспределения работы. Смысл этой фазы состоит в том, чтобы каждая еще не обработанная к текущему моменту порция работы D_i была бы вновь назначена (перераспределена) освободившимся, т.е. уже выполнившим свои порции менеджерам. Перераспределение повторяется до тех пор, пока все выданные и перераспределенные порции не будут обработаны. После этого входной пул D_1^n считается выполненным, и менеджер M_l^m возвращает результат работы менеджеру вышестоящего уровня *l-1*.

Отметим важные преимущества предложенного метода иерархического разделения данных.

Во-первых, контрольная точка на каждом уровне иерархии l в каждый момент времени будет определяться невыполненной на этот момент времени работой, которую составляют:

– текущий остаток невыполненной работы $D_l = D_{i\times n_{l+1}+1}^n$, который полностью определяется номерами граничных слайсов $i \times n_{l+1} + 1$ и n;

– множество из c_{l+1} порций работы, розданных менеджерам нижестоящего уровня M_{l+1}^j, причем каждая порция работы также полностью будет определяться номерами первого и последнего слайсов.

Таким образом, для записи контрольной точки достаточно зафиксировать границы текущего остатка и границы каждой порции розданной работы. Вся остальная вычислительная работа из пула D_1^n в этот момент времени уже заведомо выполнена. Для восстановления менеджера M_l^m с контрольной точки достаточно восстановить текущий остаток D_l и розданные менеджерам M_{l+1}^j порции работы, которые будет необходимо повторно передать этим менеджерам для выполнения.

Рассмотренный подход делает контрольную точку компактной, а ее фиксацию быстрой. При этом отпадает необходимость ведения базы данных учета обработанных и необработанных порций данных. На каждом уровне иерархии l в каждый момент времени мы имеем зафиксированную компактную контрольную точку, полностью определяющую оставшуюся для выполнения вычислительную работу.

Во-вторых, автоматически обеспечивается отказоустойчивость вычислений. При отсутствии ответа от любого менеджера нижестоящего уровня, его текущая порция вычислительной работы будет гарантированно выполнена на фазе

перераспределения. Поэтому если какой-либо контролируемый этим менеджером вычислительный узел или вычислитель выходит из строя, вычисления будут выполнены другими вычислителями или узлами с некоторым снижением общей производительности.

В-третьих, аналогично отказоустойчивости обеспечивается масштабируемость вычислений. При появлении в решающем поле новых узлов или вычислителей достаточно добавить в систему менеджеров соответствующего уровня иерархии. Асинхронный механизм распределения работы в предложенном методе разделения данных обеспечит динамическое подключение новых подчиненных менеджеров, которым автоматически начнут выделяться порции вычислительной работы.

В-четвертых, обеспечивается возможность организации вычислений на гетерогенных вычислителях, вычислительные узлы которых имеют архитектурные различия (например, построены на программно-несовместимых микропроцессорах) или (и) состоят из вычислителей различного типа (например, вычислительные узлы содержат универсальные и графические процессоры). Разница в производительности ВУ будет автоматически компенсироваться асинхронным механизмом распределения работы, поскольку очередная порция будет передаваться соответствующему менеджеру только после его освобождения. Чем быстрее будет производить расчеты соответствующий менеджеру вычислитель, тем больше порций работы получит этот менеджер.

В п. 5.4 рассматриваются архитектура и структура программного комплекса (ПК) «Пирамида» [35], в основе которого лежит метод иерархического разделения данных. ПК «Пирамида» выполняет распараллеливание по данным по модели ОПП на ВУ суперкомпьютера, организованных в виде иерархической подсистемы. В отличие от рассмотренных существующих технологий распараллеливания по данным, при применении ПК «Пирамида» пользователь освобождается от разработки служебных программ, вместо которой достаточно настроить конфигурацию ПК.

П. 5.5 посвящен экспериментальному сравнению технологий распараллеливания по данным, реализованных в разных ПК [12, 39]. Для проведения такого сравнения была предложена методика, базирующаяся на понятии элементарной вычислительной работы, под которой понимается обработка неделимой (элементарной) порции входных данных. Пусть элементарная вычислительная работа выполняется за время τ, а весь объём входных данных составляют N элементарных порций. На одном процессорном ядре одним экземпляром ОПП весь объём входных данных будет обработан за время $T_1 = N \cdot \tau$. При использовании p процессорных ядер идеальное время обработки T_p составит

$$T_p = \frac{T_1}{p} = \frac{N \cdot \tau}{p}.$$

Пусть исследуемый ПК произвёл обработку входных данных на p процессорных ядрах за время $T_{экс}(p)$. Тогда доля накладных расходов μ, привносимых ПК, составит

$$\mu = 1 - \frac{T_p}{T_{экс}(p)} = 1 - \frac{N \cdot \tau}{p \cdot T_{экс}(p)}. \qquad (8)$$

Доля накладных расходов зависит от параметров N, τ и p. Варьируя значения одного из параметров при фиксированных значениях двух других, можно оценить динамику изменения накладных расходов на организацию распараллеливания по данным для каждого из исследуемых ПК.

Экспериментальное сравнение ПК «Пирамида» с программными средствами, реализованными на основе технологий MapReduce, MPI, BOINC, а также с ПК X-COM, показало, что ПК «Пирамида» вносит существенно меньшие накладные расходы при распараллеливании, основанном на переборе комбинаций параметров. Например, на рисунке 7 (группа столбцов «3 параметра») показаны доли накладных расходов исследуемых ПК для тестовой ОПП перебора трех параметров. В ходе экспериментов было обнаружено, что ПК «Пирамида» уступает ПК X-COM при обработке пула входных данных, представленного в виде набора строк в файле (группа столбцов «Файл»).

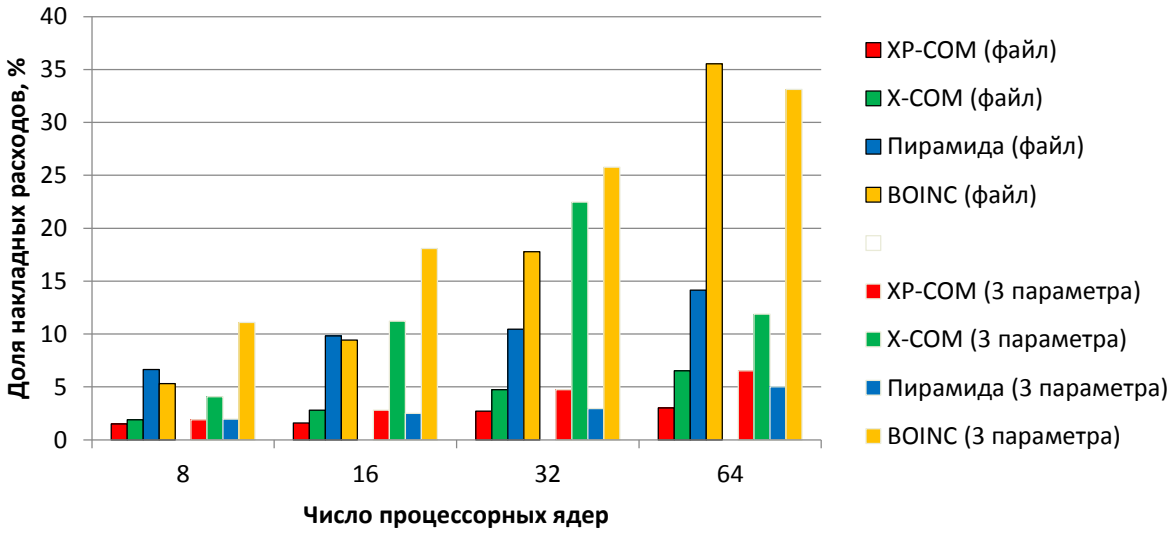

Рис. 8. Доля накладных расходов для перебора трех параметров и строк из файла

В п. 5.6 рассмотрены предложенные автором архитектурные и технические решения по объединению программных комплексов «Пирамида» и X-COM [7]. Основу гибридного ПК, получившего название XP-COM, составили транспортная инфраструктура ПК X-COM и программные модули ПК «Пирамида», реализующие метод иерархического деления данных. Результаты экспериментов, представленные на рисунке 8 (группа столбцов «Файл»), продемонстрировали, что гибридный комплекс XP-COM привносит меньшие накладные расходы по сравнению как с ПК X-COM, так и с ПК «Пирамида».

В заключении основные результаты и выводы диссертации представлены как комплекс новых научно обоснованных архитектурных, технических и технологических решений, внедрение которых вносит значительный вклад в развитие научных суперкомпьютерных центров коллективного пользования, как неотъемлемой части исследовательской инфраструктуры страны. На основе предложенных решений была создана Система управления прохождением параллельных заданий (СУППЗ), обеспечивающая комплекс возможностей, качественных характеристик и количественных показателей, соответствующий мировому уровню. Эффективность предложенных решений подтверждается их успешным применением в составе СУППЗ в российских суперкомпьютерах, установленных в 1999-2024 гг. в МСЦ РАН – НИЦ «Курчатовский институт», ИПМ им. М.В. Келдыша РАН и ряде других научных и образовательных организаций, в том числе статистическими

данными о работе этих суперкомпьютеров, а также результатами экспериментов и имитационного моделирования.

Результаты выполненных автором исследований применяются пользователями суперкомпьютерных центров коллективного пользования НИЦ «Курчатовский институт» и ИПМ им. М.В. Келдыша РАН при проведении научных исследований по темам государственных заданий, грантов РНФ и Минобрнауки России.

Результаты диссертации внедрены в практическую деятельность Федерального научного центра Научно-исследовательский институт системных исследований Национального исследовательского центра «Курчатовский институт», Федерального исследовательского центра Институт прикладной математики им. М.В. Келдыша Российской академии наук, Научно-исследовательского института «Квант», Института математики и механики им. Н.Н. Красовского Уральского отделения Российской академии наук, а также использованы для проведения высокопроизводительных расчетов в практике Института биохимической физики им. Н.М. Эмануэля Российской академии наук и Физико-технического института им. А.Ф. Иоффе Российской академии наук.

Материалы диссертации используются в учебном процессе при подготовке специалистов по специальностям «Вычислительные машины, комплексы, системы и сети» и «Информационная безопасность автоматизированных систем».

Результаты диссертационной работы могут быть использованы для исследований и разработки систем управления заданиями суперкомпьютеров, организации параллельных вычислений с распараллеливанием по данным, создания и развития суперкомпьютерных центров коллективного пользования.

ПУБЛИКАЦИИ ПО ТЕМЕ ДИССЕРТАЦИИ

1. Баранов А.В. Построение системы управления заданиями пользователей суперкомпьютера на основе иерархической модели // Программные продукты и системы, 2025, №2, с. 345-360. doi: 10.15827/0236-235X.150.345-360
2. Baranov A.V. HPC Scheduling Method Based on Debug and Background Job Classes Division // Lobachevskii Journal of Mathematics, 2024, vol. 45, no. 10, pp. 4899-4911. doi: 10.1134/S1995080224606118
3. Baranov A., Aladyshev O., Bragin K. Job Mapping Cyclic Composite Algorithm for Supercomputer Resource Manager// Lecture Notes in Computer Science, vol. 15406, pp. 377–389, 2025. doi: 10.1007/978-3-031-78459-0_27
4. Baranov A.V., Lyakhovets D.S., Konstantinov P.A. A Method for Combining Heterogeneous Workflows in HPC Systems // Lobachevskii Journal of Mathematics, 2024, vol. 45, No. 10, pp. 5111-5125. doi: 10.1134/S1995080224606131
5. Баранов А.В., Киселёв Е.А., Телегин П.Н., Сорокин А.А. Программное средство GraphHunter поиска отображения параллельной программы на структуру суперкомпьютерной системы // Программные продукты и системы, 2022, №4, с. 583-597. doi: 10.15827/0236-235X.140.583-597
6. Baranov A.V., Kiselev E.A., Shabanov B.M., Sorokin A.A., Telegin P.N. Comparison of Three Job Mapping Algorithms for Supercomputer Resource Managers // Lobachevskii Journal of Mathematics, 2022, no 43 (10), pp. 121-133. doi: 10.1134/S199508022213008X

7. Baranov A., Aladyshev O., Kiselev E. et al. XP-COM hybrid software package for parallelization by data // Communications in Computer and Information Science, 2020, vol. 1263, pp. 3-15. doi: 10.1007/978-3-030-55326-5_1

8. Баранов А.В., Киселев Е.А. Облачные сервисы для научных высокопроизводительных вычислений на базе платформы Proxmox // Вычислительные технологии, 2019, Т.24, №6, с. 5-12. doi: 10.25743/ICT.2019.24.6.002

9. Баранов А.В., Николаев Д.С. Использование контейнерной виртуализации в организации высокопроизводительных вычислений // Программные системы: теория и приложения, 2016, Т.7, №1(28), с. 117-134. doi: 10.25209/2079-3316-2016-7-1-117-134

10. Баранов А.В., Николаев Д.С. Применение машинного обучения для прогнозирования времени выполнения суперкомпьютерных заданий // Программные продукты и системы, 2020, №2, 218-228. doi: 10.15827/0236-235X.130.218-228

11. Баранов А.В., Киселёв Е.А., Ляховец Д.С. Квазипланировщик для использования простаивающих вычислительных модулей многопроцессорной вычислительной системы под управлением СУППЗ // Вестник Южно-Уральского государственного университета. Серия: Вычислительная математика и информатика, 2014, №3(4), с. 75-84. doi: 10.14529/cmse140405

12. Baranov A., Kiselev E., Chernyaev D. Experimental comparison of performance and fault tolerance of software packages pyramid, X-COM, and BOINC // Communications in Computer and Information Science, 2016, vol. 687, pp. 279-290. doi: 10.1007/978-3-319-55669-7_22

13. Баранов А.В., Ляховец Д.С. Методы и средства моделирования системы управления суперкомпьютерными заданиями // Программные продукты и системы, 2019, №4, с. 581-594. doi: 10.15827/0236-235X.128.581-594

14. Baranov A., Savin G., Shabanov B. et al. Methods of Jobs Containerization for Supercomputer Workload Managers // Lobachevskii Journal of Mathematics, 2019, vol. 40, no. 5, pp. 525-534. doi: 10.1134/S1995080219050020

15. Savin G.I., Shabanov B.M., Telegin P.N., Baranov A.V. Joint Supercomputer Center of the Russian Academy of Sciences: Present and Future // Lobachevskii Journal of Mathematics, 2019, vol. 40, pp. 1853-1862. DOI: 10.1134/S1995080219110271

16. Shabanov B., Baranov A., Telegin P., Tikhomirov A. Influence of Execution Time Forecast Accuracy on the Efficiency of Scheduling Jobs in a Distributed Network of Supercomputers // Lecture Notes in Computer Science, 2021, vol. 12942, pp. 338-347. doi: 10.1007/978-3-030-86359-3_25

17. Баранов А.В., Тихомиров А.И. Планирование заданий в территориально распределенной системе с абсолютными приоритетами // Вычислительные технологии, 2017, Т. 22, № S1, с. 4-12.

18. Baranov A., Telegin P., Tikhomirov A. Comparison of auction methods for job scheduling with absolute priorities // Lecture Notes in Computer Science, 2017, vol. 10421, pp. 387-395. doi: 10.1007/978-3-319-62932-2_37

19. Шабанов Б.М., Овсянников А.П., Баранов А.В. [и др.] Проект распределенной сети суперкомпьютерных центров коллективного пользования // Программные системы: теория и приложения, 2017, Т. 8, № 4(35), с. 245-262.

20. Баранов А.В., Зонов А.А. Вариант организации облачного сервиса для высокопроизводительных вычислений // Программные системы: теория и приложения, 2016, №7:3(30), с. 3-23. doi: 10.25209/2079-3316-2016-7-3-3-23

21. Баранов А.В., Ляховец Д.С. Влияние пакетирования на эффективность планирования параллельных заданий // Программные системы: теория и приложения, 2017, Т. 8, № 1(32), с. 193-208. doi: 10.25209/2079-3316-2017-8-1-193-208

22. Savin G.I., Shabanov B.M., Fedorov R.S., Baranov A.V., Telegin P.N. Checkpointing Tools in a Supercomputer Center // Lobachevskii Journal of Mathematics, 2020, vol. 41, no. 12, pp. 2603-2613. doi: 10.1134/S1995080220120355

23. Aladyshev O.S., Baranov A.V., Ionin R.P., Kiselev E.A., Shabanov B.M. Variants of deployment the high performance computing in clouds // IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus), Russia, Moscow, 2018, 1453–1457. doi: 10.1109/EIConRus.2018.8317371

24. Савин Г.И., Шабанов Б.М., Баранов А.В. [и др.] Об использовании федеральной научной телекоммуникационной инфраструктуры для суперкомпьютерных вычислений // Вестник Южно-Уральского государственного университета. Серия: Вычислительная математика и информатика, 2020, Т.9, №1, с. 20-35. doi: 10.14529/cmse200102

25. Lyakhovets D.S., Baranov A.V. Group Based Job Scheduling to Increase the High-Performance Computing Efficiency // Lobachevskii Journal of Mathematics, 2020, vol. 41, no. 12, pp. 2558-2565. doi: 10.1134/S1995080220120264

26. Savin G.I., Lyakhovets D.S., Baranov A.V. Influence of Job Runtime Prediction on Scheduling Quality // Lobachevskii Journal of Mathematics, 2021, vol. 42, pp. 2562–2570. doi: 10.1134/S1995080221110196

27. Lyakhovets D.S., Baranov A.V. Efficiency Thresholds of Group Based Job Scheduling in HPC Systems // Lobachevskii Journal of Mathematics, 2022, vol. 43, no. 10, pp. 2863-2876. doi: 10.1134/S1995080222130261

28. Аладышев О.С., Баранов А.В., Ионин Р.П., Киселёв Е. А., Орлов В.А. Сравнительный анализ вариантов развертывания программных платформ для высокопроизводительных вычислений // Вестник УГАТУ, 2014, Т.18, №3(64), с. 295-300.

29. Баранов А.В. Метод и алгоритмы осуществления оптимального отображения параллельной программы на структуру многопроцессорного вычислителя // Материалы конференции «Высокопроизводительные вычисления и их приложения», Черноголовка, 2000, с. 65-67. URL: https://parallel.ru/conferences/chg2000works.html (дата обращения 21.04.2025).

30. Баранов А.В., Лацис А.О., Храмцов М.Ю. Организация многопользовательского режима работы многопроцессорных вычислительных систем // Труды Всероссийской научной конференции «Высокопроизводительные вычисления и их приложения», г. Черноголовка, 2000, с. 67-69. URL: https://parallel.ru/conferences/chg2000works.html (дата обращения 21.04.2025).

31. Корнеев В.В., Киселев А.В., Баранов А.В. [и др.] Опыт практической реализации сетевой среды распределенных вычислений // Научный сервис в сети Интернет: технологии параллельного программирования: Труды Всероссийской научной конференции, Новороссийск, 2006, с. 148-149.

32. Баранов А.В., Смирнов С.В., Храмцов М.Ю., Шарф С.В. Модернизация СУПЗ МВС-1000 // Научный сервис в сети Интернет: решение больших задач: Труды Всероссийской научной конференции, Новороссийск, 2008, с. 226-227.

33. Баранов А.В., Голинка Д.М. Исследование возможности использования планировщика Maui в составе СУПЗ МВС-1000 // Научный сервис в сети Интернет:

решение больших задач: Труды Всероссийской научной конференции, Новороссийск, 2008, с. 223-225.

34. Баранов А.В., Голинка Д.М. Система управления прохождением задач и планировщик Maui для МВС-100К // Научный сервис в сети Интернет: масштабируемость, параллельность, эффективность: Труды Всероссийской суперкомпьютерной конференции, Новороссийск, 2009, с. 314–315.

35. Баранов А.В., Киселёв А.В., Киселёв Е.А., Корнеев В.В, Семёнов Д.В. Программный комплекс «Пирамида» организации параллельных вычислений с распараллеливанием по данным // Труды Международной суперкомпьютерной конференции «Научный сервис в сети интернет: суперкомпьютерные центры и задачи». М: Изд-во МГУ, 2010, с. 299-302.

36. Баранов А.В., Киселев А.В., Старичков В.В. и др. Сравнение систем пакетной обработки с точки зрения организации промышленного счета // Научный сервис в сети Интернет: поиск новых решений: Труды Международной суперкомпьютерной конференции, 2012, с. 506–508.

37. Баранов А.В., Ляховец Д.С. Сравнение качества планирования заданий в системах пакетной обработки SLURM и СУППЗ // Научный сервис в сети Интернет: все грани параллелизма: Труды Международной суперкомпьютерной конференции, Новороссийск, 2013, с. 410–414.

38. Баранов А.В., Киселёв А.В., Киселёв Е.А. [и др.] Применение программного комплекса «Пирамида» для SPMD-вычислений на гетерогенных массово-параллельных ВС // Научный сервис в сети Интернет: все грани параллелизма: Труды Международной суперкомпьютерной конференции, Новороссийск, 2013, с. 308-310.

39. Алексеев А.В., Баранов А.В., Киселёв А.В., Киселёв Е.А. Экспериментальное сравнение технологий распараллеливания по данным Пирамида, MapReduce и MPI // Суперкомпьютерные технологии (СКТ-2014): Материалы 3-й Всероссийской научно-технической конференции. Ростов-на-Дону: Издательство ЮФУ, 2014, Т.1, с. 77-80.

40. Шабанов Б.М., Овсянников А.П., Баранов А.В. [и др.] Методы управления параллельными заданиями суперкомпьютера, требующими развёртывания отдельных программных платформ и виртуализации сетей // Суперкомпьютерные дни в России: Труды международной конференции, Москва, 2017, с. 616-627.

41. Баранов А.В., Киселев Е.А., Кормилицин Е.С. и др. Модернизация подсистемы сбора и обработки статистики центра коллективного пользования вычислительными ресурсами МСЦ РАН / // Труды научно-исследовательского института системных исследований Российской академии наук, 2018, Т. 8, № 4, с. 136-144. doi: 10.25682/NIISI.2018.4.0016

42. Баранов А.В., Аладышев О.С., Овсянников А.П. [и др.] Методы и средства совмещения потоков заданий от облачных платформ и менеджеров управления ресурсами суперкомпьютера // Программные продукты, системы и алгоритмы, 2018, №4, с. 8.

43. Баранов А.В., Долгов Б.В., Федотов А.В. Контейнеризация пользовательских заданий в суперкомпьютерной системе коллективного пользования // Труды научно-исследовательского института системных исследований Российской академии наук, 2019, Т. 9, № 6, с. 123-131. doi: 10.25682/NIISI.2019.6.0016

44. Баранов А.В., Киселев Е.А. Интеграция систем управления заданиями SLURM и СУППЗ // Труды научно-исследовательского института системных исследований Российской академии наук, 2019, т. 9, № 5, с. 29–35.

45. Аладышев О.С., Баранов А.В., Дербышев Д.Ю. Методы и алгоритмы обеспечения прохождения пользовательских заданий с заданным уровнем обслуживания // Труды научно-исследовательского института системных исследований Российской академии наук, 2019, Т. 9, № 5, с. 15-22.