Федеральное государственное бюджетное учреждение
«Национальный исследовательский центр «Курчатовский институт»

На правах рукописи

Ляховец Дмитрий Сергеевич

Методы и средства имитационного моделирования систем управления заданиями для высокопроизводительных вычислений

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

Диссертация на соискание ученой степени
кандидата технических наук

Научный руководитель
кандидат технических наук, доцент
Баранов Антон Викторович

Москва – 2025

Оглавление

Введение

Одним из приоритетов научно-технологического развития Российской Федерации определён переход к передовым технологиям проектирования и создания высокотехнологичной продукции, основанным на применении высокопроизводительных вычислительных систем [1]. При проведении фундаментальных и прикладных научных исследований в таких областях, как синтез лекарств, разработка новых материалов с заданными свойствами, прогнозирование месторождений полезных ископаемых, машинное обучение, анализ климатических изменений и других, применение высокопроизводительных вычислительных систем позволяет существенно сократить сроки исследования и ускорить разработку новых опытных образцов.

Современные высокопроизводительные вычислительные системы, часто называемые суперкомпьютерами, строятся на базе большого числа отдельных вычислительных узлов (ВУ), объединённых высокопроизводительной коммуникационной средой. Компоненты таких систем, как правило, представляют собой высокотехнологичные решения малой серийности и высокой стоимости, что обуславливает их эксплуатацию преимущественно в режиме коллективного пользования. Для получения доступа к суперкомпьютерным ресурсам пользователь оформляет запрос – вычислительное задание, включающее прикладную программу, требования к объёму вычислительных ресурсов, требуемое время выполнения, входные данные.

Жизненный цикл задания включает в себя постановку в очередь, ожидание в очереди, запуск задания в соответствии с расписанием, инициализацию вычислительных ресурсов для обработки задания, непосредственно обработку задания – выполнение прикладной программы, освобождение вычислительных ресурсов после завершения обработки. Реализацию жизненного цикла заданий осуществляют специальные программные системы управления заданиями (СУЗ), среди которых наибольшее распространение получили такие системы, как Slurm [1], PBS [2], IBM Spectrum LSF [3]. Среди отечественных разработок

необходимо выделить Систему управления прохождением параллельных заданий (СУППЗ) [4]. В процессе своего развития системы управления заданиями эволюционировали в сложные комплексные системы с множеством конфигурационных настроек, к которым следует отнести алгоритм планирования заданий и его параметры, схему приоритезации заданий, пользователей и пользовательских групп, различные ограничения в расписании, настройки подсистем квот и резервирования ресурсов и др.

С точки зрения повышения эффективности использования суперкомпьютерных ресурсов, научный интерес представляют исследования влияния параметров СУЗ и характеристик входного потока заданий, таких как интенсивность, однородность, средний размер задания и др. на показатели качества обработки заданий: загрузку вычислительных ресурсов, время ожидания заданий в очереди, длина очереди и другие [6, 7]. В настоящее время одним из главных инструментов для исследования СУЗ является имитационное моделирование, при проведении которого возникает задача валидации используемой имитационной модели.

Для имитационного моделирования СУЗ применяются такие симуляторы, как Alea [8], RM-Replay [9], Slurm Simulator [10-12], Batsim [13] AccaSim [14], MONARC [15], ElastiSim [16], CloudSim [17] и его модификации, например, CloudSim 7G [18] или модернизация, предложенная в работе В.В. Топоркова [19], а также программные платформы для построения имитационных моделей, например, GridSim [20] и GPSS [21]. В актуальных научных публикациях отмечаются трудности валидации имитационных моделей, которую авторы либо не проводят, либо рассчитывают интервальные статистические показатели качества, либо применяют методику визуального сравнения графиков изменения показателей качества. В частности, применяемые методы валидации в работах Н.А. Симакова [11], A. Lucero [10], M. D'Amico [12], PF. Dutot [13] дают только качественное представление о точности имитационной модели СУЗ. Вопросы оценки точности моделей рассматриваются в работах Balci O. [30-32], Carson J.S. [33], и других [34]. Результаты представленного в разделе 2.3 анализа выявили

отсутствие общепринятых методов оценки точности имитационных моделей при помощи количественного (численного) показателя, что обуславливает актуальность исследований и разработок методов и средств имитационного моделирования с количественной оценкой точности имитационной модели.

Важным этапом жизненного цикла задания являются инициализация задания и необходимых вычислительных ресурсов для его обработки. Во время инициализации вычислительные ресурсы не выполняют полезной работы, и с этой точки зрения затраченное на инициализацию время представляет собой накладные расходы. Это время определяется процедурой инициализации, которая для разных заданий может различаться. Задания с одинаковой процедурой инициализации будем называть заданиями одного типа. Определим накладные расходы на обработку задания (НРО) как соотношение времени инициализации и обработки задания. Задания можно разделить на две большие категории [60] – с малыми НРО, когда возможно пренебречь временем инициализации, и с большими НРО, когда время инициализации оказывает существенное влияние на показатель загрузки вычислительных ресурсов, и этим временем пренебречь нельзя.

С развитием гибридных архитектур суперкомпьютеров, расширением сферы применения облачных вычислений число заданий с большими НРО возрастает. Инициализация таких заданий может включать, помимо выделения подмножества вычислительных узлов и проверки их работоспособности, такие длительные действия, как перепрограммирование ускорителей на базе программируемых логических интегральных схем (ПЛИС), развёртывание виртуальных машин или контейнеров, инициализацию программы обработки задания, обращение в удалённую базу данных для загрузки входных данных и другие.

Одним из способов снижения накладных расходов при обработке заданий с высокими НРО является группировка однотипных заданий в пакеты или мета-задания. Запуск и завершение производится однократно для всего пакета заданий, а обработка пакета осуществляется путём последовательной обработки всех заданий пакета одного за другим. Анализ актуальных исследований по

группировке заданий показывает, что рассматриваются, как правило, потоки однотипных заданий. Способы формирования пакетов из однотипных заданий по таймеру и по числу заданий в очереди рассмотрены в работах M. Maheswaran [22], P. Rosemarry [23]. Метод формирования пакетов из однотипных заданий, основанный на сложности заданий и производительности вычислительных ресурсов, рассматривается в работах N. Muthuvelu [24], S. Gomathi [25]. Среди работ отечественных учёных в этой области следует отметить исследования В.В. Топоркова [26, 27], А.И. Костогрызова [28], А.С. Румянцева [29].

Анализ научных публикаций и результатов современных исследований в области организации высокопроизводительных вычислений позволяет сделать вывод, что для повышения эффективности использования суперкомпьютерных ресурсов актуальными являются исследования и разработки методов и средств пакетирования однотипных заданий с высокими НРО. В качестве основного инструмента при проведении исследований в этой области целесообразно применить имитационное моделирование.

Объектом исследования является имитационное моделирования систем управления заданиями. **Предмет исследования** – методы и средства имитационного моделирования систем управления заданиями.

Цель диссертационной работы состоит в разработке методов и средств моделирования для анализа и улучшения характеристик систем управления заданиями в суперкомпьютерах.

Для достижения цели необходимо решить следующие задачи исследования:

– анализ современных средств моделирования систем управления заданиями;

– разработка метода имитационного моделирования систем управления заданиями с оценкой точности модели;

– разработка и реализация архитектуры программного комплекса имитационного моделирования систем управления заданиями для исследования характеристик системы управления заданиями;

– разработка методики формирования пакетов заданий по типам; экспериментальная оценка её эффективности, определение степени влияния пакетирования на показатели качества обработки заданий с помощью разработанного программного комплекса имитационного моделирования.

Методология и методы исследования. В диссертации использованы методы проектирования программных систем, методы математической статистики, имитационное моделирование. Работа является продолжением исследований и разработок в области организации высокопроизводительных вычислений в режиме коллективного пользования, основы которых заложены в работах таких известных исследователей, как В.К. Левин, А.В. Забродин, Г.И. Савин, Вл.В. Воеводин, В.В. Корнеев, Б.М. Шабанов, В.В. Топорков, А.О. Лацис, В.В. Кореньков.

Научная новизна работы.

1. Разработан метод имитационного моделирования систем управления заданиями, отличающийся от известных применением количественного показателя близости выходных потоков заданий для сравнительной оценки точности различных имитационных моделей и поддержкой итеративного процесса настройки параметров модели.

2. Разработана архитектура программного комплекса имитационного моделирования систем управления заданиями, обеспечивающая интеграцию с функционирующими СУЗ (включая СУППЗ) без модификации их исходного кода и реализующая механизм обратной связи для адаптивной настройки параметров моделирования в соответствии с характеристиками входного потока заданий.

3. Разработана методика формирования пакетов заданий по типам, отличающаяся от аналогов использованием весов очередей на основе приоритета, времени ожидания и целесообразности формирования пакета, а также обеспечением очереди СУЗ нулевой длины в штатном режиме. Определены границы применимости методики в виде минимальных значений накладных расходов на обработку задания для входных потоков различной интенсивности и однородности.

Теоретической значимостью для развития имитационных моделей систем управления заданиями обладают предложенный количественный показатель точности моделей систем управления заданиями, метод имитационного моделирования систем управления заданиями, используемый для анализа и улучшения характеристик систем управления заданиями и оценки точности моделей, методика формирования пакетов заданий для обработки потока заданий разных типов.

Практическая значимость. Разработанный метод имитационного моделирования и методика формирования пакетов заданий реализованы в виде программного комплекса [35], который применяется в Отделении суперкомпьютерных систем и параллельных вычислений НИЦ «Курчатовский институт» при выполнении научно-исследовательских работ по программам фундаментальных научных исследований государственных академий наук. Внедрение комплекса позволило снизить накладные расходы на инициализацию заданий за счет применения методики формирования пакетов. Результаты работы могут быть применены в суперкомпьютерных центрах коллективного пользования для проведения имитационного моделирования систем управления заданиями с целю повышения эффективности использования вычислительных ресурсов.

Положения, выносимые на защиту

1. Метод имитационного моделирования систем управления заданиями, позволяющий оценивать точность исследуемых моделей при помощи количественного показателя близости выходных потоков заданий и улучшать характеристики систем управления заданиями для заданного входного потока заданий в ходе итеративного процесса настройки параметров модели.

2. Архитектура программного комплекса имитационного моделирования систем управления заданиями, позволяющая проводить динамическую настройку систем управления заданиями в соответствии с характеристиками входного потока заданий за счет реализации механизма обратной связи для адаптивной настройки параметров моделирования и обеспечивающая интеграцию с

различными системами управления заданиями без модификации их исходного кода.

3. Методика формирования пакетов заданий по типам, основанная на организации для заданий каждого типа отдельной очереди с определяемыми весами и позволяющая за счет группировки однотипных заданий в пакеты и однократной инициализации заданий пакета повысить полезную загрузку вычислительных ресурсов.

Достоверность результатов работы подтверждается результатами моделирования, опытом применения в реальных системах, согласованностью с данными, имеющимися в отечественной и зарубежной литературе.

Апробация работы. Основные результаты работы докладывались и обсуждались на международных, всероссийских и региональных научных конференциях, в том числе:

1. Всероссийская научная конференция «Научный сервис в сети Интернет: поиск новых решений», 17-22 сентября 2012 г., Новороссийск.

2. Всероссийская научная конференция «Научный сервис в сети Интернет: все грани параллелизма», 23-28 сентября 2013 г., Новороссийск.

3. Всероссийская научная конференция «Научный сервис в сети Интернет: многообразие суперкомпьютерных миров», 22-27 сентября 2014 г., Новороссийск.

4. Национальный Суперкомпьютерный Форум (НСКФ-2016), 29 ноября - 02 декабря 2016 г., Переславль-Залесский.

5. 2019 Federated Conference on Computer Science and Information Systems (FedCSIS), Leipzig, Germany, 01-04 сентября 2019.

6. Международная конференция «Суперкомпьютерные дни в России: Труды международной конференции», Москва, Россия, 21–22 сентября 2020 г.

7. 2020 15th Conference on Computer Science and Information Systems (FedCSIS), Sofia, Bulgaria, 6-9 сентября 2020 г.

8. Научный семинар в Межведомственном суперкомпьютерном центре Российской академии наук – филиале Федерального государственного учреждения «Федеральный научный центр Научно-исследовательский институт

системных исследований Российской академии наук», Москва, Россия, 9 марта 2021 г.

9. Международная конференция «Суперкомпьютерные дни в России», Москва, Россия, 26-27 сентября 2022 г.

10. Национальный суперкомпьютерный форум (НСКФ-2024), г. Переславль-Залесский, 26-29 ноября 2024 г.

Публикации. По теме диссертации автором опубликовано 23 печатные работы [36-58], из них 11 опубликованы в журналах, рекомендованных ВАК, в том числе 4 работы в журналах из перечня ВАК [38, 40, 45, 54] и 7 работ [47, 49, 50, 51, 53, 55, 57] в периодических научных журналах, индексируемых Web of Science и Scopus. Получено свидетельство о государственной регистрации программы для ЭВМ [35].

Структура и объем диссертации. Диссертация состоит из введения, четырёх глав, заключения, списка литературных источников и двух приложений. Каждая глава соответствует отдельному направлению исследования. Общий объем диссертации 125 страниц, в том числе 38 рисунков и 15 таблиц. Список литературы состоит из 122 источника.

В первой главе представлен существующий технологический тракт обработки заданий в суперкомпьютерах коллективного пользования, реализующий жизненный цикл заданий при помощи СУЗ. Рассматривается проблема высоких накладных расходов на инициализацию задания, существующие методы и средства её решения и формулируются задачи диссертационной работы.

Вторая глава посвящена имитационному моделированию систем управления заданиями. Экспериментально показана невозможность обеспечить полное соответствие результатов работы имитационной модели и реальной системы. Предложены понятия точности модели СУЗ в узком и широком смыслах. Для количественной оценки точности двух имитационных моделей предложен показатель близости выходных потоков заданий модели СУЗ. Разработан метод имитационного моделирования СУЗ, использующий

количественный показатель близости выходных потоков заданий для оценки точности разных имитационных моделей для заданного входного потока заданий. Представлены результаты применения разработанного метода для последовательности моделей СУЗ с заведомо разной точностью.

В третьей главе рассмотрена предлагаемая методика формирования пакетов по типам, состоящая из 5 этапов. Представлена частная модель системы обработки линейно масштабируемых заданий с поддержкой формирования пакетов заданий по типам. Для подтверждения эффективности методики представлен программный комплекс имитационного моделирования систем управления заданиями с новой архитектурой, которая отличается от известных решений количественной оценкой точности имитационного моделирования, динамической адаптивностью к входному потоку заданий и интеграционной совместимостью с произвольной системой управления заданиями.

В четвертой главе представлены результаты имитационного моделирования влияния разработанной методики пакетирования на показатели качества обработки заданий в СУЗ. Представлены результаты моделирования трёх серий экспериментов. В первой серии экспериментов показано положительное влияние методики формирования пакетов заданий по типам на полезную загрузку вычислителя. Во второй серии экспериментов определены границы применимости методики формирования пакетов заданий, при которой пакетирование улучшает показатели качества обработки заданий, в зависимости от минимального значения НРО, а также интенсивности и однородности входного потока заданий. В третьей серии экспериментов исследовано влияние показателя масштабирования для фиксированного входного потока заданий. Сформулирована методика автоматизированного подбора значения показателя масштабирования для фиксированного входного потока заданий.

В заключении представлены основные результаты и направления дальнейших исследований.

1. Обработка заданий в многопроцессорных вычислительных системах

Глава посвящена обзору предметной области и постановке основных задач исследования. В разделе 1.1 рассмотрен технологический процесс обработки информации в суперкомпьютерных центрах коллективного пользования. В разделе 1.2 рассмотрена типовая модель обработки суперкомпьютерных заданий. В разделе 1.3 представлен обзор показателей качества в системах управления заданиями. В разделе 1.4 рассмотрена проблема длительного времени инициализации вычислительных ресурсов. В разделе 1.5 приведён обзор существующих методов и средств формирования пакетов заданий. В разделе 1.6 сформулированы и обоснованы научно-технические задачи диссертационной работы.

1.1. Технологический процесс обработки информации в суперкомпьютерных центрах коллективного пользования

Современные суперкомпьютеры представляют собой высокопроизводительные вычислительные системы, строящиеся преимущественно на базе кластерной архитектуры. Основу таких систем составляют вычислительные узлы (ВУ), объединённые высокоскоростной коммуникационной средой [59].

Каждый вычислительный узел является автономным сервером, оснащённым одним или несколькими процессорами, оперативной и дисковой памятью, и может комплектоваться дополнительными ускорителями вычислений, такими как графические процессорные устройства (ГПУ) и программируемые логические интегральные схемы (ПЛИС). Как правило, ВУ суперкомпьютеров производятся на базе новейших архитектурных решений, имеют малую серийность и высокую стоимость. Суперкомпьютерные ВУ оснащаются специализированным стеком системного и инструментального программного обеспечения, который включает операционную систему, наборы драйверов для высокоскоростных сетевых устройств, различные библиотеки параллельного программирования, а также

системы виртуализации и контейнеризации, адаптированные для работы в высокопроизводительной среде [4].

Эксплуатация современных суперкомпьютеров требует наличия развитой инженерной инфраструктуры в составе специализированных центров обработки данных (ЦОД), включающей подсистемы охлаждения, бесперебойного электропитания, физической безопасности и другие. Обслуживание таких центров осуществляется высококвалифицированным персоналом, обладающим соответствующими знаниями, навыками и опытом. Высокая стоимость создания и эксплуатации суперкомпьютерных комплексов обусловливает работу суперкомпьютерных ЦОД как центров коллективного пользования (ЦКП).

Для получения доступа к суперкомпьютерным ресурсам пользователь оформляет запрос в виде вычислительного задания – информационного объекта, включающего программу, требования к ресурсам и, возможно, входные данные [59] . Требования к ресурсам включают планируемое время выполнения (или заказанное время счёта) и требуемый объём вычислительных ресурсов (количество узлов или процессорных ядер, объём оперативной памяти и др.), а также могут содержать требования к типу вычислительных узлов, о необходимости использования специализированных ускорителей (ГПУ, ПЛИС), требования к программному окружению (версии компиляторов, библиотеки), информацию о необходимой процедуре инициализации, требования к загрузке данных и др.

Рассмотрим жизненный цикл задания [40]. Задание поступает в очередь суперкомпьютерной системы, после прохождения которой начинает выполнение на выделенных ему вычислительных ресурсах. Выполнение задания i состоит из следующих этапов.

1. Инициализация ВУ a_i, которая может включать в себя выбор ВУ для обработки задания, выделение заданию выбранных ВУ, опциональная проверка работоспособности выбранных ВУ, предоставление прав доступа пользователю на выбранные ВУ.

2. Инициализация задания b_i, которая может включать в себя перепрограммирование ПЛИС, копирование на ВУ значительного объёма исходных данных, инициализацию ГПУ, обращение к некоторой базе данных для получения данных, запуск виртуальной программной платформы (виртуальной машины или контейнера) и т.п.

3. Непосредственно обработка задания c_i, то есть запуск пользовательской программы, реализующей прикладной алгоритм.

4. Завершение задания d_i, которое может включать в себя сохранение результатов обработки, освобождение выделенных ВУ, завершение всех пользовательских процессов, прекращение доступа пользователя на ВУ.

Время каждого из этих этапов может зависеть от числа ВУ (например, последовательная проверка работоспособности выделенных ВУ) или быть относительно постоянным (например, параллельная проверка работоспособности всех выделенных ВУ).

Назовём этапы 1, 2 и 4 инициализацией задания. Инициализацию задания можно отнести к накладным расходам на обработку задания. Задания с одинаковой процедурой инициализации будем называть заданиями одного типа. В русскоязычной литературе по теории расписаний используется термин «переналадка». Тип задания характеризуется длительностью накладных расходов, которые существенно зависят от процедуры его инициализации. Для задания одного типа возможна однократная инициализация с последующей последовательной обработкой нескольких заданий этого типа.

Накладные расходы на обработку (НРО) для задания *i* определим, как

$$z_i = \frac{a_i + b_i + d_i}{c_i} \times 100\% \qquad (1)$$

Например, для 1 минуты накладных расходов и 1 минуты обработки задания НРО составят 100%, а для 1 минуты накладных расходов и 2 минут обработки задания НРО составят или 50%.

Жизненный цикл задания в суперкомпьютерных системах реализуется специальным программным обеспечением – системой управления заданиями

(СУЗ), которая обеспечивает режим коллективного пользования. СУЗ управляет ресурсами суперкомпьютера и отвечает за приём входного потока пользовательских заданий, ведение очереди заданий, определение очередного задания для обработки, выделения заданию необходимого количества вычислительных ресурсов, запуск, управление и завершение задания на выделенных вычислительных ресурсах по истечении планируемого времени выполнения. Примерами распространённых СУЗ являются Slurm [1], PBS [2], IBM Spectrum LSF [3]. На ряде отечественных суперкомпьютеров применяется система управления прохождением параллельных заданий (СУППЗ) [4].

Рисунок 1 иллюстрирует типовую архитектуру суперкомпьютерной системы коллективного пользования.

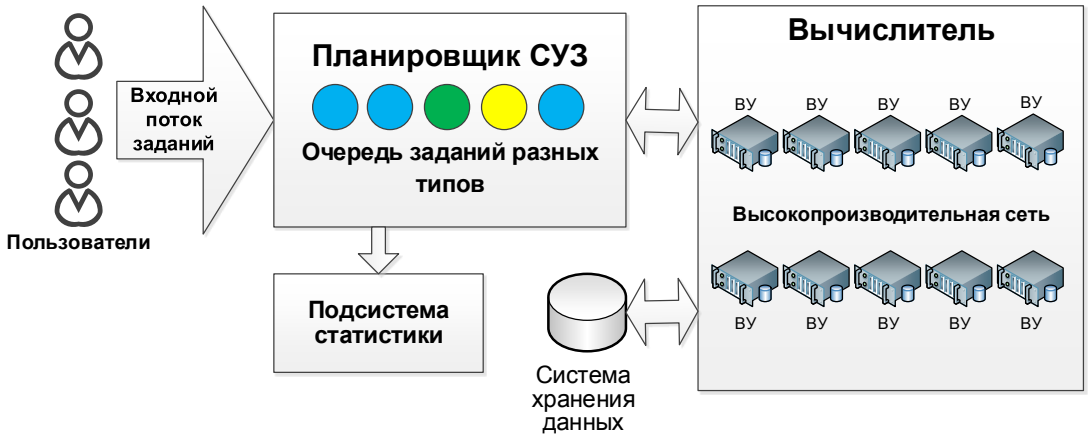

Рисунок 1 – Типовая архитектура суперкомпьютерной системы коллективного пользования

Входной поток заданий формируется из отправленных пользователями СУЗ заданий для обработки на суперкомпьютере. Время ожидания в очереди зависит от приоритета задания. Существуют два способа учёта приоритетов: вытесняющие (абсолютные) приоритеты и невытесняющие (относительные) приоритеты. При использовании вытесняющих приоритетов поступившее задание с высшим приоритетом немедленно вытесняет с выполнения задания с низшим приоритетом. При использовании невытесняющих приоритетов задание с высшим приоритетом становится первым в очереди, и запускается при освобождении необходимых вычислительных ресурсов. В настоящей работе рассматриваются невытесняющие приоритеты.

Планировщик СУЗ строит план запуска заданий (или расписание), то есть определяет время запуска и завершения заданий из очереди с учётом наличия доступных вычислительных узлов. В план запуска заданий постоянно вносятся изменения. Например, план запуска будет изменён при выходе из строя ВУ или досрочном завершении выполнения задания из-за неточного указания пользователем планируемого времени выполнения [61, 62].

Существуют два подхода к указания времени поступления задания в систему. В так называемом статическом планировании (offline-подход) время поступления всех заданий известно заранее, до начала их обработки. Offline-подход часто используется в теории расписаний [63]. При динамическом планировании (online-подход) время прихода очередного задания является случайной величиной с определёнными характеристиками, и задана вероятность появления очередного задания. Online-подход используется при изучении систем массового обслуживания и более полно отражает поток заданий, обрабатываемых в многопользовательских СУЗ. В настоящей работе рассматривается online-подход.

Подсистема статистики ведёт учёт информации о пользователях, заданиях, вычислительных узлах в журнале работы и статистических базах данных. Фиксируются различные события, возникающие в процессе функционирования СУЗ: поступление, запуск и завершение заданий, изменение состава ВУ, изменение параметров планирования (составления расписания запусков) заданий и другие. С помощью подсистемы статистики появляется возможность анализа состояния СУЗ в текущий момент времени или за некоторый период её функционирования ранее. В разделе 1.3 приведены показатели качества обработки заданий, которые может предоставлять подсистема статистики.

Результаты обработки задания сохраняются в системе хранения данных, и пользователю обеспечивается доступ к результатам обработки своих заданий.

В процессе своего развития СУЗ (Slurm, PBS, IBM Spectrum LSF, СУППЗ и др.) эволюционировали в сложные программные комплексы с множеством конфигурационных настроек, к которым следует отнести алгоритм планирования

заданий и его параметры, схему приоритезации заданий, пользователей и пользовательских групп, разделение вычислителя на логические разделы, различные ограничения в расписании, настройки подсистем квот и резервирования ресурсов и др. Алгоритм планирования может обеспечивать динамическое изменение приоритета заданий в зависимости от различных факторов, таких как время ожидания задания в очереди (чем дольше задание ожидает, тем выше его приоритет) и приоритет пользователя (например, можно понижать приоритет заданий для наиболее активных пользователей). Меняющиеся со временем характеристики входного потока заданий оказывают влияние на показатели качества обработки потока заданий [6].

Для дальнейшего исследования построим модель обработки заданий в СУЗ, определив существенные характеристики входного потока заданий и вычислителя.

1.2. Типовая модель обработки заданий в системах управления заданиями

Поток независимых друг от друга заданий $J_1, J_2 \ldots, J_n$ разных типов поступает на обработку в СУЗ, вычислитель которой состоит из m ВУ. Рисунок 2 иллюстрирует модель системы обработки заданий в СУЗ.

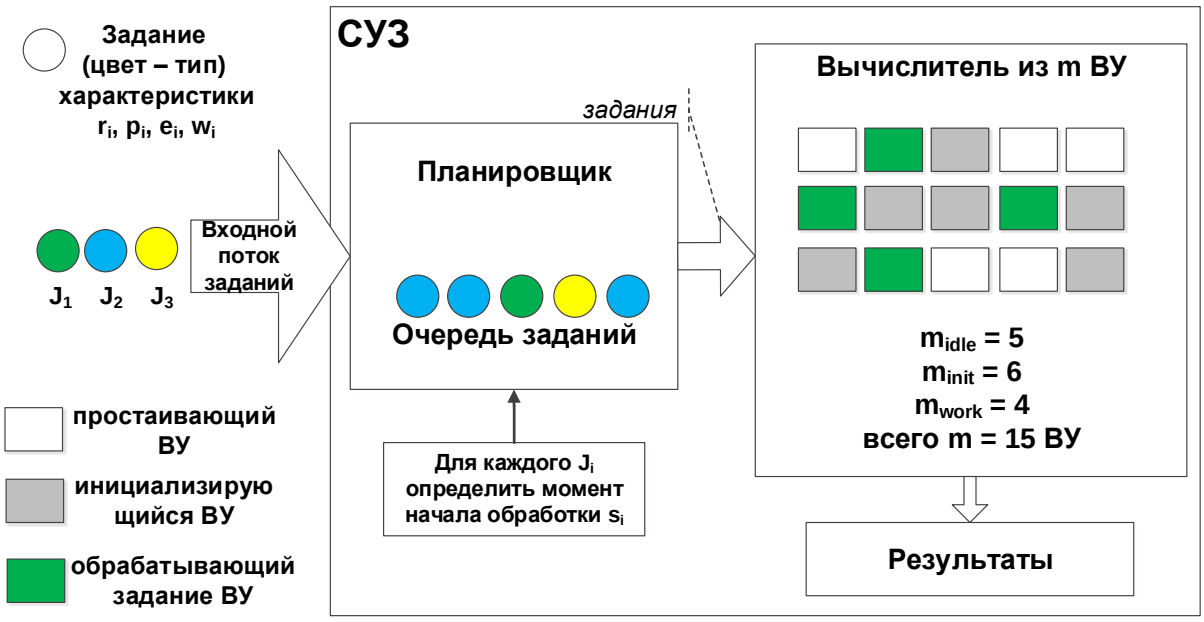

Рисунок 2 – Модель обработки заданий в СУЗ

Задание J_i, где $i \in \overline{1,n}$ представляет собой некоторую параллельную программу с исходными данными для обработки. Задание J_i имеет следующие характеристики:

1. Момент поступления задания в систему r_i.

2. Требуемое количество вычислительных узлов p_i.

3. Планируемое время выполнения e_i (задание принудительно завершится по истечении этого времени).

4. Реальное время выполнения w_i, $0 \leq w_i \leq e_i$, которое состоит из:

- времени инициализации ВУ a_i;
- время инициализации задания b_i;
- времени обработки задания c_i;
- времени завершения задания d_i.

5. опциональный тип задания $type_i$.

Производными характеристиками задания являются:

- время ожидания задания в очереди $q_i = s_i - r_i$;
- время пребывания в системе $f_i = q_i + w_i$;
- момент завершения задания $g_i = s_i + w_i$;
- накладные расходов на обработку (НРО) согласно (1).

Реальное время выполнения w_i недоступно при построении расписания планировщиком, значение w_i появляется после завершения выполнения задания.

Распространенной практикой является монопольное выделение ВУ заданию [7], то есть выделенный для выполнения одного задания ВУ, не может быть одновременно предоставлен еще какому-либо другому заданию, даже если на узле остаются свободные простаивающие процессоры или другие ресурсы. Это необходимо для исключения взаимовлияния заданий друг на друга и потенциального информационного обмена между заданиями, а также недопущения конкуренции нескольких заданий за физические ресурсы одного вычислительного узла, такие как процессор, оперативная или дисковая память, дополнительное оборудование. Будем считать, что ВУ является минимальной единицей вычислительного ресурса, которая может быть выделена заданию. Под

объёмом выделяемых заданию вычислительных ресурсов будем понимать количество вычислительных узлов.

Кроме того, будем считать, что все ВУ вычислителя являются гомогенными, то есть одинаковыми с точки зрения производительности и иных характеристик, однородность загрузки ВУ работой не имеет значения, коммуникационные связи между всеми узлами одинаковы. В теории расписаний зачастую рассматриваются обслуживающие устройства равной производительности [63]. Рассмотрение алгоритма выбора конкретных вычислительных узлов для обработки заданий выходит за рамки настоящей работы.

Вычислитель состоит из множества ВУ одинаковой производительности с одинаковыми коммуникационными связями между ВУ. Каждый вычислительный узел характеризуется состоянием «свободен», «инициализируется для обработки задания типа *j*», «обрабатывает задание J_i». Обозначим m_{idle} – число свободных ВУ в некоторый момент времени, m_{init} – число занятых запуском или завершением задания ВУ, m_{work} – число обрабатывающих задания ВУ. Тогда вычислитель может быть описан тройкой чисел (m_{idle}, m_{init}, m_{work}), где $m_{idle} + m_{init} + m_{work} = m$.

Пусть изменения в системе (добавление заданий, изменение количества доступных вычислительных узлов и др.) происходят в моменты времени t_i. Разобьём всё время работы системы на промежутки времени $\Delta t_i = t_{i+1} - t_i$. Обозначим $m(t_i)=m_i$ – количество работоспособных вычислительных узлов до i-го момента времени.

Под узло-часом будем понимать вычислительный ресурс, эквивалентный 1 вычислительному узлу, доступному для обработки задания в течение 1 часа. 1 узло-час означает, что 1 ВУ работал 1 час или 2 ВУ работали по 0,5 часов.

S_{all} – количество узло-часов, доступных для обработки заданий за определённый временной период.

$$S_{all}(t) = \sum_{i=0}^{t} m_i \Delta t_i \qquad (2)$$

S_{busy} – количество узло-часов, использованных для выполнения заданий за тот же период (учитываются и инициализирующиеся ВУ, и обрабатывающие задания).

$$S_{busy}(t) = \sum_{i=0}^{t}(m_{init} + m_{work})\Delta t_i \text{, где } m_{init}, m_{work} \in (0, m_i) \quad (3)$$

S_{work} – количество узло-часов, использованных для обработки заданий за тот же период (без учёта инициализирующихся узлов).

$$S_{work}(t) = \sum_{i=0}^{t} u_i \Delta t_i \text{, где } u_i \in (0, n_i) \quad (4)$$

Планировщику необходимо определить моменты времени начала обслуживания всех заданий s_i, $i \in \overline{1,n}$. При этом все задания должны быть полностью обеспечены ресурсами, то есть должно выполняться $\sum_{i=1}^{n} p_i \varphi_i(t) \leq m$, где $\varphi_i(t) = 1$, если задание i выполняется в момент времени t, и $\varphi_i(t) = 0$ в противном случае.

Несмотря на многообразие существующих методов и алгоритмов планирования заданий [78-80], в реальных высокопроизводительных системах планирования заданий наибольшее распространение получили алгоритмы FCFS и backfill. FCFS (First Come, First Served), также называемый FIFO (First Input, First Output), является примером самого простого алгоритма планирования и представляет собой обычную очередь [81].

Планируемое время выполнения, по истечении которого задание снимается с вычислений, используется в алгоритме обратного заполнения (backfill [64]), который используется по умолчанию в 90% СУЗ [82]. Алгоритм обратного заполнения позволяет использовать простаивающие ВУ для запуска заданий с низким приоритетом вне очереди, если их выполнение не повлияет на время старта более приоритетных заданий. Такое возможно, например, в случае, если для задания А, стоящего в очереди ранее, недостаточно ресурсов, и при этом задание Б, стоящее в очереди позже, успеет завершиться до момента, когда освободится достаточное количество ресурсов для запуска задания А. Никакое менее приоритетное задание не может занять ВУ так, чтобы это отодвинуло старт более приоритетного. Планировщик определяет время завершения выполняющихся заданий и освобождения занятых ВУ, используя заказанное

время выполнения, указанное пользователем для задания. Существуют различные модификации алгоритма обратного заполнения [83-87], такие как Conservative Backfill, Aggressive Backfill, EASY Backfill.

Рассмотрим существующие показатели качества обработки заданий в СУЗ и определим используемые в настоящей работе характеристики.

1.3. Показатели качества обработки заданий

В настоящий момент не сформировано единого подхода к определению показателей качества обработки заданий в СУЗ. Выбор показателей качества и критериев эффективности работы системы в каждом конкретном случае ложится на авторов проводимых исследований.

Некоторые показатели явно лидируют по количеству упоминаний в публикациях по тематике высокопроизводительных вычислений. Среди основных показателей качества обработки заданий в СУЗ можно выделить такие, как загрузка вычислителя и среднее значение времени нахождения задания в очереди.

Выбор показателей качества обусловлен особенностями рассматриваемой системы. Разные категории специалистов, работающие с системой, заинтересованы в оптимизации различных показателей. Например, для пользователя СУЗ актуально минимизировать время ожидания их заданий в очереди, владельца вычислительных ресурсов интересует показатель загрузки вычислительных ресурсов. Более того, важность показателей качества у различных категорий специалистов в различное время может значительно меняться. Из-за динамического изменения важности показателей качества нет, и, весьма вероятно, не будет единого показателя качества, подходящего всем и всегда.

При этом многие показатели качества взаимозависимы [86]. Так, любые перестановки заданий в очереди с целью повышения загрузки ресурсов неизбежно приводят к изменению времени ожидания запуска заданий. Реализация стратегии минимального среднего времени обслуживания заданий, весьма вероятно, приведёт к уменьшению средней загрузки вычислительных ресурсов.

Современная система управления заданиями – сложная в конфигурировании система с множеством настраиваемых параметров. Изменение любого из параметров может повлечь за собой значительные, но неочевидные изменения показателей качества. Заранее предсказать изменение показателей качества часто не представляется возможным.

Расчёт показателей качества возможно проводить по статистическим данным работы СУЗ. Для исследования влияния настроек СУЗ на показатели качества требуется, чтобы исследуемая СУЗ с нужными настройками функционировала на имеющейся вычислительной системе и обслуживала реальные задания от пользователей. Подобный эксперимент может негативно сказаться на качестве обработки текущих заданий и занимает длительное время, поскольку расчёт показателей качества для нивелирования влияния случайных процессов, являющихся следствием вероятностных характеристик потока заданий имеет смысл проводить только за большой промежуток времени. В разделе 2.1 рассматривается порядок исследования показателей качества на имитационной модели СУЗ, что позволяет избежать влияния экспериментов на обслуживание реальных пользователей.

В результате обзора наиболее часто используемых в отечественной [26, 71, 74] и зарубежной литературе [7, 8, 75] были отобраны следующие показатели качества, каждый из которых будет описан далее.

1. Полная загрузка вычислительных ресурсов.
2. Полезная загрузка вычислительных ресурсов.
3. Среднее (или медианное) время ожидания задания в очереди.
4. Средняя длина очереди.

Загрузка вычислительных ресурсов (иногда используется термин утилизация). Некоторые авторы используют «простой вычислительных ресурсов» как обратный загрузке показатель качества. Загрузка вычислительных ресурсов обычно определяется как отношение количества работающих ресурсов к числу всех доступных за определённый временной период. Если за 5 часов работы вычислительной системы из 10 вычислительных узлов на обработку заданий

затрачено 40 узло-часов (т. е. 10 узлов работали 4 часа, или 8 узлов работали 5 часов), то утилизация составит 40 / (5 x 10) = 0,8. Простой ресурсов, соответственно, составит 1 – 0,8 = 0,2.

Для настоящей работы определим показатели полной и полезной загрузки. Показатель полной загрузки эквивалентен общепринятой утилизации ресурсов. **Показатель полной загрузки** (далее просто загрузка) вычислительных ресурсов *L* за определённый период времени рассчитывается по формуле

$$\mathrm{L(t)} = \frac{\mathrm{S_{busy}(t)}}{\mathrm{S_{all}(t)}} \times 100\% \quad (5)$$

Полная загрузка не позволяет учесть, что инициализируемые ресурсы недоступны для вычислений, и, фактически, должны быть отнесены к простою. **Показатель полезной загрузки** вычислительных ресурсов *U* за определённый период времени может быть определён как

$$\mathrm{U(t)} = \frac{\mathrm{S_{work}(t)}}{\mathrm{S_{all}(t)}} \times 100\% \quad (6)$$

Время ожидания задания в очереди с момента поступления задания в систему до начала обработки. Встречается также название «время ожидания запуска». Определим **среднее время ожидания задания в очереди** *Q*. Для *k* заданий этот показатель может быть рассчитан как

$$\mathrm{Q} = \frac{\sum_{\mathrm{i=0}}^{\mathrm{k}} \mathrm{q_i}}{\mathrm{k}} \quad (7)$$

Время ожидания в очереди относительно планируемого времени выполнения. Схожей характеристикой является замедление (slowdown). Очевидно, что ожидание запуска в 1 час может быть приемлемым для задания с заказанным временем счёта в 24 часа. Тот же час ожидания для задания с заказанным временем счёта в 10 минут уже может являться показателем низкого качества обработки.

Определим среднее приведённое время ожидания задания в очереди Q_p

$$\mathrm{Q}_p = \frac{\sum_{\mathrm{i=0}}^{\mathrm{k}} {q_i}/{e_i}}{\mathrm{k}} \quad (8)$$

Определим **среднее время пребывания задания в система** *F* (или время полного обслуживания). Для *k* заданий этот показатель может быть рассчитан как

$$F = \frac{\sum_{i=0}^{k} f_i}{k} \quad (9)$$

Кроме того, в других исследованиях рассматриваются ещё ряд показателей качества, использование которых выходит за рамки настоящего исследования. Выделяют время полного обслуживания **задания** от поступления в систему до окончания обработки, среднее время обработки всех заданий, учёт директивных сроков завершения работ, средний темп, пропускную способность системы, «честность» по отношению к задаче (как дисперсию времени ожидания задания в очереди), сбалансированность загрузки (как равномерную загруженность всех ресурсов вычислительной системы), энергоэффективность, экономические затраты, эффективность системы (в виде количественной характеристики «ресурсозатратности» достижения требуемого качества).

Рассмотрим существующие накладные расходы на обработку задания и задачу повышения качества обработки заданий разных типов с большими НРО.

1.4. Обработка заданий с большими накладными расходами на обработку

Существует несколько категорий заданий с длительной инициализацией:

1. Для задания требуется инициализация высокопроизводительного ускорителя (ГПУ или ПЛИС). Перепрограммирование ПЛИС и загрузка программы на ГПУ могут занимать длительное время [65].

2. Для задания необходима специфическая виртуальная программная платформа, развёртывание такой платформы перед стартом задания может занять существенное время. Для реализации виртуальной программной платформы используют виртуальные машины [66] или контейнеры [67].

3. Для задания необходимо до начала обработки загрузить на вычислительные узлы значительный объём входных или вспомогательных данных [68].

Малое время обработки заданий характерно для областей, в которых законченной единицей обработки является небольшой фрагмент работы. Такая ситуация возникает при решении задач фото- и видеобработки [70], таких как

анализ движения, расчёт освещённости после добавления или удаления объектов, компьютерная анимация. Обработка одного кадра может занимать непродолжительное время. Схожая ситуация возникает в таких областях, как секвенирование генома, тестирование лекарств, ядерная физика и био-информатика.

На базе работы [68] проиллюстрируем численную оценку доли накладных расходов в Aneka. Aneka – это способ организации облачных вычислений в виде платформы-как-сервис (PaaS), предоставляющий возможность построения распределённых приложений на базе Грид-систем или в облачной среде, в том числе гибридной. В работе проводятся эксперименты на примере набора данных для BLAST (Basic Local Alignment Search Tool – средство поиска основного локального выравнивания), широко используемой для поиска похожих последовательностей белков и ДНК. В работе [68] приведены данные, что для решаемой задачи BLAST динамическое предоставление виртуальных машин с применением Aneka на базе облачного провайдера Azure от Microsoft в среднем занимает 150 секунд (усреднённое время для 10 запусков). При этом среднее время выполнения задания составляет 10 минут (600 секунд). Таким образом, НРО составляют 150/600 = 25%.

В настоящей работе рассматриваются накладные расходы, которые являются константой для всех заданий одного типа. Подсистема статистики большинства СУЗ учитывает выполнение задания как неделимую величину, не выделяя инициализацию задания. Таким образом, подсистема статистики СУЗ относит инициализацию к полезной загрузке, завышая реальную загрузку ресурсов. Более того, при инициализации могут быть задействованы процессор, оперативная память и сеть, так что даже интеллектуальная система мониторинга не способна достоверно отличить инициализацию задания от процесса обработки. В то же время потери времени на инициализацию равнозначны потери эффективности использования ресурсов.

При этом объём накладных расходов является случайной величиной и может зависеть от многих факторов. Для примера, типовые накладные расходы

могут составлять 10-20 секунд для СУЗ общего назначения, 50-70 секунд для узлов с ГПУ, 150 секунд при перепрограммировании ПЛИС, 300 секунд для запуска виртуальных машин. Как будет показано в разделе 3, существенное значение для дальнейшего изложения будет иметь не абсолютное значение накладных расходов, а НРО согласно формуле (1), то есть отношение времени накладных расходов на обработку задания ко времени обработки задания.

Для заданий с большими НРО возникает ситуация, при которой накладные расходы на обработку задания начинают заметно влиять на качество обработки заданий, что значительно уменьшает эффективность использования вычислителя. Для уменьшения доли накладных расходов рассмотрим объединение заданий одного типа в пакет или мета-задание. Затраченное на инициализацию время может быть уменьшено за счёт однократной инициализации ресурсов для выполнения нескольких заданий одного типа. Будем называть пакетом группу заданий одного типа, выполняемых на однократно инициализированных ресурсах. Пакет заданий может включать в себя одно и более заданий одного типа.

Рисунок 3 иллюстрирует пример формирования пакетов заданий для заданных 5 входных пользовательских заданий с номерами от 1 до 5 Задания 1, 2 и 3 были сгруппированы в один пакет заданий, задания 4 и 5 – в другой пакет.

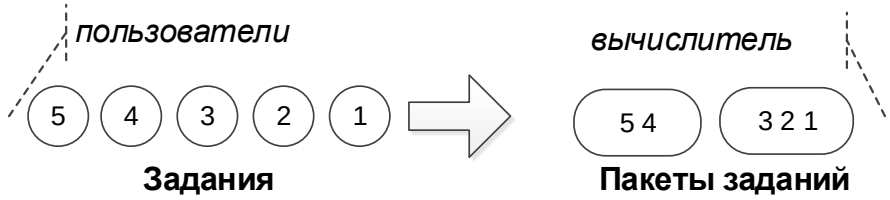

Рисунок 3 – Пример заданий и соответствующих им пакетов заданий

Рассмотрим приоритеты заданий в пакете. Пусть есть два типа заданий, в очереди находятся одно высокоприоритетное задание 1 типа и десять заданий среднего приоритета 2 типа. Для того, чтобы определить, из каких заданий сформировать пакет, можно предложить вычислить взвешенную сумму приоритетов всех заданий одного типа. В этом случае нужно произвести

экспертное оценивание весов приоритетов, что является отдельной задачей, кроме того, зависящей от особенностей потока обрабатываемых заданий.

При объединении заданий в пакеты возникает задача определения времени формирования пакета заданий и состава пакета, решение которой рассмотрено в разделе 3.2. При этом требует проработки вопрос определения нижней границы НРО, при котором целесообразно использование пакетирования, решение рассмотрено в разделе 4.3.

1.5. Анализ современных методов и средств формирования пакетов заданий

Рассмотрим существующие методы и средства формирования пакетов заданий в СУЗ. В рассмотренной литературе пакетирование заданий используется для уменьшения накладных расходов на запуск одного задания (издержки назначения задания на узлы) [72]. Часто речь идёт о распределённых вычислениях в грид [73], поэтому единицей планирования является обобщенный «вычислительные ресурс», которым может являться как один вычислительный узел, так и суперкомпьютер целиком.

Nitro High Throughput [76] в СУЗ Moab [77] позволяет группировать задания в пакеты для запуска на одном вычислительном узле, но не позволяет использовать задания разных типов и производить обработку пакета более, чем на одном ВУ.

В статье [22] предлагаются две стратегии пакетирования. Первая стратегия заключается в формировании пакета заданий по таймеру (например, каждые 10 минут). Вторая стратегия заключается в ожидании указанного администратором системы количества заданий (например, 50 заданий), при наличии которого формируется пакет заданий. Для практических нужд предлагается ввести параметр «максимальное время ожидания задания в очереди на формирования пакета», по достижении которого обязательно формируется пакет заданий. Разные типы заданий не рассматриваются.

В работе [24] предложен алгоритм группировки мелкозернистых заданий в крупнозернистые для уменьшения издержек назначения заданий на узлы без

учёта типов заданий. Администратор системы указывает «размер зерна» T, то есть желаемое время обработки пакета заданий. Пусть каждое из n заданий требует для своей обработки M_j миллионов инструкций, где $j \in \overline{1,n}$. Тогда на вычислительный ресурс с производительностью V миллионов инструкций в секунду сформируется такой пакет из k заданий, чтобы обработка пакета выполнялась за время T. Другими словами, должно выполняться соотношение

$$\frac{\sum_{j=1}^{k} M_i}{V} \leq T \qquad (10)$$

В работе [25] при группировке заданий учитывается размер задания в миллионах инструкций, требуемая заданию оперативная память, размер данных задания в мегабайтах и заказанное время выполнения. Со стороны вычислителя учитывается производительность, доступная оперативная память, пропускная способность сети. Алгоритм состоит в следующем: выбирается одно задание, раньше всех поступившее в систему (обычная очередь). Выбирается свободный вычислительный ресурс наивысшей вычислительной производительности. Пакет формируется из выбранного задания, к которому добавляются задания из очереди до тех пор, пока на выбранном вычислительном ресурсе есть свободная оперативная память и суммарное время обработки пакета не превысит заданную величину. Сформированный пакет размещается на выбранном вычислительном ресурсе. После чего процедура повторяется. Эксперименты по определению качества обработки заданий проведены в GridSim. Измерены следующие показатели качества: время выполнения всех заданий, загрузка вычислителя, простой CPU. Разные типы заданий не рассматриваются.

В работе [88] изложен алгоритм, похожий на предложенный в работе [25]. Исследуется зависимость времени обработки всех заданий от размера формируемого пакета заданий. Эксперименты по определению качества обработки заданий проведены в GridSim. Разные типы заданий не рассматриваются.

В работе [23] рассматривается влияние порядка обработки заданий внутри пакета на показатели качества обработки заданий. Эксперимент проводился в

Matlab. Использовались следующие показатели качества: суммарное время обработки всех заданий, полная загрузка вычислительных ресурсов. Разные типы заданий не рассматриваются.

На производстве исследуются преимущественно вопросы сокращения времени инициализации (переналадки) за счёт технических или организационных способов [94-98]. Рассматриваются разные типы заданий, но расчёты проводятся для 10 заданий, известных заранее (offline-подход).

В работе [26] термин «пакет» употребляется в значении «подмножества» при разбиении всего множества заданий на подмножества для снижения сложности алгоритма планирования заданий. Не предлагается алгоритма выбора заданий в пакет.

Рассмотренные методы и средства формирования пакетов заданий призваны сократить расходы на инициализацию ресурсов, возникающие вследствие выделения большого числа ВУ для задания. Не учитываются существующие типы заданий, для которых необходима различная инициализация. Не найдено ответа на вопрос, сколько ресурсов выделять для обработки пакета заданий. Большинство исследователей решают обратную задачу – сколько заданий включить в пакет, который выполнится на выбранном вычислительном ресурсе. Проведённый обзор существующих систем управления заданиями показал, что на текущий момент в рассмотренных СУЗ не поддерживается группировка заданий в пакеты для запуска на множестве вычислительных узлов.

1.6. Задача построения программного комплекса имитационного моделирования систем управления заданиями

Результаты представленного в разделах 1.4 и 1.5 анализа позволяют утверждать, что для снижения накладных расходов за счёт однократной инициализации группы заданий необходимо разработать новую методику формирования пакетов заданий по типам (ФПЗТ). Методика должна применяться для непрерывно поступающего потока заданий разных типов. Методика должна определять в том числе момент запуска пакета заданий, состав пакета заданий

одного типа и необходимый объём вычислительных ресурсов для выполнения сформированного пакета заданий.

За счёт уменьшения времени, затрачиваемого на инициализацию задания, пакетирование должно положительно повлиять на показатели качества. При этом невозможно однозначно предсказать влияние пакетирования на качество обработки заданий. В условиях постоянного изменения характеристик входного потока заданий необходимо исследовать влияние пакетирования в различных условиях. Для исследования влияния методики формирования пакетов и её параметров на показатели качества актуальным становится задача имитационного моделирования [99].

Имитационная модель отличается от реальной системы, что приводит к необходимости определения точности моделирования. Существующие способы оценки точности моделей (рассматриваются в разделе 2.3.3) либо ограничиваются сравнением интегральных показателей качества (таких как загрузка или среднее время ожидания задания в очереди), либо визуальным сравнением графиком показателей качества, либо сводятся к визуальному анализу планов запуска. Это не позволяет проводить корректное сравнительное исследование различных моделей в идентичных условиях. Необходима разработка нового метода имитационного моделирования, к которому предъявим следующие требования.

1. **Универсальность**. Метод должен позволять работать с различными моделями СУЗ.

2. **Количественная оценка точности модели**. Для выбора наиболее точной модели необходим численный показатель.

3. **Цикличность**. Метод должен поддерживать итеративный процесс исследования, позволяя уточнять параметры модели и входные данные для исследования различных условий.

Метод должен быть реализован в составе программного комплекса имитационного моделирования с новой архитектурой, которая позволит проводить исследования для широкого спектра входных потоков заданий с динамической настройкой параметров моделируемой СУЗ, в том числе

параметров подсистемы объединения заданий в пакеты. К программному комплексу предъявим следующие требования.

1. **Модульность.** Программный комплекс должен быть разделена на независимые модули. Это позволяет заменять отдельные модули без изменения всей системы.

2. **Адаптивность** (или **оптимизация на основе обратной связи**). Программный комплекс должен поддерживать автоматизированную адаптацию параметров формирования пакетов заданий под произвольный входной поток заданий.

3. **Интеграционная совместимость с произвольной СУЗ.** Реализованный программный комплекс должен быть интегрируем с произвольной СУЗ с небольшой модификацией.

4. **Формирование пакетов заданий по типам** для уменьшения накладных расходов на инициализацию заданий.

5. Возможность **оценки точности** имитационного моделирования.

6. Возможность **проведения имитационного моделирования** для оценки показателей качества для различных входных потоков.

Выводы по главе 1

Проведённый анализ предметной области показал, что современные высокопроизводительные вычислительные системы эксплуатируются преимущественно в режиме коллективного пользования под управлением специальных программных систем управления заданиями, таких как Slurm, PBS, IBM Spectrum LSF, СУППЗ. Системы управления заданиями являются сложными программными комплексами с множеством настраиваемых параметров. Для исследования и улучшения характеристик этих систем необходимо проводить моделирование, при котором возникает задача определения точности модели как степени соответствия реальной системе. Сделан вывод о необходимости разработки нового метода моделирования систем управления заданиями, который должен быть применим к произвольной системе управления заданиями, позволит

оценивать точность исследуемых моделей и должен поддерживать итеративный процесс исследования с пошаговым уточнением параметров модели и входных данных.

Выявлена проблема длительной инициализации пользовательских заданий, которая может быть решена с помощью объединения заданий в пакеты заданий по типам с однократной инициализацией. Задания с одинаковой процедурой инициализации относятся к одному типу. Анализ актуальных отечественных и зарубежных исследований по группировке заданий показывает, что в них рассматриваются, как правило, потоки однотипных заданий. Представленные в публикациях методы формирования пакетов заданий, в основном, призваны сократить расходы на инициализацию ресурсов, возникающие вследствие выделения большого числа вычислительных узлов для задания. В существующих методах не учитываются множественные типы заданий, для каждого из которых необходима отдельная процедура инициализации. Сделан вывод о недостаточной проработке вопроса пакетирования заданий для снижения накладных расходов на инициализацию.

Для исследования влияния пакетирования заданий на взаимозависимые показателями качества обработки заданий для различных входных потоков необходимо проводить имитационное моделирование, которое позволит оценить эффект от внедрения предлагаемого решения формирования пакетов как для существующего входного потока заданий исследуемого центра коллективного пользования, так и для ожидаемых в будущем вариантов входных потоков. С этой целью необходимо разработать программный комплекс имитационного моделирования систем управления заданиями с новой архитектурой, которая позволит проводить исследования для широкого спектра входных потоков заданий с динамической настройкой параметров моделируемой СУЗ, в том числе параметров подсистемы объединения заданий в пакеты.

2. Моделирование систем управления заданиями

В главе рассмотрены методы и средства моделирования (раздел 2.1) и способы формирования входного потока заданий (раздел 2.2). Рассмотрен вопрос точности имитационного моделирования системы управления заданиями (раздел 2.3) и предложен способ оценки точности имитационной модели (раздел 2.4), который лёг в основу разработанного метода имитационного моделирования (раздел 2.5). Предложенный метод апробирован в серии экспериментов на имитационных моделях с заведомо разной точностью (раздел 2.6).

2.1. Методы и средства моделирования систем управления заданиями

Рисунок 4 иллюстрирует общий случай моделирования СУЗ, который представляет собой процесс преобразования входного потока заданий в выходной поток. На вход модели СУЗ поступает входной поток заданий с некоторыми характеристиками (представлены в разделе 1.2). В процессе работы модели происходят некоторые внутренние события, такие как запуск и завершение задания. Результатом работы модели СУЗ является выходной модельный поток заданий с другим набором характеристик. Обозначим характеристики выходного потока заглавными буквами.

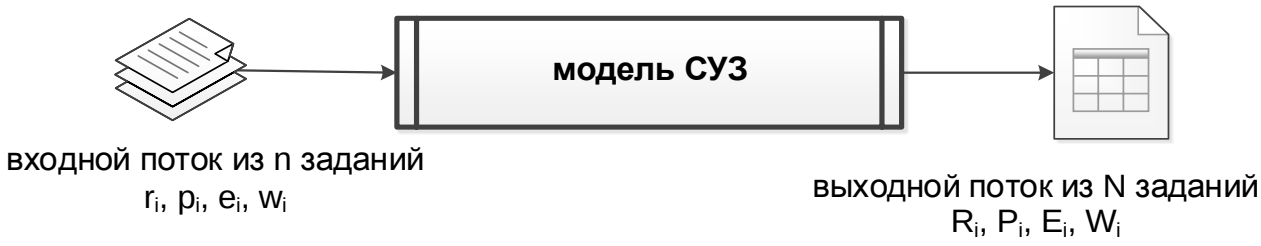

Рисунок 4 – Моделирование СУЗ

Существующие методы исследования системы управления заданиями возможно свести к следующим вариантам [45]:

- натурный эксперимент с существующей СУЗ;
- построение и исследование аналитической модели СУЗ;
- исследование СУЗ с виртуальным вычислителем;

− построение и исследование имитационной модели СУЗ.

Под **натурным экспериментом** будем понимать воспроизведение входного потока событий на реально работающем суперкомпьютере. По определению, модель СУЗ в натурном эксперименте будет точно воспроизводить реальную СУЗ. Тем не менее, далее покажем, что натурный эксперимент не может обеспечить воспроизведение результатов моделирования (т.е. выходного потока заданий) со 100% точностью.

Построение и исследование **аналитической модели СУЗ** (далее – аналитическая модель). При помощи методов теории массового обслуживания возможно построить и исследовать аналитическую модель СУЗ, рассчитывая необходимые показатели качества. В этой области следует выделить успехи отечественной школы теории массового обслуживания. В таких работах, как [100-102] представлены результаты аналитического исследования приоритетных систем обслуживания, в том числе многоэтапных приоритетных систем как наиболее общих моделей высокопроизводительных расчетов на ЭВМ [100], рассмотрены методы анализа мультипрограммных систем [101] их применение для исследования сетей ЭВМ [102]. В работах [28, 103] рассмотрены вопросы пакетной обработки заявок и комбинации различных дисциплин приоритетного обслуживания заявок в вычислительных системах.

В теории массового обслуживания суперкомпьютерные системы, состоящие из множества узлов, получили название многосерверных систем [104]. При этом в работах [105-107] отмечаются трудности применения методов теории массового обслуживания для исследования СУЗ, такие как возможность простоев ВУ при непустой очереди или «тяжелые хвосты распределений», ограничивающие применение при анализе модели экспоненциальных распределений. Нахождение явных решений для характеристик СУЗ считается трудоемким даже для систем малого размера [107]. Кроме этого, аналитическая модель позволяет исследовать влияние изменений параметров СУЗ на её интегральные показатели качества, но не предоставляет механизма предсказания времени запуска отдельных заданий, необходимого для достижения цели работы. Перечисленные факторы заставляют

исследователей прибегать к различным методам имитационного моделирования и машинного обучения [108].

Простейшим методом имитационного моделирование является исследование **СУЗ с виртуальным вычислителем** (далее – СУЗ с ВВ). Метод предполагает работу СУЗ в режиме, когда вместо запуска заданий на вычислительных узлах делается пометка о том, что узлы заняты, например, Slurm Simulator [10, 90, 91] или симулятор СУППЗ [48]. Реальных вычислений при обработке заданий при этом не производится [92]. Можно выделить два режима использования СУЗ с ВВ: в режиме реального времени и в режиме модельного времени. В режиме реального времени СУЗ с ВВ функционирует с той же скоростью, что и в натурном эксперименте, что позволяет говорить о точности, схожей с натурным экспериментом. При этом не требуются существенные вычислительные ресурсы, необходимые для натурного эксперимента. В основе исследования **СУЗ с ВВ в режиме модельного времени** лежит идея о «продвижении» системного времени в те моменты, когда с точки зрения СУЗ событий не происходит. Для примера, если в некоторый момент времени эксперимента новых заданий не поступает, в текущий момент одно задание обрабатывается, и оно завершится через час – то существует возможность не ожидать этот час, а пропустить его. За счёт такого подхода возможно существенно ускорить эксперимент. СУЗ с ВВ в режиме модельного времени обеспечивает высокую точность, не требуя существенных вычислительных ресурсов и уменьшая время проведения эксперимента. Для реализации СУЗ с ВВ в режиме модельного времени требуется разработка программного средства продвижения системного времени, после которой необходимо удостовериться в точности воспроизведения эксперимента. Экономия времени на проведения эксперимента зависит от реализации продвижения времени и входных данных для эксперимента.

Построение и исследование **построенной имитационной модели СУЗ** [93] (далее – имитационная модель). Существуют специализированные языки для построения имитационных моделей, как AnyLogic [109], ExtendSIM [110],

GridSim [20], GPSS [21], Micro Saint [111], Simulink [112]. Языки моделирования позволяют сосредоточится на описании имитационной модели, обеспечивая сам процесс моделирования – работу модельного времени и взаимодействие объектов в системе. Кроме специализированных языков моделирования существуют так называемые **симуляторы СУЗ**: Alea [8], RM-Replay [9], Batsim [13] AccaSim [14], CloudSim [17], OptorSim [113], WorkflowSim [114]. Отдельно можно выделить категорию эмуляторов, например, MicroGrid [115], которые позволяют перейти к полунатурному моделированию, совмещая компоненты модели и реальной системы.

Существующие средства имитационного моделирования позволяют проводить эксперименты на любом входном потоке заданий. Рассмотрим способы формирования входного потока заданий.

2.2. Способы формирования входного потока заданий для имитационного моделирования

Можно выделить три способа формирования входного потока заданий [44]. Первый способ состоит в использовании журнала учета заданий реальной СУЗ. Способ позволяет воспроизвести входной поток заданий реальной суперкомпьютерной системы с учетом всех ее особенностей. Достоинством такого способа является точное отображение ситуации в существующей системе. К недостаткам можно отнести сложность выбора периода для симуляции (выбранный случайным образом отрезок времени может содержать редкие события вроде праздников или профилактики, которые существенно влияют на показатели качества) и невозможность исследования вариантов изменившегося входного потока (например, исследование ситуации, при которой вдвое возрастёт число поступающий заданий).

Второй способ основан на существующем стандарте записи журнала заданий SWF [116] (Standard Workload Format), в котором публикуются журналы заданий некоторых суперкомпьютеров, в том числе университетских. Использование журналов сторонних СУЗ позволяет провести исследование для

широкого круга разных суперкомпьютерных систем с различными характеристиками входных потоков заданий. Использование публичных SWF-журналов позволяет также воспроизвести эксперименты других исследователей. Существенным минусом является неполнота потока заданий: в SWF отражены только события, связанные с продвижением заданий в очереди, и отсутствует информация об изменении характеристик решающего поля (изменении числа ВУ), удалениях заданий из очереди или прерываниях выполнения заданий пользователем.

Третий способ заключается в генерации входного потока заданий. Каждый параметр задания (время поступления, заказанное и реальное время выполнения, требуемые вычислительные ресурсы) представляет собой случайную величину с определённым законом распределения. Закон и параметры распределения подбираются, как правило, на основе анализа журналов событий исследуемых суперкомпьютерных систем. В ряде работ [117-118] изучаются распределения параметров заданий для входных потоков различных СУЗ, предложены наиболее подходящие распределения и числовые параметры этих распределений. Этот способ позволяет формировать несколько разных экземпляров входных потоков с одинаковыми распределениями.

Рисунок 5 демонстрирует общую схему получения входного потока заданий.

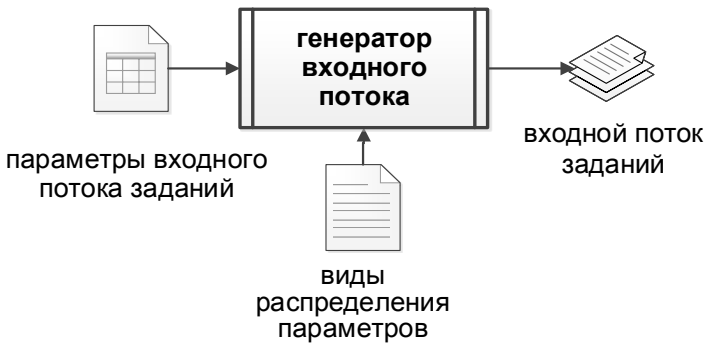

Рисунок 5 – Схема генерации входного потока заданий

Исходными данными для генератора являются виды функций распределения случайных величин, соответствующих параметрам заданий, и параметры этих распределений.

Преимуществом генерации входного потока заданий является возможность исследования СУЗ в различных условиях путём изменения параметров функций распределения, а при необходимости и видов этих функций. Например, возможно исследовать влияние интенсивности входного потока заданий путем генерации множества входных потоков разной интенсивности.

Для любой имитационной модели необходимо производить валидацию результатов экспериментов с целью определения точности модели. Для этого рассмотрим предлагаемый количественный показатель точности модели.

2.3. Точность модели системы управления заданиями

2.3.1 Точность модели СУЗ в узком и широком смыслах

Под точностью моделирования понимается степень схожести результатов экспериментов и реальной эксплуатации системы той же конфигурации. Для натурного эксперимента и СУЗ с ВВ эксперименты проводятся с реальной СУЗ, что позволяет обеспечить некоторую точность полученных в эксперименте результатов, сравнимую с точностью повторной обработки заданий на реальной системе. Как будет показано в разделе 2.3.2, 100% точность натурного эксперимента не достижима. Для имитационной модели необходимо производить оценку точности получаемых результатов. Модель, по точности не отличающаяся от натурного эксперимента, адекватно моделирует реальную систему и может быть использована в качестве её замены.

Возможно выделить два способа оценки точности модели. Точность в узком смысле определим степенью схожести результирующего потока внутренних событий с аналогичным внутренним потоком событий реальной системы. Очевидно, что точная в узлом смысле модель будет иметь идентичные с реальной системой статистические показатели качества, рассчитываемые за интервал времени: загрузку решающего поля суперкомпьютера, среднее время ожидания задания в очереди, среднюю длину очереди и другие.

Точность в широком смысле определим разницей статистических показатели качеств, при этом к ней не выдвигаются требования к совпадению внутренних потоков событий. В этом случае модель СУЗ будет адекватной при идентичности реальной системе интервальных статистических показателей, то есть среднее значение и дисперсия измеряемых показателей качества не превышает среднего значения и дисперсии этих показателей в устоявшемся режиме.

Рассмотрим пример. Пусть время ожидания задания в очереди для реальной системы составляет 60 минут со среднеквадратичным отклонением в 10 минут. Тогда можно назвать точной в широком смысле модель СУЗ, которая для того же входного потока событий позволяет получить выходной поток со средним временем ожидания задания в очереди 63 минуты с дисперсией в 5 минут.

2.3.2 Оценка точности натурного эксперимента

Покажем фундаментальную неточность натурного эксперимента. Время обработки прошедшего очередь задания складывается из трех в общем случае случайных величин:

– времени запуска задания, затрачиваемого СУЗ на выделение вычислительных узлов и их конфигурацию в соответствии с требованиями задания;

– времени непосредственного выполнения задания на выделенных ВУ;

– времени завершения задания, затрачиваемом СУЗ на освобождение выделенных ВУ, в т.ч. на контроль завершения всех процессов задания, удаление созданных заданием временных файлов и разделяемых ресурсов и т.п.

Сумма времён запуска и завершения задания является случайной величиной и может зависеть от многих факторов, таких как сетевые задержки, изменение состояния вычислений в ядре операционной системы и т.п.

Рассмотрим экспериментальные данные по измерению накладных расходов, полученные путем многократного запуска задания с известным и неизменным временем выполнения в 60 секунд. Эксперименты производились на

суперкомпьютере МВС-10П ОП [4], раздел Broadwell, функционирующем в МСЦ РАН – Отделении суперкомпьютерных систем и параллельных вычислений НИЦ «Курчатовский институт». Пиковая производительность суперкомпьютера составляет 181 ТФлопс, раздел включает 136 ВУ на базе 16-ядерных процессоров Intel Xeon с микроархитектурой Broadwell, каждый узел включает в себя 2 процессора Intel Xeon E5-2697Av4 и 128 ГБ оперативной памяти.

Каждый запуск задания производился 5 раз, результаты усреднялись, и рассчитывалось среднеквадратическое отклонение. В таблице 1 показана зависимость среднего времени обработки от числа ВУ для задания со временем выполнения 60 секунд.

Из таблицы 1 можно сделать ошибочный вывод, что накладные расходы на запуск и завершение задания существенно растут с числом ВУ.

Таблица 1. Время выполнения задания со временем обработки 60 секунд

Число ВУ	Средние накладные расходы, с	Среднеквадратическое отклонение, с
1	9,7	1,2
2	7,3	2,9
3	10,3	0,6
4	10,7	0,6
5	10,7	1,2
20	15,0	6,2

Детальное рассмотрение серии экспериментов для 20 ВУ показывает, что накладные расходы составили для разных запусков трижды 11 секунд, один раз 17 секунд и один раз 25 секунд. Значение в 25 секунд можно интерпретировать как «выброс», связанный со сложностью одновременного выделения 20 ВУ для задания. С ростом числа ВУ можно говорить о возрастании дисперсии накладных расходов. Изменение времени обработки задания (1 минута, 3 минуты, 10 минут) не оказало влияния на средние накладные расходы, которые сохранились на отметке в 10 секунд. Время запуска задания в эксперименте в среднем составляет 30% от накладных расходов, оставшиеся 70% приходится на завершение задания.

Отметим, что в эксперименте время выполнения задания было искусственно сделано постоянным, тогда как из-за взаимного влияния каналов передачи данных и иных факторов время выполнения одного и того же реального задания при разных запусках может отличаться.

Главным недостатком длительного натурного эксперимента можно назвать его фактическую гипотетичность по причине нецелесообразности его проведения, поскольку дорогостоящие суперкомпьютерные ресурсы в ходе подобного эксперимента будут дублировать уже проведенные расчеты. Если подобный эксперимент и можно будет провести, то только в течение сравнительно небольшого временного промежутка. При этом для получения достоверных статистических данных требуется сбор информации о работе системы в течение дней и недель. На практике натурный эксперимент применяется путем ввода исследуемых параметров СУЗ в эксплуатацию. По изменению показателей качества эксплуатируемой системы принимается решение о сохранении изменений или о возврате к предыдущей версии настроек СУЗ.

Результаты экспериментов показали, что натурный эксперимент не равен самому себе, то есть не воспроизводится со 100% точностью. При функционировании СУЗ имеет место некоторая стохастичность. Рассмотрим существующие способы оценки точности имитационной модели СУЗ.

2.3.3 Существующие способы оценки точности имитационной модели

Можно выделить три способа оценки точности имитационной модели СУЗ, используемые исследователями.

Первый способ сводится к сравнению интегральных показателей качества (таких как загрузка или среднее время ожидания задания в очереди) или анализу характеристик задания (например, время запуска). В работе [9] главной метрикой точности модели является общее время обработки заданий (makespan), и вычисляется разница этой метрики в наборе исследуемых журналов. В работах [11-12] исследуется разница между временами поступления заданий в очередь и временем выполнения заданий в модели и реальной системе. Иногда

исследователи представляют плотность вероятности (density) разницы временных характеристик, например, времени начала обработки задания. Применение плотности вероятности позволяет оценить процент заданий, у которых заданная временная характеристика совпадает для реального и модельного запусков. При этом для точности моделирования важно, чтобы каждое задание сохраняло свои характеристики. Плотность вероятности не позволяет обнаружить изменение в характеристиках отдельных заданий. Расчёт средних значений показателей качества не позволяет отследить характеристики отдельных заданий. Большая разница в характеристиках однозначно свидетельствует о неточности модели, но небольшая разница не позволяет сделать вывод о высокой точности модели. В качества примера возможно выделить две разные недели работы реального суперкомпьютера, для которых среднее время ожидания задания в очереди будет совпадать. Делать вывод об идентичности расписания для этих двух недель нельзя. Недостатком этого способа является низкая детализация, в результате чего две модели с одинаковыми средними значениями будут совершенно по-разному обрабатывать входные потоки заданий.

Второй способ основан на визуальном сравнении графиков показателей качества или гистограмм разниц различных метрик. В [11-12] используются графики сравнения загрузок двух экспериментов. При этом перестановка порядка запуска заданий в модели обнаружена не будет. В работе [13] авторы демонстрируют гистограммы разниц временных характеристик задания двух сравниваемых выходных потоков: времени поступления задания, времени выполнения заданий, времени полного обслуживания задания. Анализ гистограмм проводится визуально, что не даёт возможности сравнить представленные результаты по точности с другими исследованиями.

Третий способ сводится к визуальному анализу планов запуска. Это не позволяет проводить корректное сравнительное исследование различных моделей в идентичных условиях. На представленном в работе [13] фрагменте плана запуска более 200 заданий, и визуально сравнить две диаграммы Гантта для определения точности не представляется возможным.

В отдельных работах игнорируются вопросы валидации модели, основное внимание сфокусировано на вычислительных затратах на моделирования [14] или на предоставлении инструмента для сравнения алгоритмов [8]. В некоторых работах, например, [119] сравнение производится для различных реализаций между собой, и вопрос валидации модели не поднимается. Зачастую исследователи не производят сравнения предлагаемой модели и реальной системы. Результатом их работы является улучшение качества модели СУЗ, и предполагается, что качество реальной СУЗ улучшится аналогичным образом.

Рассмотренные способы не предоставляют способа численного показателя точности модели. Актуальной является задача формирования численного показателя для оценки разницы между двумя экспериментами при фиксированном входном потоке. Этот показатель позволит сравнивать модели по точности.

2.4. Способ оценки точности имитационной модели системы управления заданиями

Сформируем показатель точности имитационной модели СУЗ на основе предлагаемого в работе [120] способа оценки достоверности модели – валидации событий (event validity), когда сравниваются потоки событий моделируемой и реальной систем. В указанной работе не приводится численных показателей, позволяющих сравнить два потока событий. Определим численный показатель близости выходных потоков заданий, на основе которого будем сравнивать точность имитационных моделей СУЗ.

Рассмотрим два выходных потока заданий двух разных имитационных моделей в результате обработки одного и того же входного потока заданий. Выходные потоки заданий представлены заданиями $j = (j_1, j_2, \ldots, j_n)$ и $J = (J_1, J_2, \ldots, J_N)$. Характеристики задания $j_i = r_i$ (время поступления задания в очередь), p_i (требуемый объем ресурсов), e_i (требуемое время обработки), w_i (реальное время обработки), s_i (время запуска задания). Аналогичными характеристиками обладает задание $J_i = R_i, P_i, E_i, W_i, S_i$.

Сформулируем критерий недостоверности имитационной модели. Модель является **недостоверной**, если число заданий при моделировании не совпало с числом заданий в реальной системе. Если какие-либо из заданий не были воспроизведены при моделировании, или возникли новые задания – то модель недостоверна. Будем также считать модель недостоверной при несовпадении числа заданий $n \neq N$, времени поступления задания в очередь в модели и реальной системе $r_i \neq R_i$, при несовпадении заказанного времени выполнения задания $p_i \neq P_i$ или заказанных вычислительных ресурсов $e_i \neq E_i$. Таким образом, модель является недостоверной, если $n \neq N$, $r_i \neq R_i$, $p_i \neq P_i$, или $e_i \neq E_i$. Для недостоверной модели показатель близости выходных потоков заданий будет не определён.

Для достоверной модели характеристики задания можно переписать следующим образом: $J_i = r_i, p_i, e_i, W_i, S_i$.

Для полностью достоверной модели в эксперименте и в реальной системе совпадают число, порядок и время поступления всех заданий. На практике построение полностью достоверной имитационной модели СУЗ фактически невозможно даже для натурного эксперимента, как было показано в разделе 2.3.2.

Назовем диапазон между недостоверной и полностью достоверной моделями линейкой (шкалой) достоверности. Очевидно, все исследуемые модели СУЗ будут располагаться на этой линейке. Введём показатель близости выходных потоков заданий (далее показатель близости), который будет служить для численной оценки разности результатов моделирования одного входного потока заданий на разных моделях. Показатель близости позволит определять место имитационной модели на линейке достоверности.

При исследовании точности в узком смысле возможно использовать математический аппарат проверки статистических гипотез. В этом случае выдвигается статистическая гипотеза – предположение, что события входного и выходного потоков для некоторого уровня значимости принадлежат одному и тому же распределению случайной величины. Модель является точной в узком смысле, если события входного и выходного потоков являются случайными

величинами, принадлежащими распределению одной и той же случайной величины.

Для исследования точности в узком смысле необходимо разработать такое описание результата моделирования, которое позволит упростить анализ. Подходящим результирующим описанием процесса моделирования может быть вектор значений, построенный как некоторый «слепок» всех характеристик события. Рассмотрим предлагаемый автором вариант перехода от множества событий с различными характеристиками к одному вектору значений, позволяющему описать результат моделирования, то есть выходной поток заданий.

Рисунок 6 иллюстрирует пример сравнения точности работы разных моделей. Один и тот же входной поток заданий обрабатывается на двух имитационных моделях СУЗ, причём в качестве модели может быть использована реальная система. Выходные потоки заданий анализируются с целью определения точности моделирования.

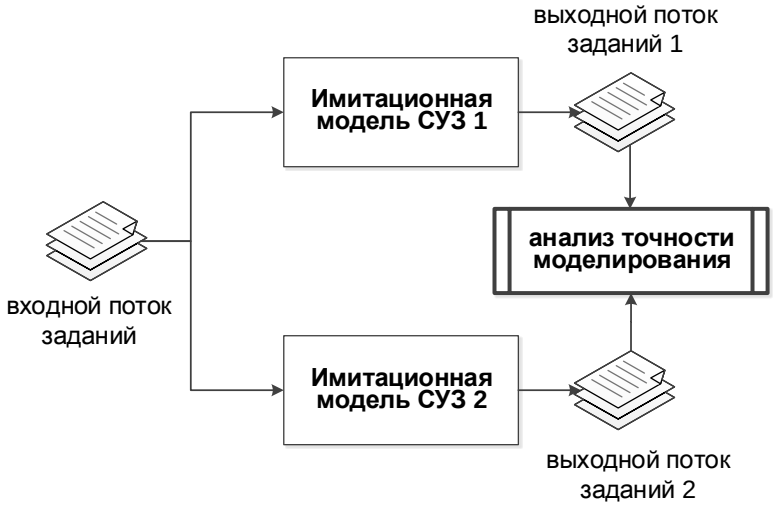

Рисунок 6 – Сравнение точности разных моделей

Построим два вектора размерности $n = N$. Для входного потока j определим вектор времен пребывания заданий в системе $v = (v_1, v_2, \ldots, v_n)$, $i \in \overline{1,n}$, где каждая компонента соответствует номеру задания в порядке его поступления в систему. Значение компоненты $v_i = (s_i - r_i + w_i)$ задаётся как время пребывания задания в системе, то есть сумма времени ожидания задания в очереди и времени его

обработки. Для потока *J* аналогично определим *вектор* $V = (V_1, V_2, \ldots, V_n)$, $V_i = (S_i - R_i + W_i)$, $i \in \overline{1,n}$.

Таким образом, мы получили два вектора – *v* и *V*, различие между компонентами которых фактически определяет различие между двумя моделями СУЗ. Естественным показателем разности двух n-мерных векторов является евклидово расстояние между ними:

$$E = \sqrt{\sum_{i=1}^{n}(V_i - v_i)^2} \qquad (11)$$

Проверим независимость евклидова расстояния от длительности модельного эксперимента, измеряемого в числе обработанных заданий. Для этого сформируем поток из 1000 заданий, взятый из статистики работы раздела Broadwell суперкомпьютера МВС-10ПОП в МСЦ РАН – Отделении суперкомпьютерных систем и параллельных вычислений НИЦ «Курчатовский институт».

Сформируем вектор *v* потока *j* реальной системы, и на его основе сгенерируем вектор *V* модельного потока *J*. При генерации вектора *V* с вероятностью 50% внесём изменения в 100 секунд в каждую компоненту вектора, т.е. с вероятностью 0,5 значение времени обработки v_i *i*-го задания меняется на 100 секунд. При постоянных частоте изменений и средней величине изменения показатель разности потоков не должен изменяться. При выполнении этого требования показатель разности для короткого эксперимента будет совпадать с показателем разности для длительного эксперимента. Так как показатель *E* представляет собой сумму положительных чисел, то с ростом *N* показатель *E* нарастает. Как показано в таблице 2, показатель (11) зависит от продолжительности процесса моделирования, что делает неприменимым евклидово расстояние в качестве показатель близости выходных потоков заданий. Проведем нормализацию показателя (11) и получим показатель разности *P* потоков j и J:

$$P = \sqrt{\frac{\sum_{i=1}^{n}(V_i - v_i)^2}{n}} \qquad (12)$$

Эту величину и определим, как показатель близости выходных потоков заданий.

Как показывает таблица 2, показатель (12), в отличие от евклидова расстояния (11), не зависит от длительности модельного эксперимента при заданных величине и вероятности изменений. Для каждого набора показателей рассчитаем коэффициент вариации как отношение стандартного отклонения к среднему значению. Чем ближе коэффициент вариации к нулю, тем более однородное значение показателя, что в нашем случае говорит о корректном отслеживании одинаковой частоты изменения сравниваемых векторов и независимости измеряемого показателя от длины эксперимента при одинаковой частоте ошибки.

Таблица 2. Сравнение показателей (11) и (12) для выходных потоков заданий в эксперименте с изменениями в 1 секунду для 50% заданий

Число заданий (размер векторов)	Показатель близости P в соответствии с (12)	Показатель разности E в соответствии с (11)	Число отличающихся заданий (компонентов сравниваемых векторов)
50	71	500	25
100	77	780	60
250	74	1170	138
500	71	1590	253
750	72	1980	392
1000	69	2190	481
Коэффициент вариации:	0,039	0,49	

Теперь при генерации модельного вектора *V* внесём изменение в 1000 секунд в каждую компоненту с вероятностью 50%. Результаты представлены в таблице 3, коэффициент вариации для меры разности равен 0,014.

Таблица 3. Показатель близости выходных потоков заданий для эксперимента с изменениями в 1000 секунд для 50% заданий

Число заданий (размер сравниваемых векторов)	Показатель близости P в соответствии с (12)	Число отличающихся заданий (компонентов сравниваемых векторов)
100	721,1	52
250	726,6	132
500	702,9	247
750	705,2	373
1000	712,7	508

Из таблиц 2 и 3 можно сделать вывод, что при сохранении средней величины изменения и частоты изменений показатель близости (12) не растёт с увеличением числа заданий в сравниваемых экспериментах. Этот факт позволяет использовать показатель близости (12) для любых по длительности модельных экспериментов в качестве показателя близости выходных потоков заданий, на основе мы сможем сравнивать точность имитационных моделей. Чем меньше значение показателя близости, тем точнее модель воспроизводит исследуемый входной поток заданий.

На основе предложенного показателя близости можно разработать метод имитационного моделирования.

2.5. Метод имитационного моделирования системы управления заданиями

Рисунок 7 представляет существующую схему применения имитационной модели для оценки её точности. Входной поток заданий обрабатывается в исследуемой СУЗ, в результате образуется выходной поток заданий. Этот выходной поток заданий направляется на модель СУЗ для обработки, результат работы модели СУЗ сравнивается с выходным потоком заданий с расчётом точности моделирования.

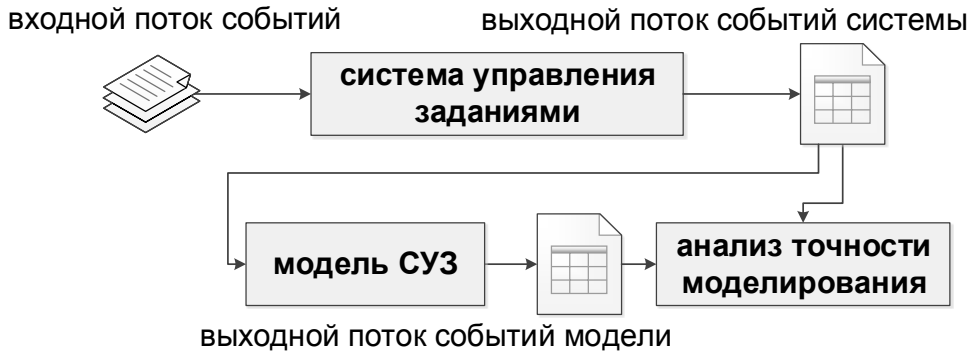

Рисунок 7. Существующая схема применения имитационной модели

Существующая схема не удовлетворяет требованиям, изложенным в 1 главе. Рисунок 8 иллюстрирует представление предлагаемого метода имитационного моделирования в виде схемы.

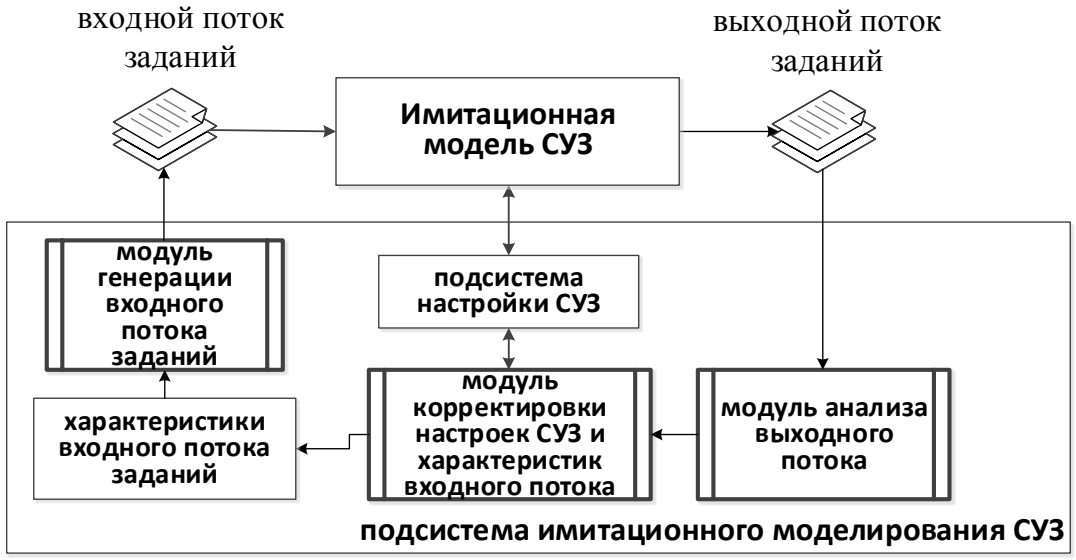

Рисунок 8 – Общая схема имитационного моделирования СУЗ

Рассмотрим метод имитационного моделирования системы управления заданиями, который позволяет улучшать их характеристики для заданного входного потока заданий с количественной оценкой точности используемых моделей. Предлагаемый метод состоит из следующих этапов.

1. Выбор схемы имитационного моделирования.

2. Выбор цели имитационного моделирования. Целью может быть поиск наилучших настроек СУЗ (таких как алгоритм планирования и его настройки),

исследование точности имитационной модели или прогнозирование времени выполнения заданий.

3. Выбор имитационной модели СУЗ для проведения моделирования. В качестве имитационной модели может быть использована реальная СУЗ.

4. Получение выходного потока заданий из реальной СУЗ и его преобразование во входной поток заданий, или генерация входного потока заданий на основе выбранных параметров и видов распределения вероятностей характеристик заданий.

5. Внесение, при необходимости, изменений в настройки модели СУЗ.

6. Проведение имитационного моделирования с входным потоком заданий, получение выходного потока заданий эксперимента.

7. Ручной или автоматизированный анализ выходного потока заданий. В зависимости от выбранной в п. 1 цели рассчитать интересующие показатели качества обработки потока заданий или количественный показатель близости модели относительно входного потока заданий из реальной системы или из предыдущей итерации эксперимента.

8. Если результаты анализа свидетельствуют о достижении цели (получены подходящие настройки СУЗ с точки зрения показателей качества обработки потока заданий или достигнута требуемая точность моделирования), то процесс моделирования завершается. Если нет, то переход к этапу 2.

На рисунке 8 виден циклический характер исследования. Многократный эксперимент позволяет исследовать различные вариации входных потоков и параметров СУЗ, т.е. определить влияние множества интересующих факторов на качество обработки потока заданий.

Наличие в этапе 3 возможности генерации входного потока заданий позволяет исследовать произвольную СУЗ в условиях различных входных потоков заданий.

Имитационная модель неотличима от реальной системы, если для фиксированного входного потока заданий значение показателя близости модели меньше или равно значению показателя близости для натурного эксперимента.

Значение показателя близости для натурного эксперимента можно считать эталонным значением показателя близости для фиксированного входного потока заданий.

Как показано в разделе 2.3.2, невозможно провести два идентичных натурных эксперимента, поэтому предлагается определять эталонное значение показателя близости путем сравнения результатов моделирования СУЗ с виртуальным вычислителем, как наиболее точного варианта моделирования.

Рассмотрим проведённый эксперимент. На основе обработки статистики суперкомпьютера МВС-10П ОП, установленного в Межведомственном суперкомпьютерном центре РАН (МСЦ РАН, ныне – отделении МСЦ Курчатовского высокопроизводительного вычислительного комплекса НИЦ «Курчатовский институт»), сформируем модельный поток из 1000 заданий, с которым было проведено моделирование отечественной СУППЗ с виртуальным вычислителем. Путем сравнения результатов моделирования с реальной системой были определены значения показателя близости модели СУППЗ с виртуальным вычислителем для разного числа заданий.

Результаты представлены в таблице 4. Столбец «число заданий» соответствует числу k первых заданий потока j (реальная СУППЗ) и потока J (СУППЗ с виртуальным вычислителем). В столбце «число отличающихся заданий» указано число заданий, для которых времена ожидания в очереди в потоках j и J отличались. Из таблицы 4 можно сделать вывод, что значение показателя близости модели СУЗ, рассчитанной по формуле (12), равняется 12.

Таблица 4. Показатель близости для модели СУППЗ с виртуальным вычислителем

Число заданий	Показатель близости в соответствии с (12)	Число отличающихся заданий
50	0	0
100	12,0	4
250	11,4	13
500	12,0	20
750	11,6	28
1000	11,2	35

Рассмотрим варианты исследования поведения СУЗ в условиях различных входных потоков.

2.6. Экспериментальное исследование показателя близости на примере имитационных моделей СУЗ с заведомо разной точностью

Эксперименты приводились для входного потока из раздела Broadwell суперкомпьютера МВС-10ПОП в МСЦ РАН – Отделении суперкомпьютерных систем и параллельных вычислений НИЦ «Курчатовский институт» за недельный период 27.03.2019-03.04.2019. Раздел находится под управлением СУППЗ. Во время профилактики задания не выполняются и очередь заблокирована. При этом у пользователей и заданий имеются приоритеты, информацию о которой назовём оперативной информацией (ОИ). После окончания профилактики задания поступают на вычислитель в порядке приоритетов.

Рисунок 9 иллюстрирует общую схему проведения эксперимента. Из статистики СУППЗ извлекаются характеристики заданий за исследуемый период. Генератор входного потока на основе этих характеристик формирует входной поток заданий, при необходимости модифицируя характеристики (например, планируемое время выполнения). Этот поток подаётся на вход симулятора СУППЗ.

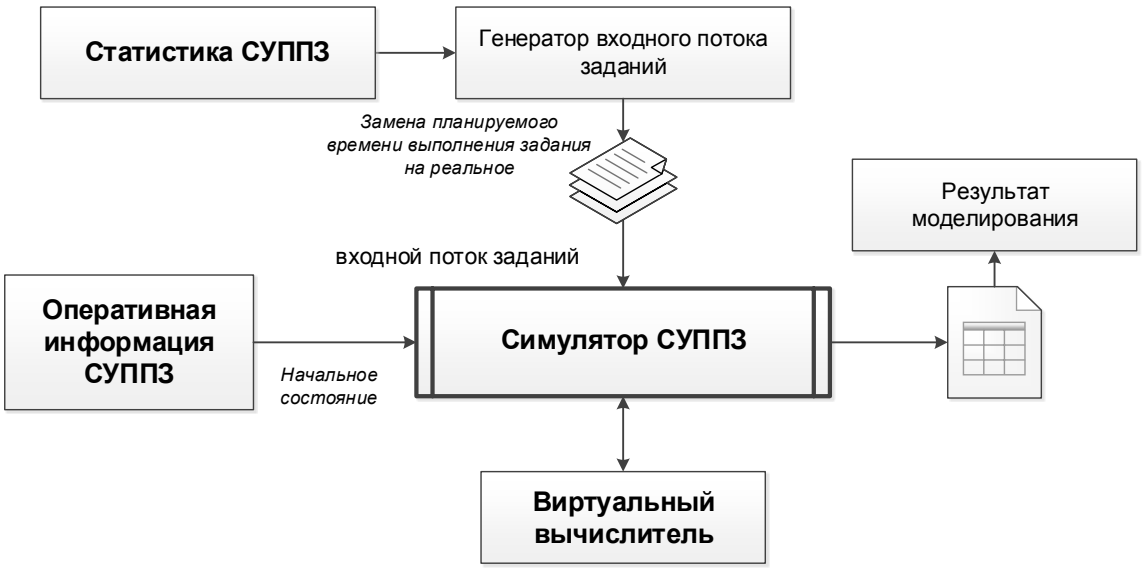

Рисунок 9 – Эксперимент с симулятором СУППЗ

Начальное состояние симулятора СУППЗ может учитывать или не учитывать оперативную информацию о приоритетах пользователей. Входной поток заданий обрабатывается на симуляторе СУППЗ с виртуальным вычислителем, в результате чего образуется выходной поток заданий для дальнейшего анализа.

Рассмотрим результаты применения разработанного метода на одном и том же входном потоке заданий, для последовательности моделей СУЗ с заведомо разной точностью:

– вариант а) – моделирование с виртуальным вычислителем с учетом оперативной информации, самый точный вариант моделирования, наиболее близкий к натурному эксперименту;

– вариант б) – моделирование с виртуальным вычислителем без учета оперативной информации, менее точный вариант моделирования, поскольку не учитываются приоритеты пользователей;

– вариант в) – моделирование с виртуальным вычислителем, без учета оперативной информации, дополнительно снижена точность моделирования за счёт замены во входном потоке заданий планируемого времени выполнения на реальное время выполнения, т.е. пользователь без ошибок указывает планируемое время, в результате чего планировщик по-другому составляет расписание запусков заданий;

– вариант г) – моделирование в Alea с алгоритмом обратного заполнения: ещё менее точная модель, поскольку в ней применяется другая реализация алгоритма планирования.

В таблице приведено сравнение интегральных характеристик СУЗ. Загрузка ресурсов находится на одном уровне, время ожидания заданий в очереди для эксперимента с оперативной информацией отличается несущественно. Показатель близости имеет порядок десятков тысяч.

Таблица 5. Результаты моделирования по точности

Показатель\Система	Реальная система	Вариант а) с учётом ОИ	Вариант б) с учётом ОИ и с модификацией	Вариант в) без учёта ОИ	Вариант г) Alea
Полная загрузка в соответствии с (5), %	85,7	87,4	87,7	85,8	84,8
Среднее приведённое время ожидания в очереди относительно реального времени обработки в соответствии с (8)	227,0	207,1	168,9	184,3	337,7
Среднее время ожидания в очереди в соответствии с (7), с	52 178	54 070	40 148	25 017	44 057
Средняя длина очереди	19,1	17,0	6,5	7,1	12,7
Показатель близости в соответствии с (12), 10^3	0	48,5	68,0	71,7	116,6

Если в таблице смотреть на интегральные показатели качества, например, загрузку вычислительных ресурсов, то эксперимент без учёта ОИ выглядит более точным. При этом в этом эксперименте среднее время ожидания задания в очереди отличается в два раза, что не позволяет говорить о высокой точности модели. Анализ одновременно различных показателей качества затруднителен. Из таблицы 5 видно, что наименьшее значение показателя близости соответствует эксперименту с наиболее точными интегральными показателями качества.

Рисунок 10 иллюстрирует графики полезной загрузки ресурсов для реальной системы (красный), модели с оперативной информацией (синий), модели без оперативной информации (зелёный), без оперативной информацией и модификацией заказанного времени (оранжевый). Визуально полезная загрузка у всех моделей схожа с реальной системой, но каждая модель отличается от реальной системы. На базе интегральных характеристик и графиков сравнения отдельных характеристик не представляется возможным численно сравнить модели и определить, какая из них точнее воспроизводит реальную систему.

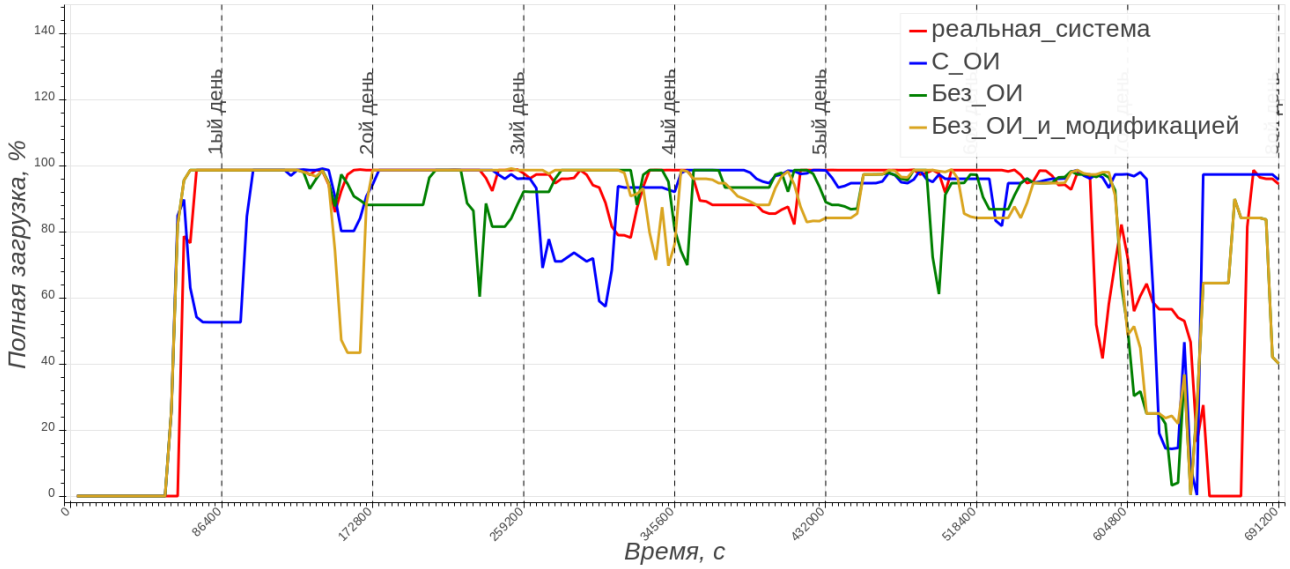

Рисунок 10 – Сравнение исходного журнала и эксперимента с оперативной информацией: загрузка

Рисунок 11 отражает результат ранжирования моделей по точности. Наиболее точная модель – симулятор СУППЗ с виртуальным вычислителем и воспроизведением оперативной информации. Следующая по точности модель без оперативной информации, что неизбежно приводит к уменьшению точности моделирования. Синяя – эксперимент, в ходе которого во входной поток вносится модификация. Важно, что показатель близости в этом случае возрастает, то есть снижается точность моделирования.

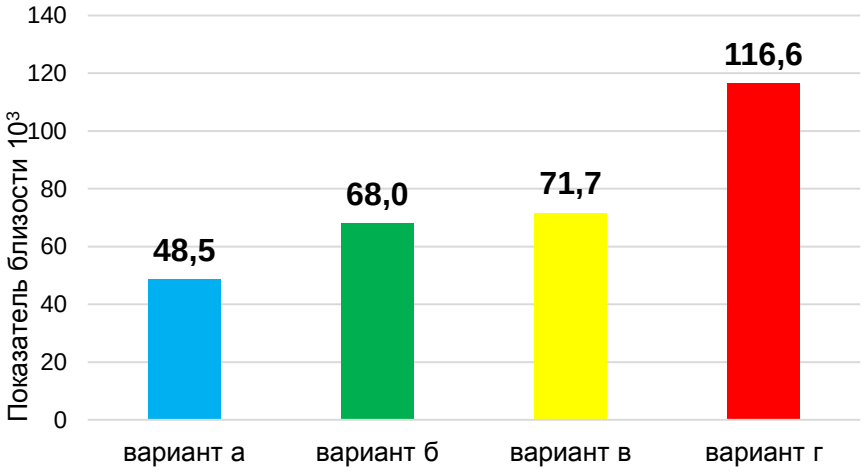

Рисунок 11 – Показатель близости выходных потоков для имитационных моделей с заведомо разной точностью

Наименее точным является эксперимент с Alea. Несмотря на схожесть алгоритмов, результаты эксперимента говорят о большей разнице между результатом работы Alea и исходным журналом, чем при использовании СУППЗ с виртуальным вычислителем.

С тем же входным потоком проведена серия экспериментов, включающая расчёт показателя близости для 8 различных алгоритмов, реализованных в Alea: FCFS, Easy Backfilling, Aggressive Backfilling, Early Deadline First, PBS_PRO, Best-Gap Backfill, FairShare Easy Backfilling, Shortest Job Policy. Рисунок 12 демонстрирует показатель близости для каждого из алгоритмов в сравнении с экспериментом с оперативной информацией с предыдущего рисунка. Для исследуемого потока заданий различные модификации алгоритма обратного заполнения не оказывают существенного влияния на показатель близости. Алгоритм Early Deadline First показывает наибольшее отличие по показателю близости от эксперимента на симуляторе СУППЗ с оперативной информацией. Алгоритм Best-Gap Backfill использовался в предыдущей серии экспериментов.

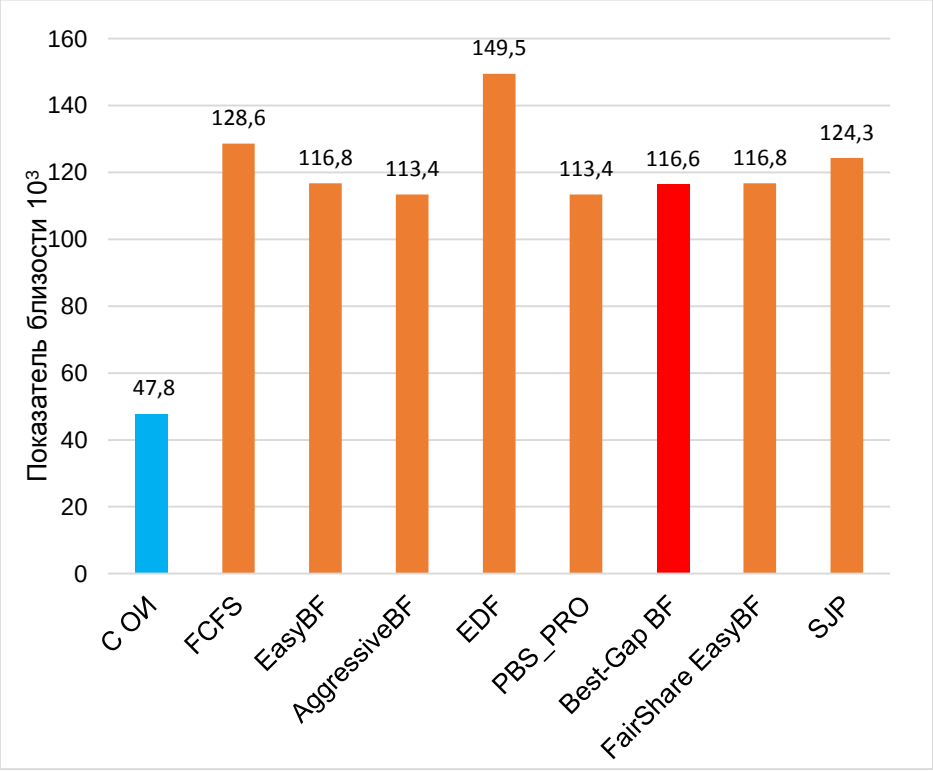

Рисунок 12 – Сравнение различных алгоритмов Alea и симулятора СУППЗ с оперативной информацией по значению показателя близости

Таблица 6 содержит показатели качества для исходного журнала, эксперимента СУППЗ с оперативной информацией и трёх алгоритмов Alea: Best-Gap Backfill, Early Deadline First, FSFC. Несмотря на совпадение полной загрузки с исходным журналом, другие характеристики существенно отличаются.

Таблица 6. Результаты моделирования алгоритмов в Alea по сравнению с экспериментом с оперативной информацией

Показатель\Система	Исходный журнал	Эксперимент с ОИ	Best-Gap Backfill	Early Deadline First	FSFC
Полная загрузка в соответствии с (5), %	84,8	87,4	84,8	84,8	84,8
Среднее приведённое время ожидания в очереди относительно реального времени обработки в соответствии с (8)	262,0	207,1	337,7	151,7	955,8
Среднее время ожидания в очереди в соответствии с (7), с.	57 529	54 070	44 057	55 456	88 298
Средняя длина очереди	19,1	17,0	12,7	16,0	25,4
Показатель близости в соответствии с (12), 10^3		47,8	116,6	149,5	128,6

Результаты экспериментов подтвердили интуитивные представления о точности моделирования – чем ниже место модели в списке, тем меньше точность. Из этого мы можем сделать вывод о том, что нормированное евклидово расстояние между векторами времен пребывания заданий в системе действительно может применяться на практике в качестве количественного показателя точности модели СУЗ.

Недостатком эксперимента для СУППЗ с виртуальным вычислителем является длительное время его проведения. Этот недостаток возможно преодолеть за счёт применение продвижения системного времени в те моменты времени, когда событий с точки зрения СУППЗ не происходит. Реализация такого механизма позволяет проводить недельный эксперимент в течение суток. При этом применение этого механизма может оказать негативное влияние на точность моделирования.

Сравним результаты работы двух два экспериментов без оперативной информации на симуляторе СУППЗ с ВВ с продвижением времени и в реальном времени. Рисунок 13 содержит графики полной загрузки (вверху) и числа заданий в очереди (внизу) для эксперимента (красный – в реальном времени, синий – с продвижением времени).

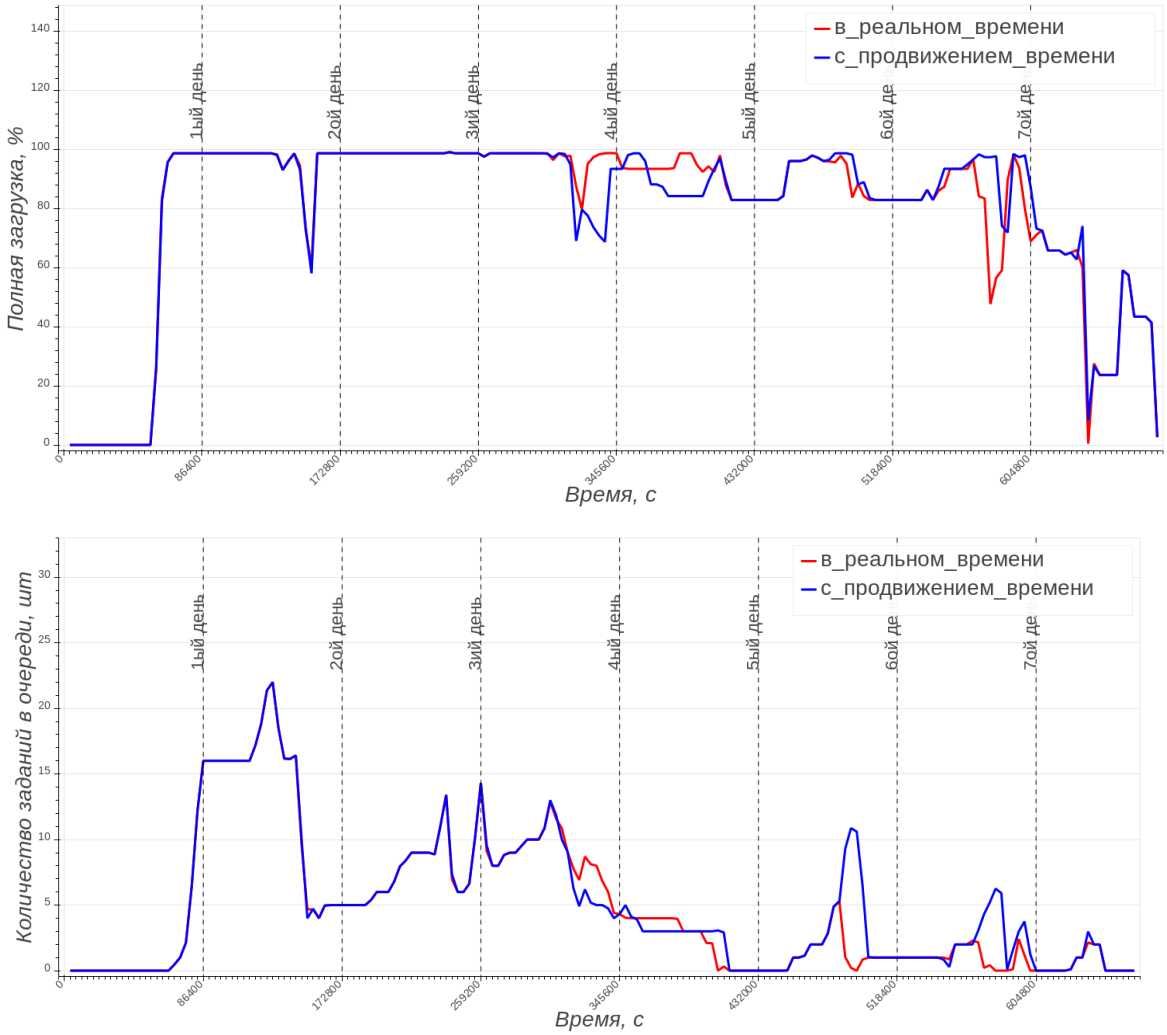

Рисунок 13 – Сравнение экспериментов с реальным временем и продвижением времени: полная загрузка и количество заданий в очереди

Видно, что на 4 день появляются отличия для эксперимента с продвижением времени. При этом по таблице 7 можно сделать вывод, что режим продвижения времени вносит несущественный вклад в снижение точности моделирования.

Таблица 7. Результаты влияния продвижения времени на показатели качества обработки потока заданий

Показатель\Система	Исходный журнал	Эксперимент в реальном времени	Эксперимент с продвижением времени
Полная загрузка в соответствии с (5), %	85,7	85,8	85,7
Среднее приведённое время ожидания в очереди относительно реального времени обработки в соответствии с (8)	227,0	184,3	203,6
Среднее время ожидания в очереди в соответствии с (7), с.	52 178	25 017	26 024,9
Средняя длина очереди	19,1	7,1	7,4
Показатель близости в соответствии с (12), 103	0	68,0	73,3

Выводы по главе 2

Проведённый анализ методов и средств моделирования СУЗ показал, что наиболее распространённым методом моделирования показателей качества обработки потока заданий является имитационное моделирование. Имитационная модель может быть построена на базе СУЗ с виртуальным вычислителем или быть реализована в виде модели-симулятора.

Показано, что вследствие случайных задержек при запуске и завершении заданий в СУЗ многократное повторение натурного эксперимента с реальной системой в одних и тех же условиях приводит к отличающимся выходным потокам. Сделан вывод, что обеспечить полное соответствие результатов работы имитационной модели и реальной системы не представляется возможным. Для оценки точности той или иной модели СУЗ может проводиться сравнение с реальной системой. Предложены понятия точности модели СУЗ в узком и широком смыслах. Точность в узлом смысле определяется степенью схожести выходных потоков заданий двух сравниваемых моделей на одном и том же входном потоке заданий. Точность в широком смысле определяется разницей

статистических показателей качества, при этом не выдвигаются требования к совпадению выходных потоков заданий.

Для определения точности модели в узком смысле предложено численное определение показателя близости выходных потоков заданий модели СУЗ. Все задания входного потока сводятся к единственному вектору, в котором каждая компонента соответствует определенному заданию и содержит время пребывания этого задания в системе. Показано, что разные натурные эксперименты с одинаковыми условиями на одном и том же потоке заданий имеют ненулевое значение показателя близости выходных потоков заданий.

Разработан метод имитационного моделирования систем управления заданиями, позволяющий оценивать точность исследуемых моделей при помощи количественного показателя близости выходных потоков заданий и улучшать характеристики систем управления заданиями для заданного входного потока заданий в ходе итеративного процесса настройки параметров модели

Для подтверждения корректности разработанного метода была сформирована последовательность из 5 различных моделей СУЗ, в которой каждая следующая модель намеренно огрублялась по сравнению с предыдущей. Результаты моделирования показали, что разработанный метод адекватно ранжирует модели по точности.

3. Методика формирования пакетов заданий по типам

В главе изложено предлагаемое решение задачи повышения качества обработки потока заданий с длительным временем инициализации. Предложена частная модель системы обработки заданий с поддержкой типов (раздел 3.1), разработана методика формирования пакетов заданий по типам (раздел 3.2). Для внедрения предлагаемой методики в систему управления заданиями разработана архитектура программного комплекса имитационного моделирования с поддержкой формирования пакетов заданий по типам (раздел 3.3).

3.1. Обработка заданий с высокой долей накладных расходов в многопроцессорных вычислительных системах

3.1.1 Частная модель системы обработки линейно масштабируемых заданий с поддержкой типов

Методы пакетирования, рассмотренные в разделе 1.5, функционируют для системы, в которой все задания одного типа. На практике существуют различные типы заданий, которые объединять в один пакет технически не представляется возможным – процедура инициализации у разных типов заданий отличается.

Рисунок 14 иллюстрирует частную модель системы обработки потока линейно масштабируемых заданий с поддержкой типов заданий.

Для каждого задания J_i, где $i \in \overline{1,n}$ модели системы обработки потока заданий в СУЗ, изложенной в разделе 1.2, определяется тип задания $type_i$, то есть способ его обработки на ВУ. Для заданий одного типа $type_i = j$ создаётся очередь заданий X_j со следующими характеристиками:

1. Длительность накладных расходов на обработку задания (т.е. запуска и завершения задания) $S_j = a_i + b_i + d_i$, значения a_i, b_i, d_i являются одинаковыми для всех заданий одного типа.
2. Приоритет PR_j.
3. Рекомендуемое максимальное время ожидания в очереди T_j^{max}, которое можно использовать при определении порядка формирования пакетов.

4. Программный модуль инициализации вычислительных ресурсов и обработки пакета заданий.

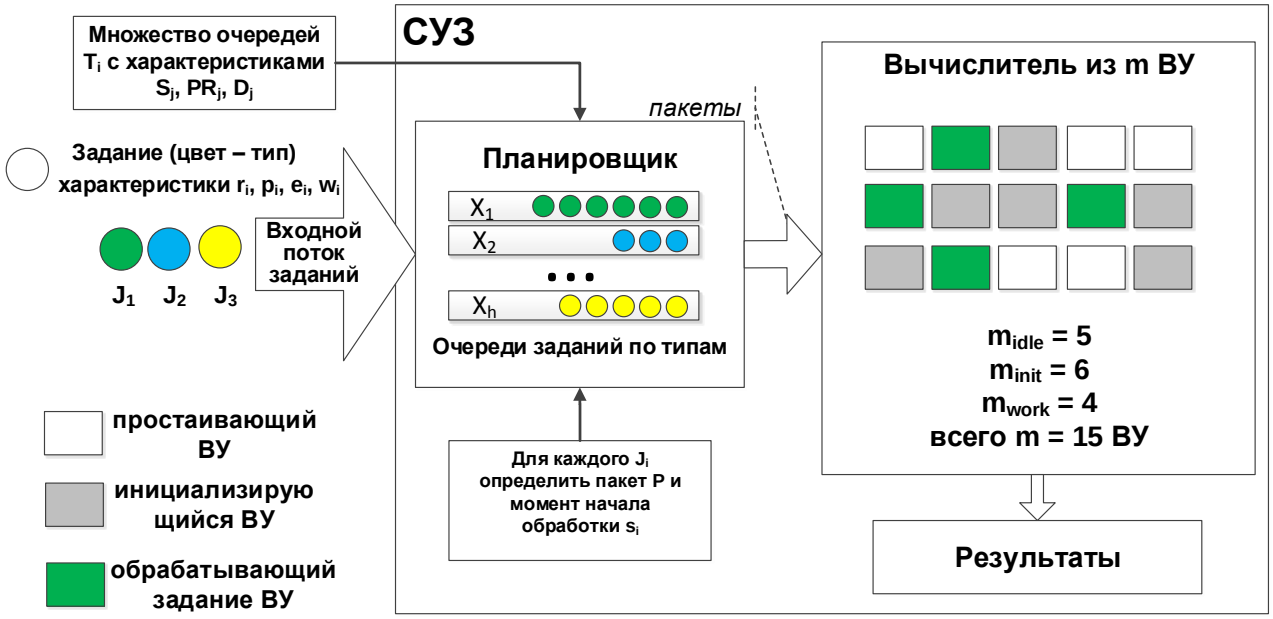

Рисунок 14 – Модель системы обработки потока заданий по типам в СУЗ

Приоритет задания не учитывается при формировании пакета заданий (причины изложены в разделе 1.4), но может учитывается для определения состава пакета (см. раздел 3.2.4).

Задания одного типа допускается выполнять одно за другим без повторной инициализации вычислительных ресурсов. Запущенное таким образом множество заданий будем называть пакетом заданий P. Для выбранного множества однотипных заданий $J_{i_1}, J_{i_2} \dots, J_{i_k}$, где $k \in \overline{1,n}$ выполняется $type_{i_1} = type_{i_2} = \cdots = type_{i_k} = j$.

Время запуска и завершения заданий одного типа совпадает.

$$a_{i_1} = a_{i_2} = \cdots = a_{i_k} = a_j \quad (13)$$

$$c_{i_1} = c_{i_2} = \cdots = c_{i_k} = c_{ji_k} \quad (14)$$

$$a_j + c_j = S_j \quad (15)$$

Будем обозначать i-ый сформированный пакет из k заданий типа j как $P_i = \{J_{i_1}, J_{i_2} \dots, J_{i_k}\}$. Планировщику для пакета заданий необходимо определить:

– множество заданий $J_{i_1}, J_{i_2} \dots, J_{i_k}$, из которых формируется пакет, при этом допустим пакет из одного задания;

- требуемое количество вычислительных узлов p_i;
- заказанное время счёта пакета $e_i = S_j + b_{i_1} \ldots + b_{i_k}$;
- момент времени начала обслуживания пакета s_i.

При этом все пакеты заданий должны быть полностью обеспечены ресурсами, то есть должно выполняться $\sum_{i=1}^{n} p_i \varphi_i(t) \leq m$, где $\varphi_i(t) = 1$, если пакет i выполняется в момент времени t, и $\varphi_i(t) = 0$ в противном случае.

Задания можно разделить на масштабируемые и немасштабируемые [69]. Немасштабируемые задания могут быть запущены только на указанном пользователем объёме вычислительных ресурсов. Масштабируемые задания могут быть запущены на различном объёме вычислительных ресурсов.

Для масштабируемых заданий пользователь явно задаёт допустимое для задания количество вычислительных ресурсов и соответствующее этому количеству время выполнения. Например, может быть указано, что для выполнения задания требуется 1 ВУ и 30 минут, или 2 ВУ и 25 минут, или 4 ВУ и 20 минут. Может быть задана аналитическая зависимость времени выполнения задания от количества выделенных заданию вычислительных ресурсов в виде формулы или таблицы.

Линейно масштабируемые задания – это такое подмножество масштабируемых заданий, для которых имеет место линейная зависимость времени выполнения от количества вычислительных ресурсов. Такие задания линейно масштабируются на вычислителе, т.е. увеличение вдвое числа задействованных ВУ уменьшает вдвое время обработки. Например, время выполнения на 1 ВУ составляет 30 минут, тогда на 2 ВУ 15 минут, на 30 ВУ 1 минуту. Зачастую линейно масштабируются задания с параллелизмом по данным.

К линейно масштабируемым заданиям также отнесём задания с определённым временем инициализации. Время инициализации в этом случае не масштабируется и является константой для задания. Время выполнения масштабируется линейно. Так, если время инициализации составляет 10 минут, время обработки 20 минут (суммарное время выполнения 30 минут) на 1 ВУ, то

на 2 ВУ время выполнения составит 10 + (20/2) = 20 минут, на 30 ВУ составит 10 + (20/30) = 13,3 минуты.

Будем рассматривать линейно масштабируемые задания с постоянным временем инициализации.

3.1.2 Расчёт минимальных накладных расходов на обработку для линейно масштабируемых заданий

При проведении вычислительного эксперимента будем рассматривать самую неблагоприятную для пакетирования ситуацию – минимальные накладные расходы на инициализацию. Определим, при каких условиях накладные расходы будут минимальными для линейно масштабируемых заданий.

Для сформированного пакета P из множества однотипных, линейно масштабируемых заданий $J_{i_1}, J_{i_2} \dots, J_{i_k}$, где $k \in \overline{1,n}$ время обработки на m вычислительных узлах составляет

$$T_P = S_j + \frac{b_{i_1} \dots + b_{i_k}}{m} \tag{16}$$

Отношение накладных расходов на обработку этого пакета ко времени обработки пакета равно

$$Z_p = \frac{\mathrm{s_j}}{\frac{b_{i_1} \dots + b_{i_k}}{\mathrm{m}}} = \frac{\mathrm{s_j} \times m}{b_{i_1} \dots + b_{i_k}} \tag{17}$$

Из формулы (17) следует, что для фиксированного времени инициализации и фиксированном времени обработки заданий с ростом числа вычислительных узлов m увеличиваются накладные расходы на инициализацию. Накладные расходы будут наименьшими при наименьшем количестве вычислительных узлов, обрабатывающих пакет заданий, то есть при обработке на 1 ВУ. Этот вывод будет использован в разделе 3.5 при формировании входного потока заданий для экспериментальных исследований.

3.1.3 Расчёт времени выполнения линейно масштабируемого задания на одном вычислительном узле

Рассмотрим, как для линейно масштабируемых заданий возможно рассчитать время выполнения на 1 ВУ для задания типа j для заданных количества вычислительных узлов p_i, заказанного временем выполнения e_i и длительности накладных расходов на обработку задания S_j.

Для задания J_i типа j константой является длительность накладных расходов на обработку задания S_j. Время обработки b_i линейно масштабируемых заданий линейно зависит от используемого количества вычислительных узлов p_i.

$$b_{\text{на 1 ВУ}} = b_{\text{на } p_i \text{ ВУ}} \times p_i \tag{18}$$

Добавим в обе части выражения S_j.

$$b_{\text{на 1 ВУ}} + S_j = b_{\text{на } p_i \text{ ВУ}} \times p_i + S_j \tag{19}$$

Преобразуем выражение, учитывая, что реальное время выполнения состоит из накладных расходов и обработки

$$w_i = S_j + b_i \tag{20}$$

$$w_{\text{на 1 ВУ}} = S_j + b_{\text{на 1 ВУ}} \tag{21}$$

$$w_{\text{на } p_i \text{ ВУ}} = S_j + b_{\text{на } p_i \text{ ВУ}} \tag{22}$$

В результате получим формулу расчёта реального времени выполнения на одном вычислительном узле для заданного реального времени выполнения на p_i вычислительных узлах.

$$w_{\text{на 1 ВУ}} = \left(w_i - S_j\right) \times p_i + S_j \tag{23}$$

Таким образом, формула (23) позволяет вычислить реальное время выполнения на одном вычислительном узле произвольного линейно масштабируемого задания для заданного времени его выполнения на p_i вычислительных узлах. Аналогичным образом возможно рассчитать заказанное время счёта задания на одном вычислительном узлах.

3.2. Методика формирования пакетов заданий по типам

3.2.1 Этапы методики формирования пакета заданий по типам

Опишем разработанную автором методику формирования пакетов заданий по типам (далее – методику ФПЗТ). Методика ФПЗТ позволяет для непрерывно поступающего в систему множества заданий различных типов определять момент запуска, состав пакета заданий и требуемое количество вычислительных узлов для его обработки (согласно требованиям, изложенным в разделе 1.7).

Пусть производится обработка заданий h различных типов на m вычислительных узлах. Рисунок 15 иллюстрирует отдельную очередь X_j для каждого типа заданий, где j $\in \overline{1,h}$.

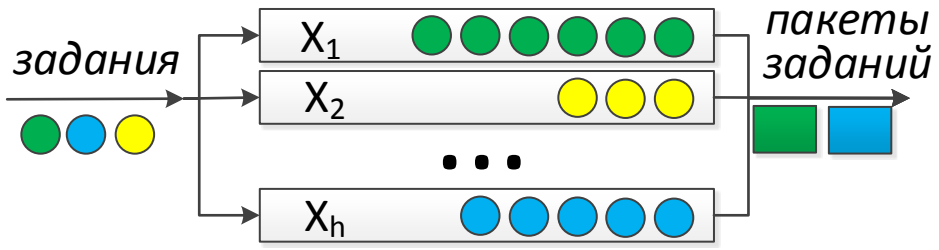

Рисунок 15 – Формирование пакетов заданий по очередям

Для каждого i задания, где $i \in \overline{1,n}$ известны (согласно модели, приведённой в разделе 1.2) его момент поступления задания в систему r_i и заказанное время счёта e_i, для каждой очереди задано рекомендуемое максимальное время ожидания в очереди X_j^{max} (согласно частной модели, рассмотренной в разделе 3.1.1). Необходимо сформировать пакет из заданий одного типа.

Для работы методики должна быть задана функция определения веса очереди $W(X_i)$ (см. раздел 3.2.3). В следующих разделах каждый из этапов предлагаемой методики будет рассмотрен подробнее. Методика ФПЗТ состоит из следующих этапов:

Этап 1. Определить момент формирования пакета заданий $t_{формирования}$ (рассматривается в разделе 3.2.2).

Этап 2. Среди всех непустых очередей выбрать очередь T_j наибольшего веса $W(X_j)$, где $W(X_j) = max(W(X_i))$, $i \in \overline{1,h}$ (рассматривается в разделе 3.2.3).

Этап 3. Из выбранной очереди X_j отобрать задания для включения в пакет (рассматривается в разделе 3.2.4).

Этап 4. Определить число вычислительных узлов $m_{пак}$ для обработки сформированного пакета (рассматривается в разделе 3.2.5).

Этап 5. Начать выполнение сформированного пакета заданий на $m_{пак}$ вычислительных узлах (рассматривается в разделе 3.2.6).

Алгоритмы планирования осуществляют ресурсоёмкую операцию построения расписания, оптимального или субоптимального. Построение расписания производится с учётом нахождения в системе других заданий, окон в плане запуска заданий и наличия других заданий в очереди. При поступлении очередного задания, досрочного завершения выполняющихся заданий или изменении количества доступных ВУ расписание перестраивается. Построение расписания является NP-полной задачей [63], поэтому увеличение числа доступных ВУ и увеличения количества заданий в очереди приводит к нелинейному росту сложности построения расписания, и, как следствие, к увеличению длительности этой процедуры.

В результате применения методики ФПЗТ формируются пакеты заданий, которые будут поданы на вход существующей СУЗ. Это повлечёт за собой двойную процедуру построения расписания – в первую очередь расписание составляется в результате работы методики ФПЗТ, далее СУЗ строит расписание для сформированных пакетов заданий.

Для обеспечения быстрого (желательно, линейно зависящего от числа заданий, числа очередей и числа ВУ) формирования пакетов автором работы предлагается, по возможности, иметь очередь пакетов нулевой длины. В этом случае в СУЗ нет очереди, и каждый поступающий пакет поступает непосредственно на выполнение. Оговорка «по возможности» относительно очереди нулевой длины сделана на случай нештатного уменьшения числа доступных ВУ, например, в результате поломки. В этом случае сформированный для обработки пакет не сможет быть запущен на заказанных ресурсах и будет размещён в очереди СУЗ.

Предлагаемая методика ФПЗТ основана на идее «жадного» алгоритма, где целью каждого этапа является выбор такого размещения заданий, которое обеспечивает наименьшие накладные расходы на обработку пакетов заданий.

Таким образом, были сформированы две особенности методики ФПЗТ:

– обеспечение очереди СУЗ нулевой длины в штатном режиме;

– малая вычислительная сложность процедуры выбора очередного пакета, далее будет показана вычислительная сложность $O(h+n)$ для h типов заданий и n заданий в сформированном пакете.

3.2.2 Определение времени готовности формирования пакета заданий

В работе [22] предложены два способа определения момента времени формирования пакета. Первый способ заключается в формировании очередного пакета каждые k секунд, где k – некоторое положительное число. Недостатком такого способа является ограничение в 1 допустимый тип заданий, и не представляется возможным его адаптация для различных типов заданий. Если каждые k секунд формировать пакет для каждой очереди, то будет образовываться очередь сформированных пакетов, для которых в текущий момент времени недостаточно ресурсов. К моменту освобождения ресурсов в эти пакеты целесообразно включить задания, которые поступили с момента завершения формирования пакета до его действительного времени запуска на освободившихся ресурсах. При использовании второго способа пакет формируется при появлении в очереди k заданий одного типа.

Для обеспечение очереди СУЗ нулевой длины в штатном режиме автором предлагается исследовать вариант формирования пакета заданий при появлении свободных вычислительных узлах и при наличии хотя бы одного задания в очереди. В момент освобождения вычислительных ресурсов выбирается очередь (раздел 3.2.3), из заданий которой формируется пакет заданий. Вычислительная сложность определения свободных вычислительных узлов сводится к запросу СУЗ со сложностью $O(1)$. Определение наличия задания в очереди может быть

произведено с вычислительной сложностью *O(1)*. Таким образом, общая вычислительная сложность первого этапа составляет *O(1)*.

3.2.3 Выбор очереди заданий наибольшего веса

Наиболее существенным этапом в методике ФПЗТ является выбор очереди заданий, для которой будет сформирован пакет заданий. От выбора очереди для формирования пакетов потенциально зависят измеряемые показатели качества обработки потока заданий.

Пакет заданий формируется из очереди наибольшего веса. Вес очереди может рассчитываться с учётом различных параметров. Предлагаемая автором формула вычисления веса очереди расчёта учитывает:

– приоритет очереди;

– время ожидания в очереди самого «старого» заданий относительно рекомендуемого максимального времени ожидания для этой очереди;

– длительности обработки всех заданий в этой очереди относительно времени инициализации заданий этого типа обработки.

Формула веса очереди T_i:

$$W(X_j) = \mathrm{C_j} \times \mathrm{P_j} \times \left(1 + \mathrm{T_j^{cur}}/\mathrm{T_j^{max}}\right) \quad (24)$$

где $W(X_j)$ – вес очереди X_j;

C_j – целесообразность формирования пакета j-й очереди, т.е. отношение времени обработки всех заданий в j-й очереди, где $\mathrm{j} \in \overline{1,h}$ ко времени инициализации этого типа заданий, вычисляемая по формуле

$$C_j = \frac{\sum_{i=1}^{n} e_i}{S_j} \quad (25)$$

$\mathrm{T_j^{cur}}$ – время ожидания самого «старого» задания в очереди T_j;

$\mathrm{T_j^{max}}$ – рекомендуемое максимальное время ожидания в очереди T_j.

Добавление очередного задания осуществляется с вычислительной сложностью *O(1)*.

Каждый из параметров, используемый в формуле, формируется по мере добавления заданий в очередь, и может быть получен за *O(1)*. Расчёт веса

необходимо произвести вычисления для каждой очереди, что составит вычислительную сложность *O(h)*.

3.2.4 Выбор заданий в формируемый пакет заданий

Предлагаемая методика формирования пакетов заданий по типам направлена на уменьшение накладных расходов, связанных с инициализацией вычислительных ресурсов для обработки заданий. Чем больше заданий будет в пакете, тем меньше будут накладные расходы. По этой причине предлагается включать в формируемый пакет все задания выбранной очереди.

Существует возможность задать ограничение на формируемый пакет заданий. Для системы формирования пакетов в целом или для каждого типа заданий в отдельности можно указать такие параметры, как:

– максимальное число заданий в пакете;

– максимальное заказанное время обработки пакета заданий.

С учётом этих параметров для формируемого пакета необходимо отобрать подмножество заданий из очереди. Выделим следующие стратегии:

– отобрать первые поступившие задания;

– отобрать задания, наиболее долго ожидающие в очереди;

– отобрать задания с наивысшим приоритетом;

– отобрать случайные задания.

Наиболее простой случай (включить в пакет все задания выбранной очереди) может быть реализован за *O(1)*. При реализации других стратегий сложность этого этапа может возрасти до *O(k)* и более, где *k* – число заданий в формируемом пакете.

3.2.5 Определение числа ВУ, выделяемых пакету заданий

Введём понятие **показателя масштабирования** формирования пакета заданий. Пусть *k* – показатель масштабирования, тогда для пакета будет выделятся столько ресурсов, чтобы его обработка в k раз превысила время инициализации. Если свободных ресурсов для обеспечения требуемого

соотношения недостаточно, то пакет будет выполняться на свободных вычислительных ресурсах. В этом случае обработка пакета заданий превысит время инициализации более, чем в *k* раз.

Рассмотрим определение числа ВУ для пакета, в который включены n заданий *j*-го способа обработки. Тогда:

s_j – время инициализации *j*-го типа задания;

e_i – время обработки *i*-го задания на 1 ВУ;

k – заданный показатель масштабирования;

Определяется $m_{масшт}$ по формуле:

$$m_{масшт} = \frac{\sum_{i=1}^{n} e_i}{s_j \times k} \quad (26)$$

Пакет формируется для выполнения на $m_{пак} = min(m_{масшт}, m_{своб})$ ресурсах из свободных $m_{своб}$.

После определения числа необходимых пакету заданий ВУ необходимо определить перечень ВУ для использования из числа доступных. Сетевая конфигурация современных вычислителей не является полносвязной, вследствие чего пропускная способность каналов между различными ВУ неоднородна. Для заданий, при обработке которых происходит интенсивный обмен данными между ВУ, правильно подобранные ВУ могут обеспечить существенное уменьшение времени обработки. Вопрос выбора конкретных ВУ выходит за рамки настоящей работы.

3.2.6 Выполнение пакета заданий

При использовании предлагаемых в разделах 3.2.2 и 3.2.5 способах к определению момента формирования пакета и определению числа ВУ для обработки пакета заданий сформированный пакет заполняет свободные вычислительные узлы и начинает выполняться. Предлагается сделать этот этап тривиальным с вычислительной сложностью *O(1)*.

В случае использования других способов или в случае выхода из строя одного из ВУ непосредственно в процессе формирования пакета указанными способами, образуется очередь из пакетов заданий.

Стратегию обработки заданий внутри пакета задаёт прикладной программист, создающий программный модуль инициализации типа заданий.

3.3. Архитектура программного комплекса имитационного моделирования систем управления заданиями

3.3.1 Схемы совмещения работы СУЗ и имитационной модели

Для обеспечения адаптивности ко входному потоку заданий необходимо имитационное моделирования СУЗ осуществлять одновременно с функционированием реальной СУЗ. Рассмотрим различные схемы реализации этой идеи.

Рисунок 16 представляет схему применения имитационной модели обеспечения непрерывного моделирования и исследования СУЗ одновременно с обработкой в СУЗ. В ней входной поток событий, поступающий в СУЗ, продублирован на модель СУЗ (рисунок). В этом случае возникнет проблема – во входном потоке событий отсутствует информация о реальном времени выполнения задания, которое зачастую меньше планируемого времени выполнения.

Рисунок 16 – Схема применения имитационной модели СУЗ в реальном времени

Таким образом, у модели СУЗ во входном потоке нет важного параметра – реального времени выполнения. Рассмотрим схемы, позволяющие его получить.

Рисунок 17 представляет схему применения имитационной модели с добавлением обратной связи от СУЗ к модели СУЗ. После завершения задания необходимо уведомить модель СУЗ о реальном времени выполнения заданий.

Рисунок 17 – Схема применения имитационной модели СУЗ в реальном времени с обратной связью

Рисунок 18 представляет схему применения имитационной модели с прогнозированием реального времени выполнения задания. В этом случае для поступающего задания на основе статистических данных строится прогноз времени его выполнения [121].

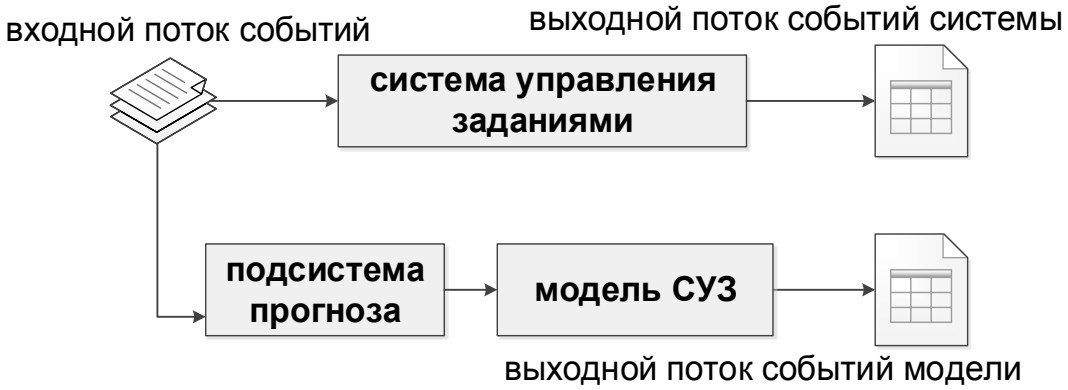

Рисунок 18 – Схема применения имитационной модели СУЗ в реальном времени с подсистемой прогноза

Рисунок 19 представляет схему, являющийся результатом объединения двух предыдущих схем: с подсистемой прогноза и обратной связью. Для каждого поступающего задания прогнозируется реальное время его выполнения. После завершения задания в СУЗ этот прогноз заменяется на реальные данные, что

позволяет непрерывно корректировать прогноз реального времени выполнения, увеличивая его точность.

Рисунок 19 – Применение модели СУЗ в реальном времени с подсистемой прогноза и обратной связью

Применение модели СУЗ в реальном времени позволяет непрерывно сравнивать выходные потоки событий реальной СУЗ и её модели, по мере необходимости корректируя модель для повышения её точности или поиска наилучших настроек.

Применение модели СУЗ в реальном времени используется в разработанной архитектуре программного комплекса имитационного моделирования, которая рассматривается в следующем разделе.

3.3.2 Компоненты разработанного комплекса

Рисунок 20 демонстрирует архитектуру программного комплекса имитационного моделирования СУЗ. Входной поток заданий поступает в подсистему формирования пакетов заданий, в которой по методике ФПЗТ определяет время формирования пакета, его состав и требуемый объём вычислительных ресурсов для его обработки. Сформированный пакет поступает в подсистему интеграции в СУЗ, которая передаёт его на обработку в СУЗ и ожидает получения результатов. Каждый из компонент архитектуры будет рассмотрен далее.

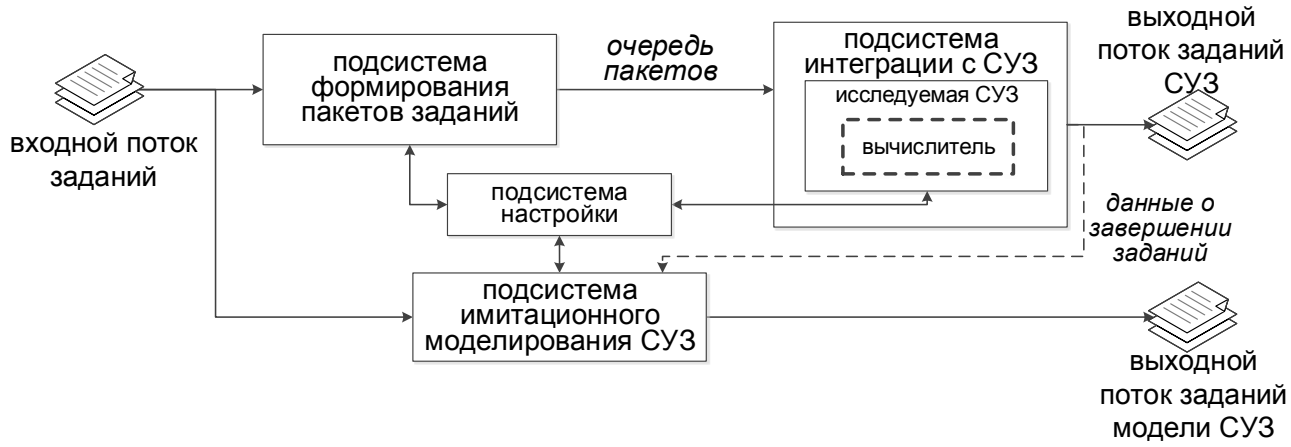

Рисунок 20 – Архитектура программного комплекса имитационного моделирования

Архитектура программного комплекса организована по модульному принципу и включает:

1. Подсистему формирования пакетов заданий, в которой реализована методика формирования пакетов заданий по типам.

2. Подсистема имитационного моделирования, которая отвечает за воспроизведение работы целевой системы. Эта подсистема отвечает за подбор параметров методики формирования пакетов и других настроек СУЗ. Она же отвечает за расчёт точности для проверки соответствия моделирования реальным результатам. В режиме моделирования не требуется наличие реального вычислителя суперкомпьютера. Вычислитель в режиме симулятора является виртуальным, что позволяет эмулировать произвольное количество вычислительных узлов. После запуска задания на виртуальном вычислителе реальных вычислений не производится, а требуемые заданию узлы помечаются занятыми на планируемое время выполнения задания. По истечении реального времени счёта задание снимается со счёта, а соответствующие узлы помечаются как свободные. Такой подход позволяет на малых ресурсах (обычный персональный компьютер) симулировать работу большой вычислительной системы в сотни вычислительных узлов.

3. Подсистема интеграции с СУЗ обеспечивает постановку заданий в произвольную СУЗ, то есть реализует абстракцию над целевой СУЗ, обеспечивающий совместимость с различными реализациями (Slurm, PBS, LSF,

СУППЗ и др.) без модификации их исходного кода. Для функционирования системы формирования пакетов требуется реализовать функции постановки в очередь СУЗ очередного пакета заданий и запроса количества свободных ВУ. Автором реализована интеграция с СУППЗ [35].

Для реализации адаптивности реализован механизм обратной связи за счёт подсистемы имитационного моделирования, где результаты применения настроенной методики ФПЗТ в имитационной модели СУЗ используются для калибровки параметров методики. Это создает замкнутый итеративный оптимизационный цикл "моделирование → оценка → коррекция", который обеспечивает исследования комплекса в различных условиях.

Входной поток заданий поступает на вход подсистеме имитационного моделирования, туда же дублируется выходной поток. Подсистема имитационного моделирования может функционировать в двух режимах.

Первый режим используется для определения точности имитационной модели. В этом режиме производится имитационное моделирование входного потока заданий и получение выходного потока заданий модели, который сравнивается с выходным потоком заданий реальной системы с помощью количественного показателя точности. При снижении точности имитационного моделирования целесообразно ставить вопрос о доработке модели.

Второй режим используется для настройки параметров СУЗ и методики формирования пакетов заданий по типам. Для фиксированного входного потока заданий по указанию администратора проводится серия экспериментов с заданным показателем, например, показателем масштабирования в методике ФПЗТ. В результате анализа выходных потоков экспериментов будут достигнуты подходящие значения показателей качества обработки потока заданий, настройки переносятся в реальную СУЗ. При необходимости процедура настройки системы может циклически повторяться.

Для обеспечения формирования пакетов заданий для произвольной СУЗ и прозрачно для пользователя, необходимо поступившее задание проанализировать и дополнить необходимой для системы формирования пакетов информацией.

Время поступления задания в очередь является характеристикой задания и используется при расчёте веса очереди (см. раздел 3.2.3). Необходимо дополнительно учитывать следующие характеристики задания:

– тип задания;

– время обработки задания на 1 ВУ;

Способы получения указанных характеристик рассматриваются ниже.

3.3.3 Модуль определения типа задания

Модуль определения типа заданий реализуется системным программистом и может реализовать различные стратегии определения типа задания. Тип задания может быть получен одним из следующих способов:

1. Из задания (если явно указан тип в паспорте задания). В настоящей работе рассматривается этот способ.
2. По анализу имени загружаемого задания. Если для заданий определённого типа характерен некий принятый для этой системы шаблон именования, то можно сопоставлять тип заданию, исходя из найденного шаблона.
3. По анализу особенностей задания. По размеру, или имени пользователя, или имени группы пользователя, загрузившего задания.
4. После использования различных методов кластеризации [122].

3.3.4 Модуль определения времени обработки задания

Время обработки задания на 1 ВУ необходимо для определения веса очереди (см. раздел 3.2.3) и определения количества ВУ для обработки сформированного пакета заданий (см. раздел 3.2.5). Время обработки обычно зависит от типа задания и может быть получено одним из следующих способов.

1. Для линейно масштабируемых заданий время обработки можно оценить с использованием комбинаторики, для этого время выполнения 1 ВУ нужно разделить на число выделенных ВУ.

2. Приемлемую в некоторых случаях точность могут обеспечить статистические методы. По базе данных обработанных заданий можно произвести оценку длительности обработки очередного задания.

3. Время обработки может быть явно указано в исходном задании. Тогда указанные выше методы (1) и (2) могут быть использованы для увеличения точности оценки времени обработки.

4. Возможно рассмотреть различные эвристики, например, получение времени обработки очередного задания как усреднение двух предыдущих запусков подобного задания.

3.3.5 Влияние разработанного программного комплекса на показатели качества

Для внедрения разработанного программного комплекса формирования пакетов необходимо задать:

1. Программный модуль определения типа задания, который возвращает тип задания при его добавлении в очередь.

2. Показатель масштабирования (подробнее см. в 3.2.5) является параметром, от которого зависит количество выделяемых для пакета заданий ВУ. Влияние этого показателя рассматривается в разделе 4.4.

3. Обработчик пакета заданий для каждого из h типов заданий. Обработчик является модулем, исполняемым при запуске обработки пакета заданий определённого типа. Для каждого типа заданий ведётся собственная очередь.

4. Программный модуль интеграции с СУЗ. Автором реализована интеграция с СУППЗ.

Существует возможность анализировать текущие показатели качества и динамику их изменения. При необходимости, администратор может изменять настройки системы для обеспечения требуемых показателей качества. Так, увеличение показателя масштабирования формирования пакета заданий приводит к укрупнению пакетов заданий, что должно положительно сказываться на эффективности использования вычислительных ресурсов и отрицательно – на среднем времени нахождения задания в системе.

Таким образом, разработанная система является инструментальным средством для исследования зависимости показателей качества от изменяемых параметров системы и поиска требуемого компромисса между взаимозависимыми показателями качества. Для этого необходимо исследовать влияние пакетирования и его параметров на показатели качества системы при заданных характеристиках входного потока заданий, что рассматривается в главе 4. На основе полученных результатов исследования необходимо разработать методику настройки системы пакетирования и рекомендации по её применению.

Выводы по главе 3

Предложена частная модель системы обработки потока линейно масштабируемых заданий с поддержкой формирования пакетов заданий по типам, обладающая следующими отличительными свойствами:

- наличием отдельных очередей для каждого типа заданий;
- общими накладными расходами на обработку задания для каждого типа.

Разработана методика формирования пакетов заданий по типам, основанная на организации для заданий каждого типа отдельной очереди с определяемыми весами и позволяющая за счет группировки однотипных заданий в пакеты и однократной инициализации заданий пакета повысить полезную загрузку вычислительных ресурсов. Методика позволяет определить момент запуска, размер пакета заданий и требуемое количество вычислительных узлов для обработки сформированного пакета.

Разработана архитектура программного комплекса имитационного моделирования систем управления заданиями, позволяющая проводить динамическую настройку систем управления заданиями в соответствии с характеристиками входного потока заданий за счет реализации механизма обратной связи для адаптивной настройки параметров моделирования и обеспечивающая интеграцию с различными системами управления заданиями без модификации их исходного кода. Архитектура соответствует требованиям, сформулированным в 1 главе, и отличается от известных решений

количественной оценкой точности имитационного моделирования, динамической адаптивностью к входному потоку заданий и интеграционной совместимостью с произвольной системой управления заданиями. Архитектура реализована в виде программного комплекса имитационного моделирования.

Изменение параметров входного потока приводит к необходимости решения актуальной научной задачи – исследования эффекта от внедрения системы пакетирования для различных вариантов входных потоков и разработки методики адаптации параметров системы пакетирования к ним.

4. Анализ эффективности методики формирования пакетов заданий по типам

Как было отмечено в главе 1, показатели качества систем управления заданиями взаимозависимы. Одной из задач исследования являлось определение степени влияния предложенной методики формирования пакетов заданий по типам на показатели качества СУЗ, основным из которых является полезная загрузка вычислителя. Глава посвящена имитационному моделированию системы управления заданиями.

Сформулирован порядок проведения экспериментального исследования, рассмотрены исследуемые входные потоки заданий (раздел 4.1), представлены результаты экспериментов по оценке влияния методики формирования пакетов заданий на показатели качества обработки заданий (раздел 4.2). Исследованы границы применимости разработанной методики (раздел 4.3), определена минимальные накладные расходы на обработку, при которой методика улучшает показатели качества в зависимости от интенсивности и однородности входного потока заданий. Исследовано влияние показателя масштабирования на показатели качества (раздел 4.4), представлена разработанная методика выбора значения показателя масштабирования (раздел 4.5).

4.1. Порядок проведения оценки влияния методики формирования пакетов заданий по типам на показатели качества обработки заданий

Для оценки влияния реализованного методики ФПЗТ использовалось имитационное моделирование. Вычислительные эксперименты с имитационной моделью проводились в три этапа:

Этап 1. Оценка влияния методики формирования пакетов заданий по типам на показатели качества обработки потока заданий.

Этап 2. Определение границ применимости методики формирования пакетов заданий по типам.

Этап 3. Определение значения показателя масштабирования заданий для обеспечения требуемых показателей качества обработки потока заданий с заданными характеристиками.

На **первом этапе**, начальном рассматривалась задача оценки влияния разработанной методики ФПЗТ на качество обработки потока заданий. Для этого оценивалось влияние методики ФПЗТ на полезную загрузку вычислительных ресурсов и другие показатели качества обработки потока заданий. Целью было удостовериться в положительном эффекте от применения предложенной методики.

На **втором этапе** целью было определить границы применимости методики ФПЗТ. Для больших НРО методика ФПЗТ положительно влияет на показатели качества обработки потока заданий. С уменьшением НРО положительный эффект от внедрения методики ФПЗТ сокращается. Экспериментально была определены минимальные НРО, при которых методика ФПЗТ положительно влияет на показатели качества обработки потока заданий. На этом этапе рассматриваются входные потоки различной однородности и интенсивности.

Целью **третьего, заключительного, этапа** было определить значение показатель масштабирования, параметра методики ФПЗТ, обеспечивающего наилучшие значения требуемых показателей качества для входного потока заданий с заданными характеристиками.

Эксперименты проводились с помощью метода имитационного, рассмотренного в разделе 2.5. На вход имитационной модели подавались различные входные потоки заданий, сформированные с помощью генерации входного потока заданий с требуемыми статистическими характеристиками (рассмотрены в разделе 2.2).

Рассмотрим используемые в экспериментах входные потоки заданий.

Для проведения экспериментов было сформировано множество входных потоков. Анализ списка топ-500 наиболее производительных суперкомпьютеров мира показывает, что медианное значение количества вычислительных узлов

составляет 398. В качестве вычислителя использовалась виртуальная вычислительная установка из 500 вычислительных узлов.

Для исследования показателей качества необходимо рассчитывать показатели за длительный период времени. После запуска эксперимента задания начинают поступать на пустой вычислитель, в результате чего исследуемая СУЗ работает в недогруженном режиме. В эксперименте необходимо дождаться наполнения вычислителя и перехода работы СУЗ в устойчивый режим, когда показатели качества варьируются в небольшом диапазоне значений. В рамках предварительных исследований было выявлено, что устойчивый режим для рассмотренных входных потоков достигался за несколько часов. Моделирования проводилось для 4 дней работы суперкомпьютера, и последнее задание поступало в очередь на 5 день. Показатели качества рассчитывались от 1 до 4 дня, то есть с запасом отбрасывался 1 день как работа в неустоявшемся режиме, и все дни после 4, когда задания в очередь уже не поступают, но есть очередь и задания продолжают обработку. То есть расчёт всех показателей качества осуществлялся без учёта начального и конечного отрезков времени проведения эксперимента, когда заданий для полной загрузки вычислителя либо ещё недостаточно, либо уже недостаточно.

Рисунок 21 иллюстрирует схему формирования входных потоков для экспериментального исследования.

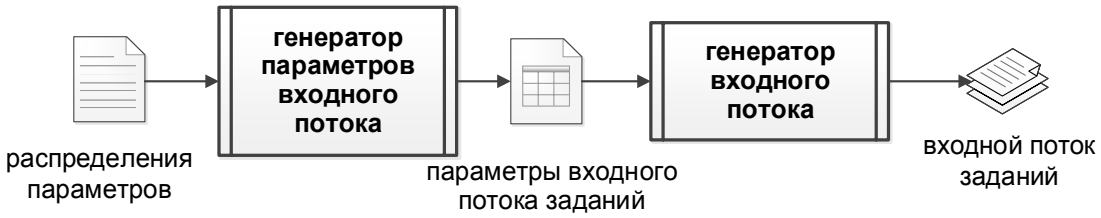

Рисунок 21 – Схема формирования входных потоков для экспериментального исследования

В качестве основы использовался генератор входного потока из работы [117] (далее Lublin&Feitelson). Генератор Lublin&Feitelson создает поток заданий, статистически схожий с реальным входным потоком суперкомпьютера на основе анализ журналов работы нескольких суперкомпьютеров за длительные периоды

времени. Генерируемыми параметрами задания являются: время поступления, время выполнения, требуемое число узлов суперкомпьютера. Каждый из параметров моделируется случайной величиной. Вид и параметры распределения этих величин взяты из работы [117]. Тип задания моделировался равномерной случайной величиной от 1 до 8.

Для каждого потока определялась расчётная загрузка по формуле

$$S_{calc} = \frac{\sum_{i=1}^{n} \mathrm{e_i} \times p_i}{\mathrm{D} \times \mathrm{P}} * 100\% \qquad (27)$$

где e_i – время обработки задания i, p_i – требуемое число узлов для задания i, n = 5000 – число заданий, D = 4 – длительность эксперимента до поступления в очередь последнего задания, P – число узлов виртуального суперкомпьютера.

Всего было сгенерировано 50 наборов входных потоков по 5000 заданий каждый с разной загрузкой. Из них было отобрано 17 потоков с реальной загрузкой от 60 до 100%.

Первая группа входных потоков под условным названием flow1 была сформирована на основе генератора Lublin&Feitelson. Как показал анализ, потоки группы flow1 характеризуются существенной неоднородностью времени обработки заданий, которое варьируется от 1 секунды до нескольких суток.

Для исследования имитационной модели в условиях более однородного входного потока генератор Lublin&Feitelson был модифицирован автором диссертации, и с использованием модифицированного генератора была сформирована группа с условным названием flow2, включающая более однородные по сравнению с flow1 входные потоки. Интервалы между поступления заданий в очередь были смоделированы распределением Пуассона с интенсивностью $\lambda=0,0142$, которая подбиралась эмпирически для обеспечения требуемой расчетной загрузки S_{calc}. Время выполнения заданий было задано гамма-распределением с параметрами $\alpha = 4,2$ *и* $\beta = 0,94$ [117]. При этом для формирования интенсивного потока небольших заданий отбрасывались задания со временем выполнения менее e^5 (148 секунд) и более e^9 (2980 секунд = 50

минут). Было сформировано множество входных потоков flow2 с расчетной загрузкой 0,85 (flow2-0.85), 0,9 (flow2-0.9) и 0,95 (flow2-0.95).

В группах flow1 и flow2 были сформированы по 3 входных потока, обеспечивающих три разных уровня полной загрузки: 0,85 (умеренная загрузка, для иллюстрации работы СУЗ в недогруженном режиме), 0,9 (высокая загрузка, в загруженном режиме) и 0,95 (интенсивная загрузка, в перегруженном режиме).

Названия используемых потоков и их расчётная загрузка приведены в таблице 8.

Таблица 8. Название используемых входных потоков

Вид потоков	Неоднородные			Однородные		
Название	Flow1-0.85	Flow1-0.90	Flow1-0.95	Flow2-0.85	Flow2-0.90	Flow2-0.95
Загрузка	Умеренная	Высокая	Интенсивн.	Умеренная	Высокая	Интенсивн.
Расчётная загрузка в соответствии с (27)	85%	90%	95%	85%	90%	95%

Для каждого из сформированных входных потоков проводилось множество экспериментов с разным временем инициализации задания. Время инициализации задавалось константой для всех заданий каждого эксперимента. Для исследования зависимости влияния доли инициализации замерялась средние накладные расходы на обработку Z (далее – НРО), вычисляемая по формуле

$$Z = \frac{\sum_{i=1}^{n} s_i}{\sum_{i=1}^{n} \mathrm{e_i}} \qquad (28)$$

где s_i – время инициализации задания i, e_i – время обработки задания i, n = 5000 – число заданий в эксперименте.

Для каждого входного потока проводилось 16 экспериментов с НРО от 0% до 10% с шагом 1, после чего от 10% до 50% с шагом с 10. Всего было проведено 96 экспериментов.

Интервалы времени между заданиями были уменьшены так, чтобы последнее задание поступило через 120 часов (завершение 5 дня) после начала эксперимента. Такое масштабирование поступления заданий в очередь позволяет обеспечить одинаковую длительность каждого эксперимента.

Сравнение производилось для алгоритмов обычной очереди (на графиках FCFS) в качестве наиболее простого варианта планирования в СУЗ, алгоритма обратного заполнения (на графиках backfill) в качестве наиболее часто используемого в СУЗ алгоритма и реализованной методики формирования пакетов заданий по типам (на графиках ФПЗТ). Исследования 1 этапа проводились на симуляторе СУППЗ с ВВ, а 2 и 3 этапов на симуляторе Alea.

Рассмотрим результаты первого этапа экспериментов по определению влияния методики ФПЗТ на показатели качества.

4.2. Влияние методики формирования пакетов заданий по типам на показатели качества обработки потока заданий

Для оценки влияния разработанного методики на показатели качества обработки потока заданий проведена серия экспериментов с входными потоками, сформированными с помощью генератора Lublin&Feitelson, где время выполнения заданий было задано гамма-распределением с параметрами $\alpha = 10$ и $\beta = 100$. Такие параметры формируют задания со средним времени обработки 1000 секунд, при этом большинство заданий выполняются от 500 до 1500 секунд. Такие небольшие по размеру задания позволяют получить близкую к 100% полную загрузку вычислителя. Время инициализации задания $t_{иниц}$ в зависимости от эксперимента составляло 10%, 25% и 100% от среднего времени выполнения задания. Эксперименты проводились на симуляторе СУППЗ с ВВ.

Рассмотрим характерную зависимость загрузки вычислителя (полной и полезной на одном графике) от времени проведения экспериментов для 100% НРО. Полная загрузка составляет 100% и совпадает с горизонтальной линией. Полезная загрузка же колеблется около 50%. Рисунок 22 иллюстрирует результаты сравнения имитационной модели без пакетирования для 100% НРО. Сплошная синяя линия соответствует полезной загрузке, а красная пунктирная – полной загрузке. Разница между ними иллюстрирует внутренний простой вычислительных ресурсов, затрачиваемый на инициализацию задания.

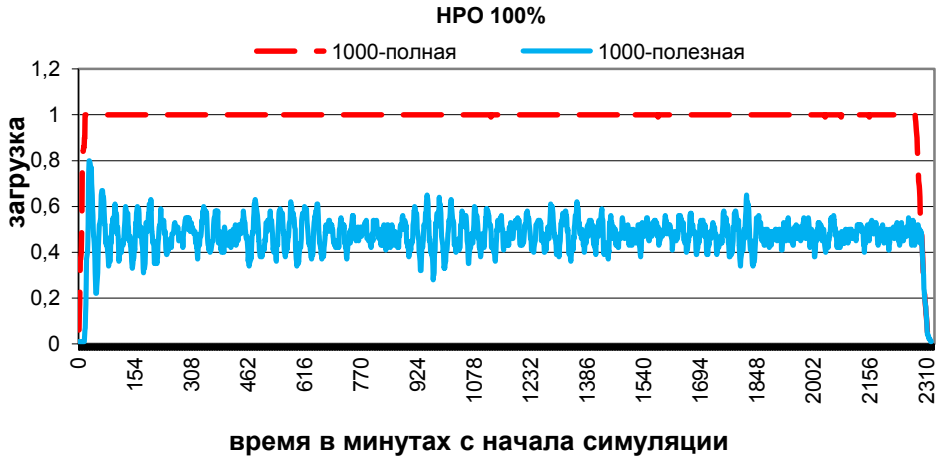

Рисунок 22 – Сравнение полной и полезной загрузки для имитационной модели без использования системы пакетирования

Для этого же потока с 100% HPO рассмотрим график полезной загрузки. Рисунок 23 демонстрирует результаты имитационного моделирования по сравнению варианта с пакетированием и без пакетирования на одном и том же входном потоке заданий. Без группировки полезная загрузка колеблется около 48%, группировка позволяет получить полезную загрузку около 90%. На правой части графика видно, что повышение полезной благодаря пакетированию позволяет раньше завершить обработку заданий. После 1272 минуты также видна особенность пакетирования – комплектование заданий в пакеты на часть свободных ресурсов, в результате чего часть вычислительных ресурсов простаивает. Но этот момент наступает после завершения поступления заданий в очередь, когда система начинает работать в недогруженном режиме.

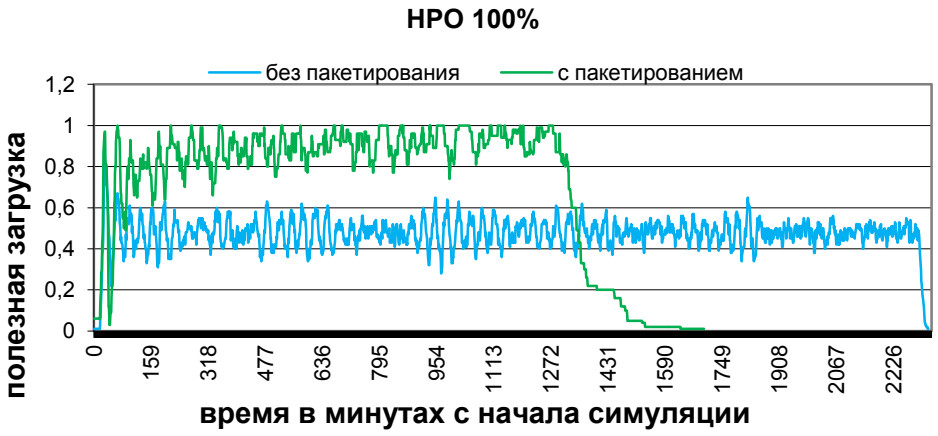

Рисунок 23 – График полезной загрузки от времени с пакетированием и без для эксперимента для 100% HPO

Рассмотрим влияние НРО относительно среднего времени обработки задания. Таблица 11 содержит результаты трёх экспериментов с разными значениями НРО. Во втором столбце показано значение полной загрузки, близкое к 100%. При этом полезная загрузка без пакетирования в третьем столбце показывает, что из-за длительной инициализации вычислитель много времени простаивает. Последний столбец показывает результаты работы пакетирования. Видно, что в ходе проведения всех экспериментов полная загрузка составляет 100% или почти 100%. Алгоритм обратного заполнения способен плотно, без пропусков сформировать план запуска заданий рассматриваемого входного потока. С полезной загрузкой ситуация иная. Для наименьшей доли времени инициализации полезная загрузка составляет уже 86%. С ростом доли времени инициализации полезная загрузка снижается ниже 50% при равном среднем времени обработки и времени инициализации.

Таблица 9. Влияние разных значений НРО на загрузку

НРО	Полная загрузка в соответствии с (5) без пакетирования, %	Полезная загрузка в соответствии с (6) без пакетирования, %	Полезная загрузка в соответствии с (6) с пакетированием, %
10%	97,1	86,2	98,9
25%	100	75,2	96,2
100%	100	48,1	89,4

Для наглядности сравним полезную загрузку с пакетированием и без на диаграмме. Рисунок 24 иллюстрирует зависимость полезной загрузки от НРО 10%, 25% и 100% Для 10% НРО полезная загрузка составляет 86%. С ростом НРО полезная загрузка снижается, при 100% доли инициализации полезная загрузка составляет менее 50%.

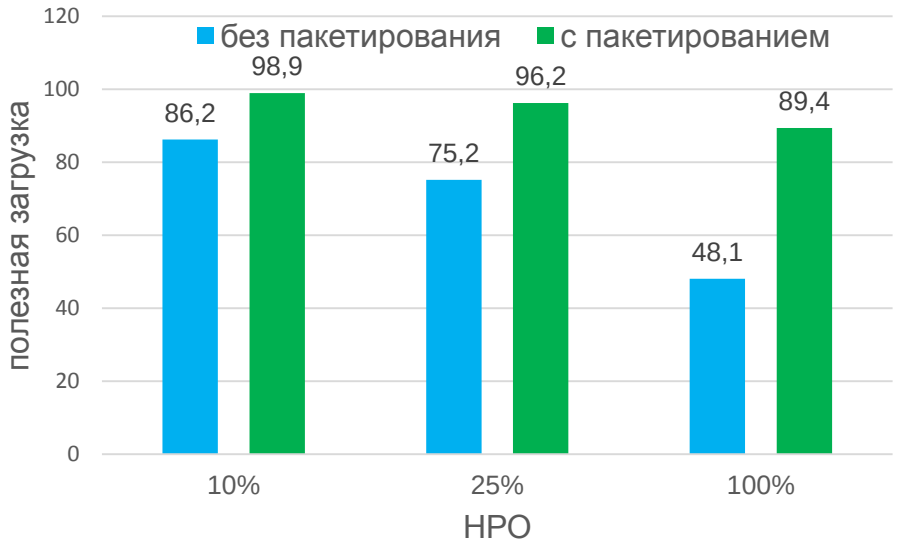

Рисунок 24 – Влияние HPO на полезную загрузку для значений 10%, 25%, 100% с формированием пакетов и без

Введём понятие выигрыша от пакетирования, которое рассчитывается по следующей формуле:

$$V = \frac{U_{\text{пакетир}} - U_{\text{без пакетир}}}{U_{\text{без пакетир}}} \quad (29)$$

Рассмотрим зависимость полезной загрузки от числа типов заданий. Предполагалось, что с ростом числа типов заданий загрузка должна уменьшаться. Таблица показывает отсутствие существенного влияния числа типов заданий на величину полезной загрузки для времени инициализации 1000 секунд. Аналогичные результаты были получены для 100 и 250 секунд инициализации.

Таблица 10. Зависимость полезной нагрузки от числа типов заданий при показателе масштабирования 1 и 100% HPO

Эксперимент	Полезная загрузка в соответствии с (6)	Выигрыш от пакетирования в соответствии с (29), %
Без пакетирования	0,481	0
Пакетирование с 8 типами заданий	0,894	86
Пакетирование с 2 типами заданий	0,874	82
Пакетирование с 1 типом заданий	0,853	77

Рисунок 25 демонстрирует зависимость среднего времени ожидания задания в очереди от показателя масштабирования для 25% HPO и 8 типов заданий. Из графика видно, что для большинства вариантов показателя масштабирования среднее время ожидания уменьшилось с 1330 секунд без пакетирования до 600-800 секунд (или 2200 секунд для показателя в 0,25). Для определения наилучшего значения показателя масштабирования требуется больше экспериментов, что рассматривается в разделе 4.4.

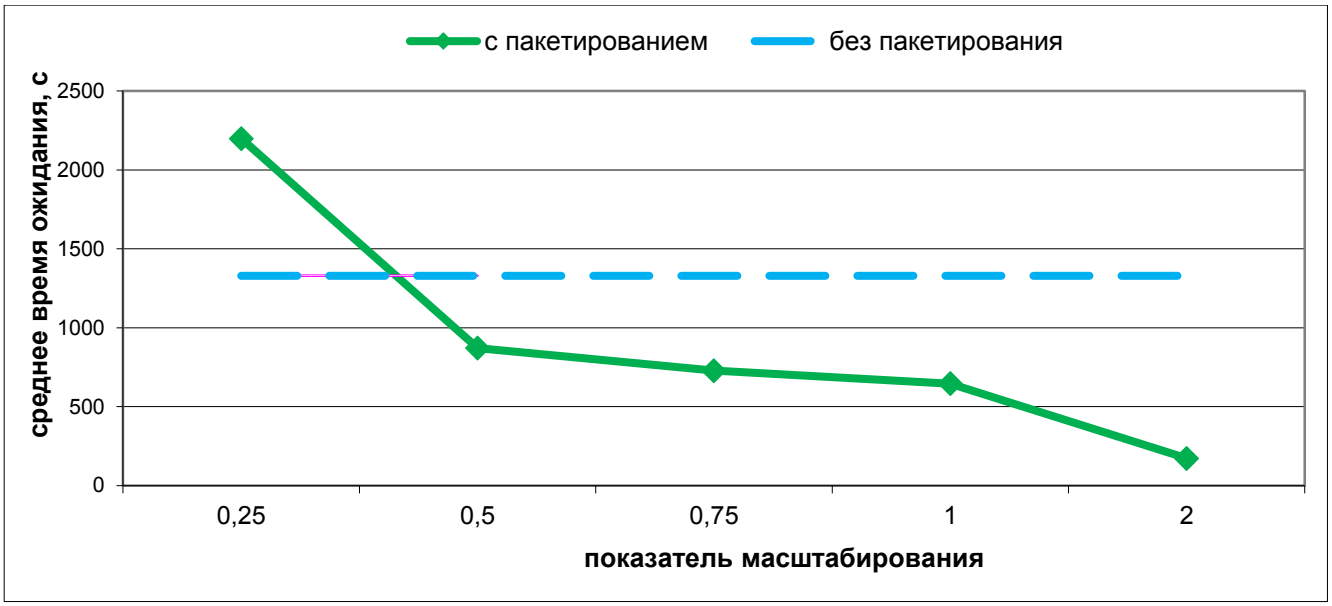

Рисунок 25 – Влияние показателя масштабирования на среднее время пребывания задания в очереди для 25% доли инициализации, 8 типов заданий

Для 10%, 25% и 100% HPO не было выявлено существенного влияния значения показателя масштабирования на полезную загрузку вычислителя. В таблице 11 приведены данные только для 100% HPO для разных значений показателя масштабирования.

В ходе экспериментального исследования на имитационной модели выявлен существенный разрыв между полной и полезной загрузкой вычислительных ресурсов. С ростом значения HPO уменьшается полезная загрузка вычислителя. Применение пакетирования позволяет существенно увеличить полезную загрузку вычислителя. По собранным экспериментальным данным, при HPO в 10% повышение полезной загрузки в результате пакетирования составляет от 5% до 15%, при HPO в 25% — от 20 до 30%, при HPO в 100% — от 75% до 90%.

Таблица 11. Зависимость полезной нагрузки от значения показателя масштабирования при одном типе заданий и времени инициализации 1000 секунд

Эксперимент	Полезная загрузка в соответствии с (6), %	Выигрыш от пакетирования в соответствии с (29), %
Без пакетирования	48,1	
Пакетирование, показатель масштабирования 1	85,3	77
Пакетирование, показатель масштабирования 2	84,3	75
Пакетирование, показатель масштабирования 10	85,5	78
Пакетирование, показатель масштабирования 100	85,8	78

4.3. Границы применимости методики формирования пакетов заданий по типам

Рассмотрим, при какой минимальном значении HPO в зависимости от однородности входного потока заданий пакетирование оказывает положительное влияние на показатели качества обработки заданий.

Для каждого из 6 потоков была проведена серия экспериментов для разного значения HPO от 0% до 50%. Для фиксированного входного потока изменялось время инициализации задания. Было выявлено, что для HPO свыше 20% разброс значений показателей качества слишком большой для отображения на графике. Например, для flow1-0.85 для 50% HPO среднее время для backfill превышает 50000 секунд, а для ФПЗТ составляет менее 5000 секунд.

Интерес представляет минимальное значение HPO, при которой алгоритм ФПЗТ становится стабильно лучше алгоритма backfill. Для удобства анализа значений показателей качества далее будем приводить значения для от HPO 0 до 20%.

Рассмотрим график зависимости числа заданий в очереди во время проведения эксперимента для неоднородного потока с умеренной загрузкой flow1-0.85 с нулевым временем инициализации. Видно, что очередь для ФПЗТ не превышала 200 заданий. При этом вычислительных ресурсов для обработки

достаточно, задания в среднем обрабатываются быстрее, чем поступают. В 3 день число поступающих заданий мало. Рисунок 26 иллюстрирует умеренную интенсивность входного неоднородного потока flow1-0.85.

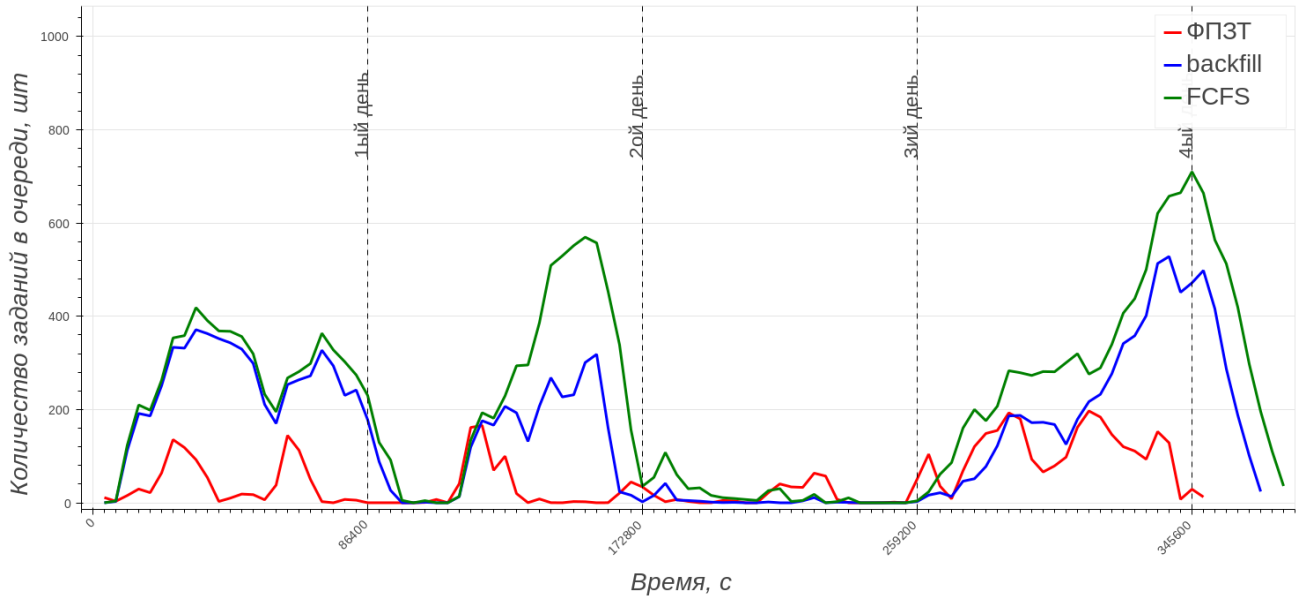

Рисунок 26 – Число заданий в очереди для неоднородного потока с умеренной загрузкой flow1-0.85 не растёт выше 200 для ФПЗТ, значение показателя масштабирования 1

Рисунок 27 иллюстрирует зависимость среднего времени ожидания задания в очереди от значения НРО для группы неоднородных входных потоков flow1. Алгоритму ФПЗТ соответствует сплошная линия, алгоритму backfill – прерывистая. Зелёным цветом с прямоугольным маркером выделена серия экспериментов с потоком умеренной интенсивности flow1-0.85, жёлтой линией с круглым маркером – с потоком высокой интенсивности flow1-0.9, красной линией без маркера – с интенсивным потоком flow1-0.95. Из рисунка видно, что для ФПЗТ среднее время ожидания задания в очереди становится меньше только для 10% НРО. То есть для НРО менее 10% ФПЗТ показывает результаты хуже, чем backfill. Анализ среднего времени ожидания для flow1-0.90 позволяет сделать вывод, что начиная с 8% НРО ФПЗТ обеспечивает меньшее среднее время ожидания, чем алгоритм backfill. Для flow1-0.95 со значения 6% НРО ФПЗТ обеспечивает меньшее среднее время ожидания, чем алгоритм backfill.

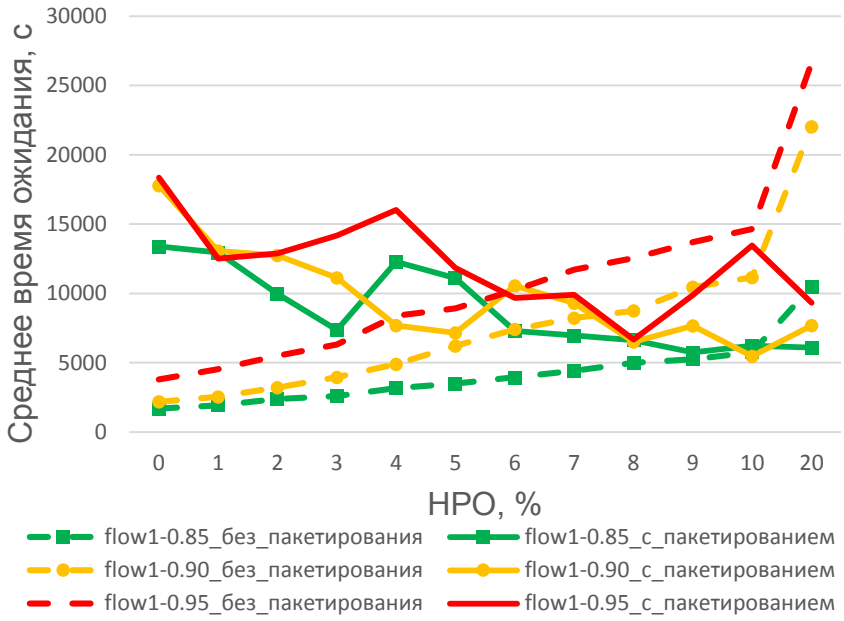

Рисунок 27 – Среднее время ожидания для группы неоднородных входных потоков flow1. Меньше – лучше

Рисунок 28 содержит графики зависимости полной загрузки от HPO заданий во входном потоке для группы неоднородных входных потоков flow1. Для потока с умеренной загрузкой flow1-0.85 для 0, 1 и 4 HPO ФПЗТ превосходит backfill по значению полной загрузки. В остальных ситуациях ФПЗТ хуже backfill по значению полной загрузки.

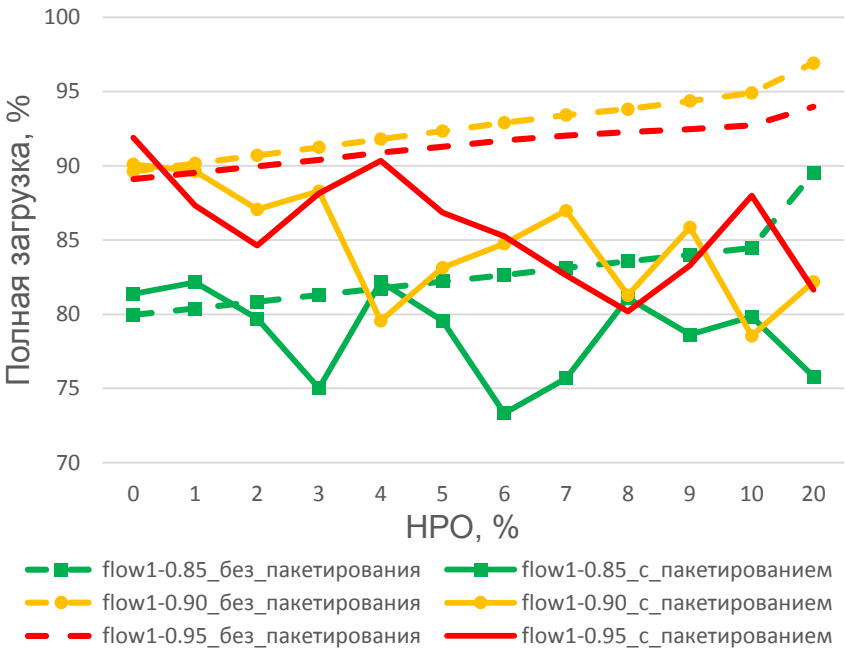

Рисунок 28 – Полная загрузка для группы однородных входных потоков flow2. Больше – лучше

Рисунок 29 содержит график зависимости полезной загрузки от HPO для группы неоднородных входных потоков flow1. ФПЗТ также оказывается лучше, чем backfill для большинства ситуаций.

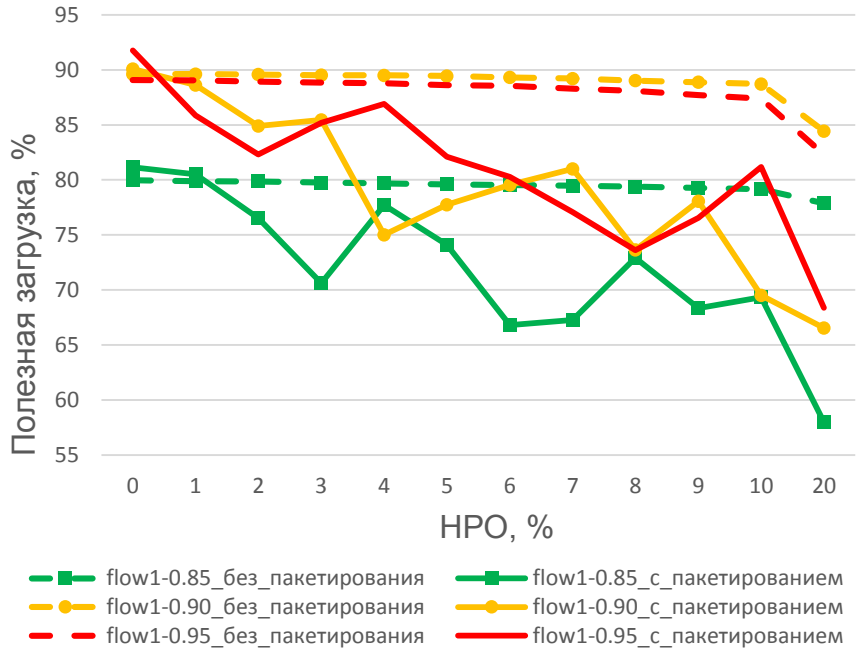

Рисунок 29 – Полезная загрузка для группы однородных входных потоков flow1. Больше – лучше

Из серии экспериментов для группы неоднородных входных потоков flow1 можно сделать вывод, что с ростом интенсивности входного потока уменьшается минимальное значение HPO, при которой ФПЗТ обеспечивает меньшее время ожидания задания в очереди, чем backfill. Для потока умеренной интенсивности flow1-0,85 минимальное значение HPO равно 10%, для высокой интенсивности flow1-0.90 минимальное HPO составляет 8%, для интенсивного потока flow1-0.95 достаточно 6% HPO, чтобы время ожидания задания в очереди для ФПЗТ стало ниже, чем для backfill.

Таблица 12 иллюстрирует, влияние интенсивности входного потока для 8% HPO. С ростом интенсивности входного потока для ФПЗТ медианное время ожидания увеличивается несущественно (3032→3234→3366), в то время как для backfill медиана времени ожидания растёт (1717→6453→9388). Средняя длина очереди для ФПЗТ находится на одном уровне (93,5→93,7→96,5), а для backfill увеличивается (74,3→126,4→178,5).

Таблица 12. Сравнение трёх однородных потоков разной интенсивности flow1 для 8% HPO

	flow1-0.85		flow1-0.90		flow1-0.95	
	backfill	ФПЗТ	backfill	ФПЗТ	backfill	ФПЗТ
Среднее время ожидания задания в очереди в соответствии с (7), с	4981	6611	8735	6501	12554	6675
Медиана времени ожидания задания в очереди, с	1717	3032	6452	3234	9388	3366
Средняя длина очереди	74,3	93,5	126,4	93,7	178,5	96,5

Рассмотрим группу однородных входных потоков flow2. Рисунок 30 демонстрирует график зависимости числа заданий в очереди во время проведения эксперимента для однородного потока умеренной интенсивности flow2-0.85. Первые 4 дня эксперимента поступают входные задания. Вычислительных ресурсов достаточно для своевременного обслуживания заданий, очередь то увеличивается, то уменьшается. Нарастание очереди происходит в 4 день эксперимента.

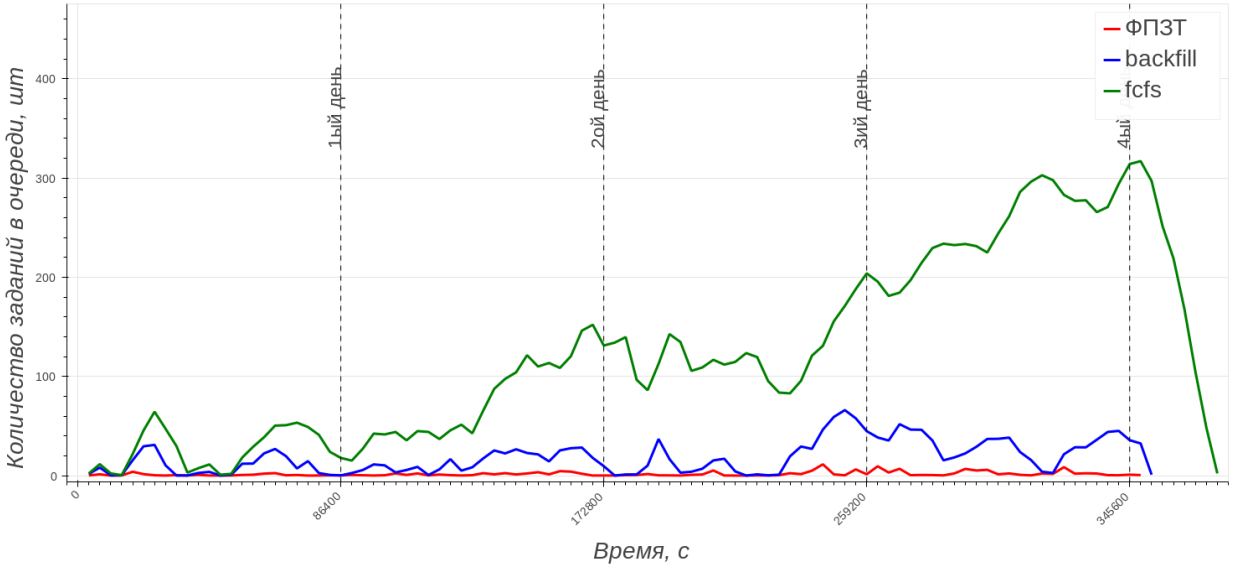

Рисунок 30 – Число заданий в очереди для однородного входного потока умеренной интенсивности flow2-0.85 растёт только в 4 день

В таблице 14 представлены измеряемые показатели качества для эксперимента flow2-0.85 для 0% HPO, то есть при отсутствии инициализации. Медианное время ожидания задания в очереди и полная загрузка для backfill и ФПЗТ сопоставимы.

Таблица 13. Результаты эксперимента для flow2-0.85 для 0% HPO

	FCFS	backfill	ФПЗТ
Среднее время ожидания задания в очереди в соответствии с (7), с	18617	1366	1787
Медиана времени ожидания задания в очереди, с	21187	774	782
Полная загрузка в соответствии с (5), %	79,7	89,0	90,1
Средняя длина очереди	263,8	19,6	26,0

Рисунок 31 отображает зависимость среднего времени ожидания задания в очереди от HPO для группы однородных входных потоков flow2. С ростом HPO время ожидания в очереди увеличивается, но пакетирование позволяет существенно сократить время ожидания относительно backfill.

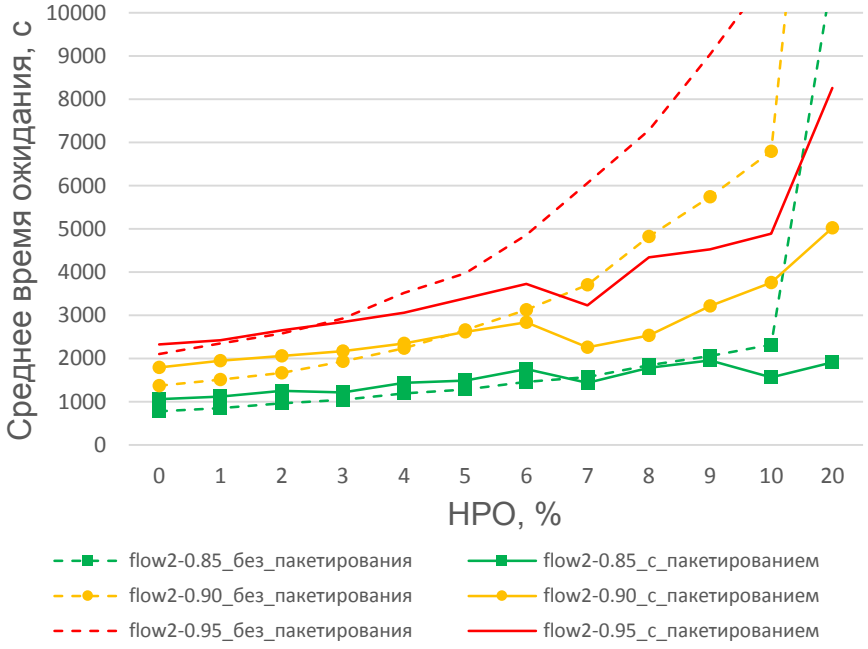

Рисунок 31 – Среднее время ожидания для группы однородных входных потоков flow2

Среднее время ожидания для однородного потока умеренной интенсивности flow2-0.85 для ФПЗТ лучше, чем для Backfill, начиная с 9% HPO. Для потока однородного потока высокой интенсивности flow2-0.9 ФПЗТ превосходит backfill, начиная с 6% HPO. Для потока интенсивного однородного потока flow2-0.95 ФПЗТ превосходит backfill, начиная с 3% HPO.

Рисунок 32 иллюстрирует график зависимости полной загрузки от HPO во входном потоке для группы однородных входных потоков flow2. Для всех экспериментов ФПЗТ обеспечил большую полную загрузку, чем backfill.

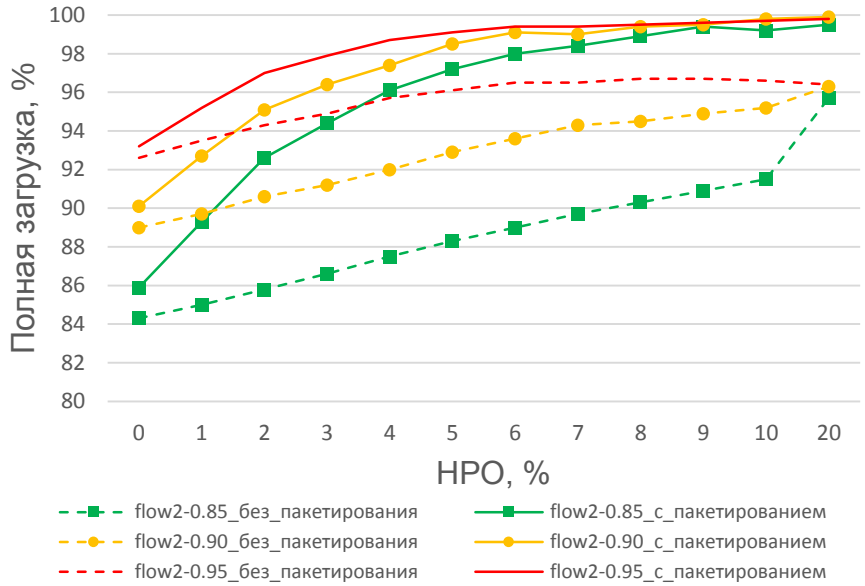

Рисунок 32 – Полная загрузка для группы однородных входных потоков flow2

Рисунок 33 содержит график зависимости полезной загрузки от HPO во входном потоке для группы однородных входных потоков flow2. Для большинства экспериментов ФПЗТ обеспечил большую полезную загрузку, чем backfill.

В таблице 15 приведены данные для экспериментов с однородным входным потоком умеренной интенсивности flow2-0.85 для разных значений HPO. При анализе экспериментов важно рассматривать все показатели качества в комплексе. Например, для приведённого эксперимента среднее время ожидания изменяется несущественно, а полная загрузка – существенно. Здесь же видно

существенное отставание алгоритма FCFS от backfill и ФПЗТ по всем измеряемым показателям качества.

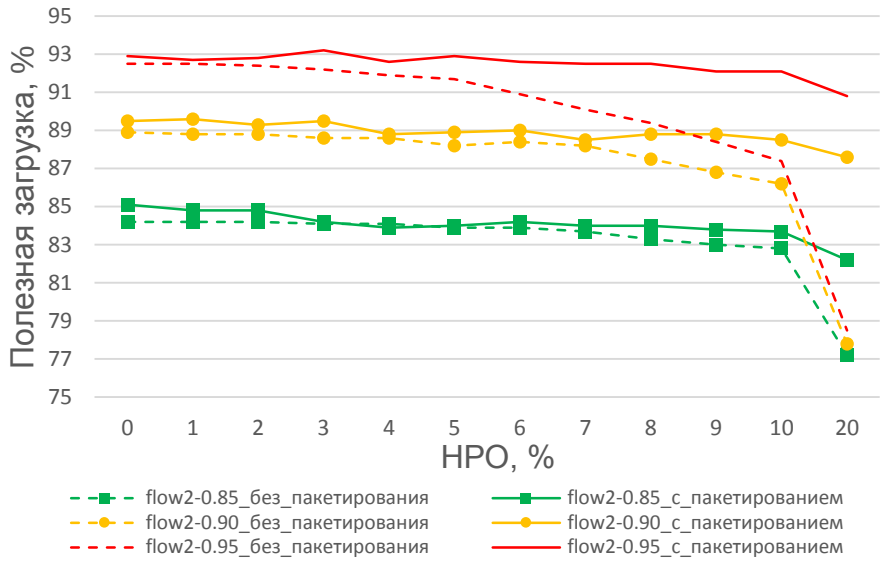

Рисунок 33 – Полезная загрузка для группы однородных входных потоков flow2

Таблица 14. Результаты эксперимента для однородного входного потока умеренной интенсивности flow2-0.85 для HPO 4-6%

HPO	4%			5%			6%		
	FCFS	backfill	ФПЗТ	FCFS	backfill	ФПЗТ	FCFS	backfill	ФПЗТ
Среднее время ожидания задания в соответствии с (7), с	22885	2233	2344	23988	2656	2609	25261	3122	2836
Медиана времени ожидания задания, с	24269	1542	1348	25370	1967	1533	26684	2395	1716
Полная загрузка в соответствии с (5), %	80,3	92,0	97,4	80,5	92,9	98,5	80,6	93,6	99,1
Средняя длина очереди	323,6	31,9	33,4	339,0	37,9	37,3	356,2	44,5	40,4

Серия экспериментов с группой однородных входных потоков flow2 подтверждает вывод о том, что с ростом интенсивности входного потока уменьшается минимальное значение HPO, при которой ФПЗТ обеспечивает меньшее время ожидания задания в очереди, чем backfill.

Таблица 15 представляет сводные результаты обработки проведённых экспериментов. Было определено, что для неоднородного потока умеренной

интенсивности flow1-0.85 ФПЗТ оказывает улучшение показателей качества СУЗ уже с 10% НРО. С ростом интенсивности и увеличением однородности входного потока положительное влияние на показатели качества достигается с 6% НРО для неоднородного входного потока и с 3% для однородного входного потока.

Таблица 15. Минимальное значение НРО, для которого ФПЗТ обеспечивает улучшение показателей качества

Вид потоков	Неоднородные			Однородные		
Название	Flow1-0.85	Flow1-0.90	Flow1-0.95	Flow2-0.85	Flow2-0.90	Flow2-0.95
Загрузка	Умеренная	Высокая	Интенсивн.	Умеренная	Высокая	Интенсивн.
Миним. НРО	10%	8%	6%	9%	6%	3%

4.4. Влияния показателя масштабирования на показатели качества обработки потока заданий

Показатель масштабирования является параметром методики формирования пакетов заданий по типам, влияющим на число выделяемые пакету заданий ВУ согласно формуле (26). В разделе 4.2 показано, что показатель масштабирования влияет на показатели качества. Одним из важных управленческих решений является определение значения показателя масштабирования для методики ФПЗТ. Рассмотрим влияние показателя масштабирования на показатели качества.

Рисунок 34 иллюстрирует снижение среднего времени ожидания с ростом показателя масштабирования для однородного потока высокой интенсивности flow2-0.90 при НРО 30%. Наименьшее ожидание достигается при показателе 1.

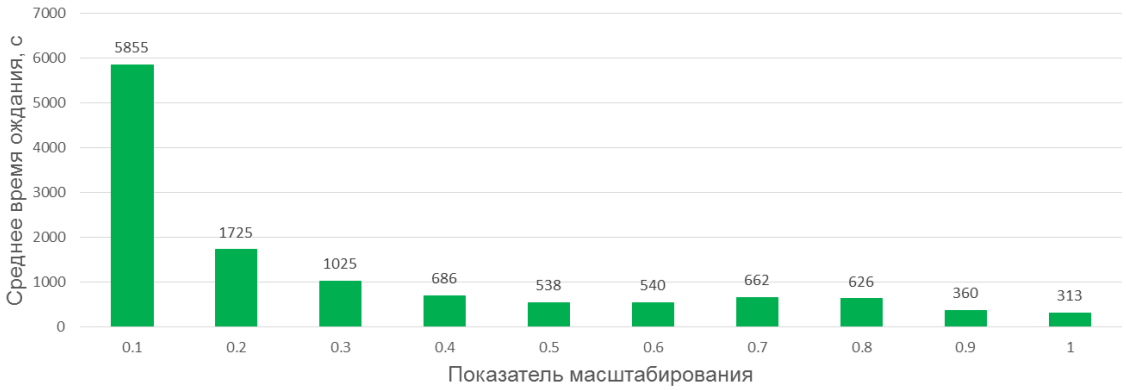

Рисунок 34 – Зависимость среднего времени ожидания от показателя масштабирования (однородный поток высокой интенсивности flow2-0.90, НРО 30%). Меньше – лучше

Рисунок 35 иллюстрирует влияние показателя масштабирования на полную загрузку для однородного потока высокой интенсивности flow2-0.90 при HPO 30%. Наилучшее значение полной загрузки достигается при показатели масштабирования 0,1.

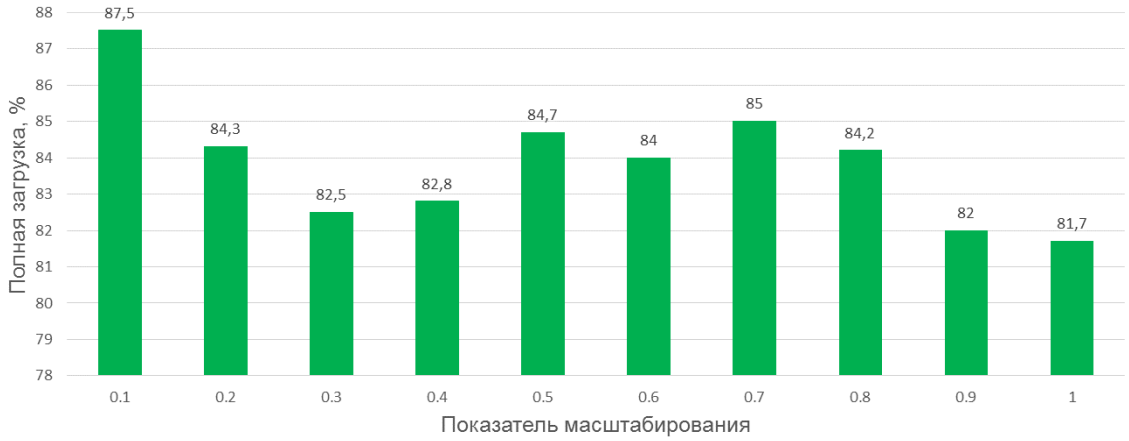

Рисунок 35 – Гистограмма полной загрузки в зависимости от показателя масштабирования (однородный поток высокой интенсивности flow2-0.90, HPO 30%). Больше – лучше

Рисунок 36 иллюстрирует влияние показателя масштабирования на полезную загрузку для однородного потока высокой интенсивности flow2-0.90 при HPO 30%. Наилучшее значение полезной загрузки достигается при показатели масштабирования 0,1.

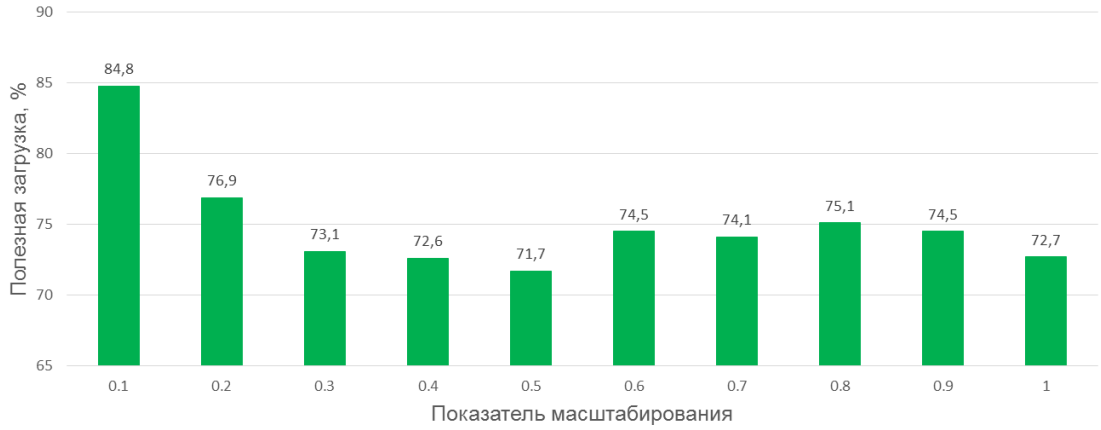

Рисунок 36 – Гистограмма полезной загрузки в зависимости от показателя масштабирования (однородный поток высокой интенсивности flow2-0.90, HPO 30%). Больше – лучше

Таким образом, наилучшее среднее время ожидания достигается при показателе масштабирования 1, а наилучшая загрузка при показателе масштабирования 0,1. Выбор показателя для достижения приемлемого компромисса между показателями качества ложится на администратора СУЗ. Рассмотрим влияние интенсивности входного потока.

Рисунок 37 иллюстрирует влияние показателя масштабирования на среднее время ожидания задания в очереди для группы однородных потоков разной интенсивности flow2 при HPO 5%. Видна тенденция снижения времени ожидания с ростом показателя масштабирования. Значение HPO не оказывает влияния на представленных характер зависимости, меняется только абсолютное значение среднего времени ожидания.

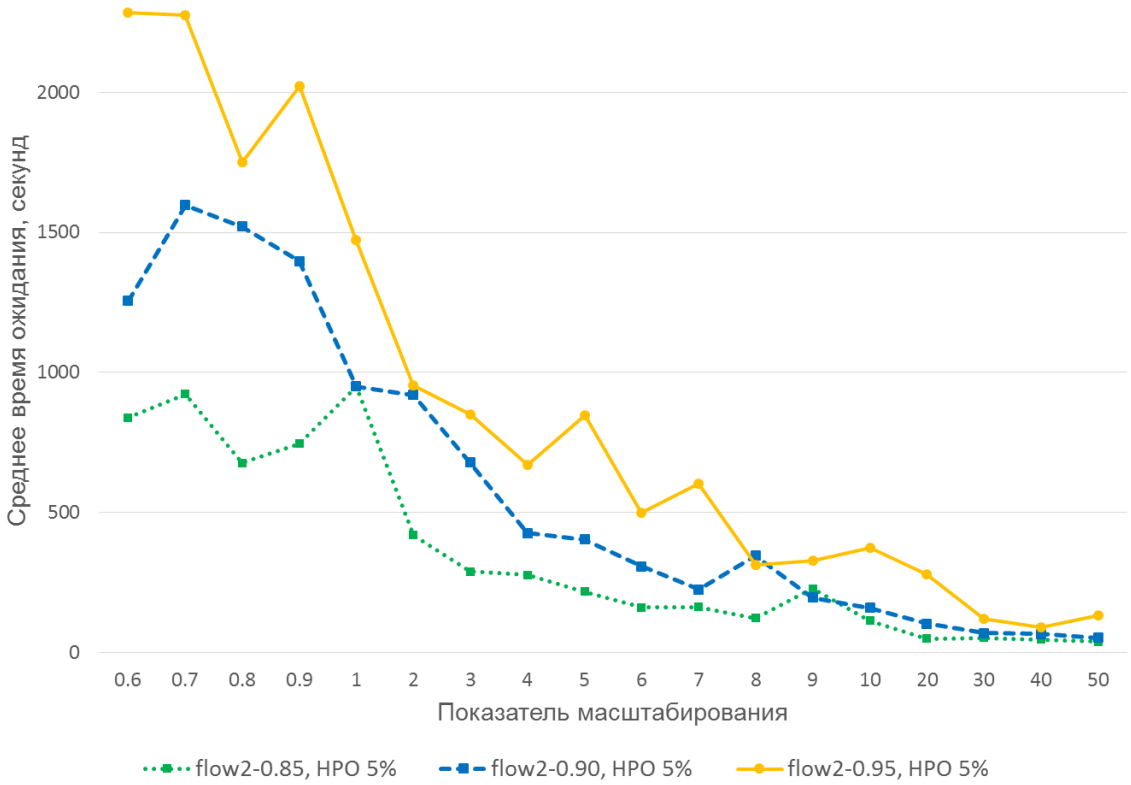

Рисунок 37 – График среднего времени ожидания в зависимости от показателя масштабирования (группа однородных потоков разной интенсивности flow2, HPO 5%). Меньше – лучше

Рисунок 38 иллюстрирует влияние показателя масштабирования на полную загрузку для группы однородных потоков разной интенсивности flow2 при HPO

5%. Видна тенденция снижения загрузки с ростом показателя масштабирования, но изменение загрузки происходит в небольшом диапазоне значений. Значение HPO не оказывает влияния на представленных характер зависимости, меняется только абсолютное значение загрузки.

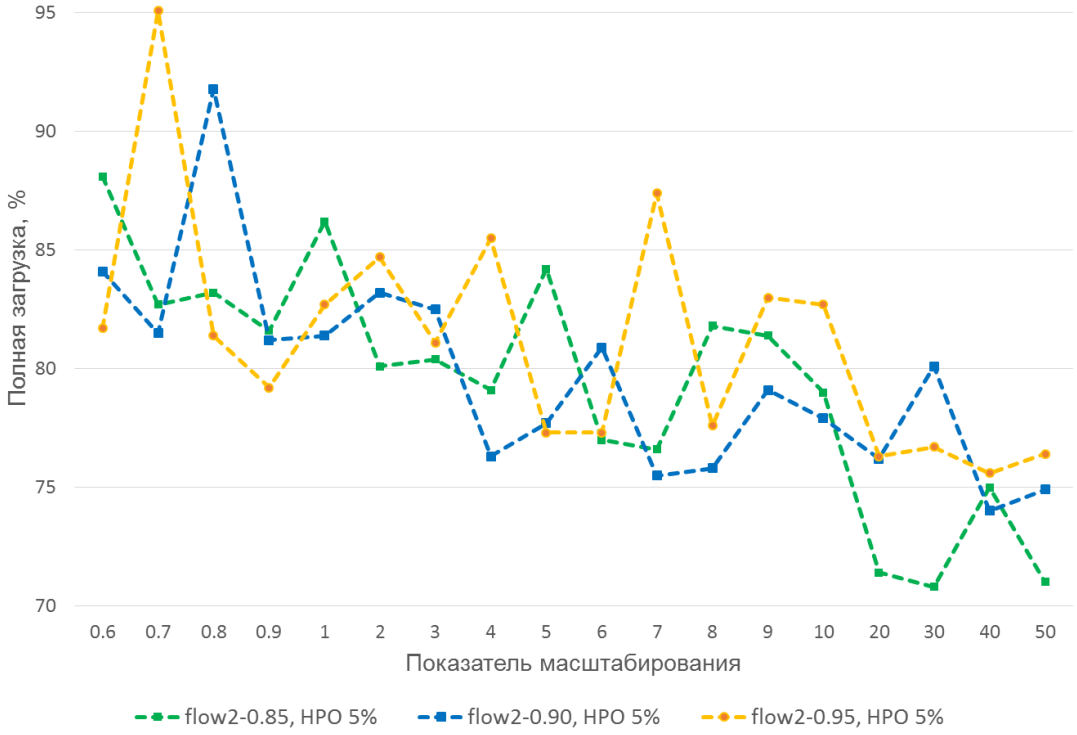

Рисунок 38 – Гистограмма полной загрузки в зависимости от показателя масштабирования (группа однородных потоков разной интенсивности flow2, HPO 5%). Больше – лучше

В результате проведенных экспериментов определено, что существуют значения показателя масштабирования, которые обеспечивают наилучшие значения каждого из показателей качества. Сформулируем методику выбора значения показателя масштабирования.

4.5. Методика выбора значения показателя масштабирования

На основе третьей серии экспериментов была разработана методика автоматизированного подбора показателя масштабирования для имеющегося входного потока заданий с целью обеспечения требуемых показателей качества.

Обозначим k_{min} – такой показатель масштабирования, при котором среднее число ВУ, выделяемых пакету заданий, равно m (количеству ВУ в решающем поле). k_{max} – такой показатель масштабирования, при котором среднее число ВУ, выделяемых пакету заданий, равно 1.

Методика конфигурирования разработанного программного средства ФПЗТ для достижения наилучшего значения выбранного показателя качества.

1. Определить нижнюю границу показателя масштабирования k_{min}. Для этого произвести имитационное моделирование на симуляторе СУЗ с исследуемым входным потоком и с исследуемыми параметрами вычислителя. Нужно подобрать такое k_{min}, чтобы любому пакету выделялся весь вычислитель. Начальное значение $k_{min} = 0{,}1$. Если при k_{min} пакетам выделяется меньше, чем общее число m узлов, необходимо уменьшить показатель масштабирования вдвое и провести моделирование повторно.

2. Определить верхнюю границу показателя масштабирования k_{max}. Нужно подобрать такое k_{max}, чтобы любому пакету выделялся только 1 узел. Начальное значение $k_{max} = 1$. Если при указанном k_{max} пакетам выделяется больше, чем 1 узел, необходимо увеличить показатель масштабирования вдвое и провести моделирование повторно.

3. После нахождения k_{min} и k_{max} всё множество значений между ними необходимо разбить на равные интервалы. Чем больше интервалов исследовано, тем выше точность настройки, но тем больше потребуется провести серий имитационного моделирования. По опыту автора, достаточно исследовать 3 интервала для грубой оценки требуемого показателя качества и 15 интервалов для получения высокой точности.

4. Для каждого значения показателя масштабируемости из интервалов необходимо повторить имитационное моделирование. По результатам построить график зависимости показателей качества от показателя масштабирования.

5. Искомым является показатель масштабирования, соответствующий лучшему значению выбранного показателя качества при достаточных значениях остальных показателей качества.

Выводы по главе 4

Оценка влияния разработанной методики формирования пакетов заданий по типам на показатели качества обработки заданий проведена в трёх сериях имитационного моделирования.

На первом этапе выявлен существенный разрыв между полной и полезной загрузкой вычислительных ресурсов в случае обработки потока линейно масштабируемых заданий с высокими накладными расходами. Результаты показали, что с ростом времени, затрачиваемого на инициализацию вычислительных ресурсов, уменьшается полезная загрузка вычислителя. Применение методики формирования пакетов заданий по типам позволяет существенно увеличить полезную загрузку вычислителя. При 10% НРО полезная загрузка за счёт пакетирования возросла с 86% до 99%, при 100% НРО – возросла с 48% до 89%. Влияние числа типов заданий в диапазоне от 1 до 8 на значение полезной загрузки выявлено не было.

На втором этапе определены границы применимости методики формирования пакетов заданий по типам, а именно минимальное значение НРО для входных потоков заданий разной однородности, при которых пакетирование оказывает положительное влияние на показатели качества обработки заданий. Показано, что для неоднородного потока с умеренной загрузкой пакетирование оказывает улучшение показателей качества обработки потока заданий, начиная с 10% НРО. С ростом интенсивности и повышением однородности входного потока положительное влияние на показатели качества достигается с 6% НРО для неоднородного входного потока и с 3% для однородного входного потока.

На третьем этапе для фиксированного входного потока заданий с заданными характеристиками определён показатель масштабирования, который обеспечивает наилучшее значение выбранного показателя качества обработки потока заданий. В результате разработана методика выбора значения показателя масштабирования для фиксированного входного потока заданий. Разработанная методика позволяет в автоматизированном режиме итеративно изменять показатель масштабирования для поиска такого значения, которое обеспечивает требуемые администратору СУЗ показатели качества обработки заданий.

Заключение

В результате диссертационной работы разработан комплекс новых научно обоснованных технических, технологических, архитектурных решений, внедрение которых вносит вклад в развитие высокопроизводительных вычислительных систем коллективного пользования, как составной части научной инфраструктуры страны. В соответствии с целью и задачами исследования получены следующие результаты.

1. Разработана модель обработки заданий в СУЗ, основанная на представлении вычислительной системы из однородных вычислительных узлов, обрабатывающая поток независимых заданий различных типов. Тип заданий определяет процедуру инициализации вычислительных узлов, длительность которой может быть достаточно велика по сравнению со временем выполнения задания. Для оценки качества обработки заданий в СУЗ выбраны показатели полной загрузки (время инициализации учитывается, как время обработки задания) и полезной загрузки (время инициализации учитывается, как время простоя вычислительных ресурсов), среднее приведённое время ожидания задания в очереди относительно планируемого времени обработки. Для улучшения качества обработки заданий с длительным временем инициализации предложено использовать объединение заданий в группы (пакетирование).

2. Проведённый анализ методов и средств моделирования СУЗ показал, что наиболее подходящим для достижения цели работы методом оценки показателей качества обработки потока заданий является имитационное моделирование. В диссертации были применены имитационная модель на базе симулятора СУППЗ с виртуальным вычислителем и модель-симулятор на базе известного симулятора Alea.

3. Предложенный метод имитационного моделирования систем управления заданиями позволяет оценивать точность исследуемых моделей при помощи количественного показателя близости выходных потоков заданий и улучшать характеристики систем управления заданиями для заданного входного потока

заданий в ходе итеративного процесса настройки параметров модели. Метод отличается от известных применением количественного показателя точности имитационных моделей. С помощью экспериментального исследования показано, что предложенный метод адекватно ранжирует априорно разные имитационные модели по точности моделирования.

4. Предложенная методика формирования пакетов заданий по типам, применимая для обработки потока заданий разных типов в системах управления заданиями позволяет повысить полезную загрузку вычислительных ресурсов за счёт однократной инициализации заданий одного типа и определять момент запуска, размер пакета заданий, требуемое количество вычислительных узлов для обработки сформированного пакета. Для методики формирования пакетов заданий по типам определены границы применимости в виде минимальных значений НРО для входных потоков различной интенсивности и однородности. Методика была реализована в программе для ЭВМ «Подсистема объединения суперкомпьютерных заданий в группы по типам» [35].

5. Предложенная архитектура программного комплекса имитационного моделирования систем управления заданиями позволяет проводить динамическую настройку систем управления заданиями в соответствии с характеристиками входного потока заданий за счет реализации механизма обратной связи для адаптивной настройки параметров моделирования и обеспечивающая интеграцию с различными системами управления заданиями без модификации их исходного кода. Архитектура отличается от известных количественной оценкой точности имитационного моделирования, адаптивностью ко входному потоку заданий и интеграционной совместимостью с произвольной системой управления заданиями.

6. С использованием реализованного программного комплекса проведена оценка влияния методики формирования пакетов заданий по типам на показатели качества обработки заданий. Применение пакетирования позволяет существенно увеличить полезную загрузку вычислителя. При 10% НРО полезная загрузка за счёт пакетирования возросла с 86% до 99%, при 100% НРО – возросла с 48% до

89%. Показано, что для неоднородного потока с умеренной загрузкой пакетирование оказывает улучшение показателей качества обработки потока заданий, начиная с 10% НРО. С ростом интенсивности и повышением однородности входного потока положительное влияние на показатели качества достигается с 6% НРО для неоднородного входного потока и с 3% НРО для однородного входного потока.

7. Разработана методика выбора значения показателя масштабирования для фиксированного входного потока заданий, позволяющая в автоматизированном режиме итеративно изменять показатель масштабирования для поиска такого значения, которое обеспечивает требуемые администратору СУЗ показатели качества обработки заданий.

Результаты по оценке эффективности, рассмотренные в разделе 4, получены в рамках госзадания НИЦ «Курчатовский институт». Результаты работы использованы при совершенствовании систем управления задания в центрах коллективного пользования в МСЦ РАН – Отделении суперкомпьютерных систем и параллельных вычислений НИЦ «Курчатовский институт» при выполнении 7 научно-исследовательских работ по программам фундаментальных научных исследований государственных академий наук на 2013-2020 и на 2021-2030 годы:

– 0065-2018-0409 «Разработка архитектур, системных решений и методов для создания вычислительных комплексов и распределенных сред мультипетафлопсного диапазона производительности, в том числе нетрадиционных архитектур микропроцессоров»;

– 0065-2018-0404 «Исследование и разработка методов сетевой интеграции ресурсов и сервисов научных организаций»;

– 0065-2019-0014 «Исследование и разработка методов и средств организации высокопроизводительных вычислений, создания, обработки, хранения и распределения больших данных и цифрового контента в распределенных информационных и вычислительных средах»;

– 0065-2019-0016 «Разработка архитектур, системных решений и методов для создания и использования высокопроизводительных вычислительных

комплексов, в том числе гетерогенных суперкомпьютеров и нетрадиционных архитектур микропроцессоров»;

– FNEF-2022-0014 «Исследование, разработка и развитие методов и средств организации высокопроизводительных вычислений, интеграции информационных ресурсов различного вида, формирования цифрового пространства научных знаний и интегрированной инфраструктуры научных, образовательных и ведомственных информационных сетей»;

– FNEF-2022-0016, FNEF-2024-0016 «Разработка архитектур, системных решений и методов для создания и использования высокопроизводительных вычислительных комплексов, в том числе гетерогенных суперкомпьютеров и нетрадиционных архитектур микропроцессоров».

Перспективными направлениями дальнейших исследований являются:

1. Развитие модели входного потока заданий при имитационном моделировании за счёт учёта не только линейно масштабируемых заданий.
2. Учёт неоднородности вычислительных ресурсов при формировании пакетов заданий.

Список литературы

1. О Стратегии научно-технологического развития Российской Федерации: указ Президента Российской Федерации № 145 от 28.02.2024 г. // Интернет-представительство Президента России: офиц. сайт. URL: http://www.kremlin.ru/acts/bank/50358 (дата обращения 21.04.2025).

2. Slurm M., Jette A., Wickberg T. Architecture of the Slurm Workload Manager // Lecture Notes in Computer Science. 2023. Vol. 14283. pp. 3–23. URL: https://doi.org/10.1007/978-3-031-43943-8_1.

3. Henderson R. L. Job scheduling under the Portable Batch System. // Lecture Notes in Computer Science, Springer, Berlin, Heidelberg. 1995. Vol. 949. pp. 279–294. URL: https://doi.org/10.1007/3-540-60153-8_34.

4. IBM Spectrum LSF Suite: Installation Best Practices Guide / D. Quintero, M. Black, A.Y. Hussein, B.S. McMillan, G. Samu, J.S. Welch // IBM Redbooks, 2020. – ISBN 9780738458571.

5. Joint Supercomputer Center of the Russian Academy of Sciences: Present and Future / Savin G. I., Shabanov B. M., Telegin P. N., Baranov A. V. // Lobachevskii J. of Mathematics. 2019. Vol. 40. No. 11. pp. 1853–1862. URL: https://doi.org/10.1134/S1995080219110271.

6. Feitelson D.G. Workload Modeling for Computer Systems Performance Evaluation // Cambridge University Press. 2015. URL: https://doi.org/10.1017/CBO9781139939690.

7. Feitelson D.G. Metrics for parallel job scheduling and their convergence // Revised Papers from the 7th International Workshop on Job Scheduling Strategies for Parallel Processing. London, UK: Springer-Verlag, 2001, pp. 188-206. URL: http://dl.acm.org/citation.cfm?id=646382.689681.

8. Klusacek D., Soysa M., Suter F.. Alea - Complex Job Scheduling Simulator // Parallel Processing and Applied Mathematics. PPAM 2019: Lecture Notes in Computer Science, pp. 217-229 (2020). URL: https://doi.org/10.1007/978-3-030-43222-5_19.

9. RM-Replay: A High-Fidelity Tuning, Optimization and Exploration Tool for Resource Management / M. Martinasso, M. Gila, M. Bianco, S. R. Alam, C. McMurtrie, T. C. Schulthess // SC18: International Conference for High Performance Computing, Networking, Storage and Analysis. 2018. pp. 320-332. URL: https://doi.org/10.1109/SC.2018.00028.

10. Lucero A. Simulation of batch scheduling using real production-ready software tools // Proceedings of the 5th IBERGRID, 2011. T.21.

11. A Slurm Simulator: Implementation and Parametric Analysis / N. Simakov, M. Innus, M. Jones, R. DeLeon, J. White, S. Gallo, A. Patra, and T. Furlani // International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems. – Cham: Springer International Publishing, 2017. – pp. 197-217. URL: http://doi.org/10.1007/978-3-319-72971-8_10.

12. Jokanovic A., D'Amico M., Corbalan J, Evaluating SLURM Simulator with Real-Machine SLURM and Vice Versa // 2018 IEEE/ACM Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS): электрон. журн. 2018. pp. 72-82. URL: https://doi.org/10.1109/PMBS.2018.8641556.

13. Batsim: A Realistic Language-Independent Resources and Jobs Management Systems Simulator / P. F. Dutot, M. Mercier, M. Poquet, O. Richard // Job Scheduling Strategies for Parallel Processing. JSSPP 2015, JSSPP 2016. Lecture Notes in Computer Science. 2017. pp. 178-197. URL: https://doi.org/10.1007/978-3-319-61756-5_10

14. AccaSim: a customizable workload management simulator for job dispatching research in HPC systems / C. Galleguillos, Z. Kiziltan, A. Netti, R. Soto // Cluster Comput, 2020, Vol. 23, pp. 107-122. URL: https://doi.org/10.1007/s10586-019-02905-5.

15. Legrand Legrand I. C., Newman H. B. The MONARC toolset for simulating large network-distributed processing systems // Proceedings of the 2000 Winter Simulation Conference: Orlando, FL, USA, 2000. Vol. 2. pp. 1794–1801. URL: https://doi.org/10.1109/WSC.2000.899171.

16. ElastiSim: A batch-system simulator for malleable workloads / T. Özden, T. Beringer, A. Mazaheri, H. Mohammadi Fard, F. Wolf // Proceedings of the International

Conference on Parallel Processing (ICPP '22): New York, NY, USA: Association for Computing Machinery, 2022. pp. 40-1–40-11. URL: https://doi.org/10.1145/3545008.3545046.

17. CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms / R. Calheiros, R. Ranjan, A. Beloglazov, C. De Rose, R. Buyya // Software Practice and Experience: 2011. Vol. 41. pp. 23-50. URL: https://doi.org/10.1002/spe.995.

18. CloudSim 7G: An Integrated Toolkit for Modeling and Simulation of Future Generation Cloud Computing Environment / R. Andreoli, J. Zhao, T. Cucinotta, R. Buyya // Software: Practice and Experience: 2025. Vol. 55. URL: https://doi.org/10.1002/spe.3413.

19. Toporkov V., Yemelyanov D., Bulkhak A. Efficient Resource Selection in Cloud Environments with Volume Discounts and Group Dependencies // Supercomputing. RuSCDays 2024 / ed. by V. Voevodin, A. Antonov, D. Nikitenko. Cham: Springer, 2025. Lecture Notes in Computer Science; vol. 15407. pp. 59-73. URL: https://doi.org/10.1007/978-3-031-78462-0_5.

20. Buyya R., Murshed M. GridSim: A Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing // Concurrency and Computation: Practice and Experience: 2002. Vol. 14. URL: https://doi.org/10.1002/cpe.710.

21. Zvonareva, G., Buzunov D. Peculiarities of Modeling a Specialized Computing System // Statistics and Economics: 2024. № 6. pp. 40-49. URL: https://doi.org/10.21686/2500-3925-2024-6-40-49.

22. Maheswaran M., Ali S. Dynamic Matching and Scheduling of a Class of Independent Tasks onto Heterogeneous Computing Systems // Journal of Parallel and Distributed Computing 1999. Vol. 59, No. 2. pp. 107–131. URL: https://doi.org/10.1006/jpdc.1999.1581.

23. Grouping based job scheduling algorithm using priority queue and hybrid algorithm in grid computing / Rosemarry P., Singh R., Singhal P., Sisodia D. //

International Journal of Grid Computing & Applications (IJGCA), 2012, Vol.3, No.4, pp. 11-20.

24. Muthuvelu, N. A Dynamic Job Grouping-Based Scheduling for Deploying Applications with Fine-Grained Tasks on Global Grids / N. Muthuvelu, J. Liu et al. // Grid Computing and e-Research (AusGrid 2005): Proceedings of the 3rd Australasian Workshop (Newcastle, NSW, Australia, January 30 – February 4, 2005). — Australian Computer Society, 2005, pp. 41–48.

25. Gomathi S., Manimegalai D. An Analysis of MIPS Group Based Job Scheduling Algorithm with other Algorithms in Grid Computing // IJCSI International Journal of Computer Science Issues 2011, Vol. 8, Issue 6, No 3, pp. 285-291.

26. Топорков В. В., Емельянов Д. М., Потехин П. А. Формирование и планирование пакетов заданий в распределенных вычислительных средах // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика 2015. № 2. С. 44-57. URL: https://doi.org/10.14529/cmse150204.

27. Toporkov V., Yemelyanov D., Tselishchev A. Allocation of Distributed Resources with Group Dependencies and Availability Uncertainties // Computational Science – ICCS 2023 / ed. by J. Mikyška, C. de Mulatier, M. Paszynski, V.V. Krzhizhanovskaya, J.J. Dongarra, P.M. Sloot. Cham: Springer, 2023. Lecture Notes in Computer Science; vol. 14077, pp. 599-613. URL: https://doi.org/10.1007/978-3-031-36030-5_48.

28. Костогрызов А.И. Исследование условий эффективного применения пакетной обработки заявок в приоритетных вычислительных системах с ограничением на время ожидания в очереди // «Автоматика и телемеханика». 1987. № 12. С.158-164.

29. Румянцев А. С., Калинина К. А., Морозова Т. Е. Стохастическое моделирование вычислительного кластера с пороговым управлением скоростью обслуживания // Распределенные компьютерные и телекоммуникационные сети: управление, вычисление, связь (DCCN-2017): материалы Двадцатой междунар. науч. конф. Москва: Техносфера, 2017. С. 286—290. URL: https://doi.org/10.17076/mat663.

30. Balci O. Credibility Assessment of Simulation Results // Proceedings of the 1986 Winter Simulation Conference WSC, 1986. pp. 39-44.

31. Balci Balci O. Validation, Verification and Testing Techniques Throughout the Life Cycle of a Simulation Study // Annals of Operation Research 1994. Vol. 53. pp. 121-173.

32. Balci, O. Verification, validation and accreditation // Proceedings of the 1998 Winter Simulation Conference. – 1998. – pp. 41-48.

33. Carson, J.S. Model verification and validation // Proceedings of the 2002 Winter Simulation Conference, 2002, pp. 52-58. URL: https://doi.org/10.1109/WSC.2002.1172868.

34. Sargent R.G. Verification and validation of simulation models // Proceedings of Winter Simulation Conference, Lake Buena Vista, FL, USA, 1994, pp. 77-87. URL: https://doi.org/10.1109/WSC.1994.717077.

35. Баранов А.В., Ляховец Д.С., Шабанов Б.М., «Подсистема объединения суперкомпьютерных заданий в группы по типам». РФ, свидетельство о государственной регистрации программы для ЭВМ № RU2025683856, 09.09.2025.

36. Сравнение систем пакетной обработки с точки зрения организации промышленного счета / А.В. Баранов, А.В. Киселёв, В.В. Старичков, Р.П. Ионин, Д.С. Ляховец // Научный сервис в сети Интернет: поиск новых решений: Труды Международной суперкомпьютерной конференции (Новороссийск, 17-22 сентября 2012 г.). М.: Изд-во МГУ, 2012. С. 506-508 – ISBN 978-5-211-06394-5

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 (23-28 сентября 2013 г., г. Новороссийск). М.: Изд-во МГУ, 2013. С. 410-414. URL: http://agora.guru.ru/abrau2013/pdf/410.pdf

38. Баранов А.В., Киселев 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, М.: Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск), 2014, т.3. 124 с, с. 75-84 – ISSN 2305-9052. URL: http://dx.doi.org/10.14529/cmse140405.

39. Баранов А. В., Киселев Е. А., Ляховец Д. С. Квазипланировщик для использования простаивающих вычислительных модулей многопроцессорной вычислительной системы под управлением СУППЗ // Научный сервис в сети Интернет: многообразие суперкомпьютерных миров: Труды Международной суперкомпьютерной конференции, Новороссийск, 22–27 сентября 2014 года: Российская академия наук Суперкомпьютерный консорциум университетов России. Новороссийск: Издательство Московского государственного университета, 2014. – С. 141-146.

40. Баранов А.В., Ляховец Д.С. Влияние пакетирования на эффективность планирования параллельных заданий // Программные системы: теория и приложения. 2017. Том 8, вып. 1. С. 193-208. URL: https://doi.org/10.25209/2079-3316-2017-8-1-193-208.

41. Баранов А.В., Ляховец Д.С. Экспериментальные оценки влияния пакетирования на некоторые показатели эффективности планирования параллельных заданий // Программные продукты, системы и алгоритмы. вып. 3. – С. 1-8, 2017. URL: https://doi.org/10.15827/2311-6749.24.268

42. Система управления заданиями распределенной сети суперкомпьютерных центров коллективного пользования / Б.М. Шабанов, А.П. Овсянников, А.В. Баранов, П.Н. Телегин, А.И. Тихомиров, Д.С. Ляховец // Труды научно-исследовательского института системных исследований Российской академии наук. 2018. т. 8, № 6, с. 65-73 – ISSN 2225-7349. URL: https://doi.org/10.25682/NIISI.2018.6.0009.

43. Баранов А. В., Лещев С. А., Ляховец Д. С. Методы и алгоритмы снижения фрагментации суперкомпьютерных ресурсов при планировании заданий // Труды научно-исследовательского института системных исследований Российской академии наук. 2018. Т. 8. №. 6. С. 94-102 – ISSN 2225-7349. URL: https://doi.org/10.25682/NIISI.2018.6.0013

44. Measure of adequacy for the supercomputer job management system model / A. Baranov, D. Lyakhovets, G. Savin, B. Shabanov, P. Telegin // Proceedings of the 2019 Federated Conference on Computer Science and Information Systems, FedCSIS 2019, ACSIS, Vol. 18, pp. 423-426. URL: https://doi.org/10.15439/2019F186.

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

46. Облачный сервис для высокопроизводительных вычислений на базе платформы OpenNebula / А.В. Баранов, Б.В. Долгов, Е.А. Киселёв, Д.С. Ляховец // Труды НИИСИ РАН. 2020. Т. 10, № 5-6. с. 5-12 – ISSN 2225-7349. URL: https://doi.org/10.25682/NIISI.2020.5_6.0001.

47. D.S. Lyakhovets, A.V. Baranov. Group Based Job Scheduling to Increase the High-Performance Computing Efficiency // Lobachevskii Journal of Mathematics. 2020. Vol. 41, No. 12, pp. 2558–2565. URL: https://doi.org/10.1134/S1995080220120264.

48. Баранов А.В., Ляховец Д.С. Симулятор системы управления суперкомпьютерными заданиями как научный сервис // Суперкомпьютерные дни в России: Труды международной конференции. 21–22 сентября 2020 г., М: МАКС Пресс, 2020, с. 140-141. URL: https://doi.org/10.29003/m1406.RussianSCDays-2020

49. Simulator of a Supercomputer Job Management System as a Scientific Service / G. Savin, B. Shabanov, D. Lyakhovets, A. Baranov, P. Telegin // ACSIS, 2020, Vol. 21, pp. 413–416, URL: http://doi.org/10.15439/2020F208.

50. Savin G.I., Lyakhovets D. S., Baranov A. V. Influence of Job Runtime Prediction on Scheduling Quality // Lobachevskii Journal of Mathematics. 2021. Vol. 42, No. 11. P. 2562-2570. URL: http://doi.org/10.1134/S1995080221110196.

51. Baranov A. V., Lyakhovets D. S. Accuracy Comparison of Various Supercomputer Job Management System Models // Lobachevskii Journal of Mathematics. 2021. Vol. 42, No. 11. P. 2510-2519. URL: http://doi.org/10.1134/S199508022111007X.

52. Баранов А.В., Ляховец Д.С. Исследование механизма пакетирования суперкомпьютерных заданий в средстве моделирования Alea // Суперкомпьютерные дни в России: Труды международной конференции, Москва, 26–27 сентября 2022 года. – Москва: ООО "МАКС Пресс", 2022. С. 144-145.

53. Lyakhovets D.S., Baranov A.V. Efficiency thresholds of group based job scheduling in HPC systems// Lobachevskii J. Math. 2022. Vol. 43, pp. 2863–2876. URL: https://doi.org/10.1134/S1995080222130261

54. Баранов А.В., Ляховец Д.С. Имитационная модель системы пакетирования суперкомпьютерных заданий на базе симулятора Alea // Программные продукты и системы. 2022. № 4. С. 631-643. URL: https://doi.org/10.15827/0236-235X.140.631-643.

55. Lyakhovets D.S., Baranov A.V., Telegin P.N. Scale Ratio Tuning of Group Based Job Scheduling in HPC Systems // Lobachevskii J Math. 2023. Vol. 44, pp. 5012–5026. URL: https://doi.org/10.1134/S1995080223110240.

56. Баранов А.В., Ляховец Д.С., Константинов П.А. Метод совмещения разнородных потоков заданий в очереди суперкомпьютерных заданий // 13 Национальный суперкомпьютерный форум (НСКФ-2024), г. Переславль-Залесский, 26-29.11.2024 г.

57. Lyakhovets D. S, Baranov A.V., Konstantinov P.A. A Method for Combining Heterogeneous Workflows in HPC Systems // Lobachevskii Journal of Mathematics. 2024. Vol. 45, No. 10. P. 5111-5125. URL: http://doi.org/10.1134/S1995080224606131.

58. Ляховец Д.С., Баранов А.В., Кудрин А.Ю. Симулятор системы управления суперкомпьютерными заданиями с внешним интерфейсом управления // Труды научно-исследовательского института системных исследований Российской академии наук. 2024. Т. 14, № 4. С. 75-83 – ISSN 2225-7349.

59. 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.

60. Reuther A. et al. Scalable system scheduling for HPC and big data // Journal of Parallel Distributed Computing 2018. Vol. 111. pp. 76–92. URL: https://doi.org/10.1016/j.jpdc.2017.06.009.

61. Hou Z., Shen H., Feng Q. Optimizing job scheduling by using broad learning to predict execution times on HPC clusters // CCF Transactions on High Performance Computing 2024. Vol. 6. pp. 365–377. URL: https://doi.org/10.1007/s42514-023-00137-z.

62. Savin G. I., Shabanov B. M., Nikolaev D. S. [и др.] Jobs Runtime Forecast for JSCC RAS Supercomputers Using Machine Learning Methods // Lobachevskii Journal of Mathematics 2020. Vol. 41, № 12. pp. 2593–2602. URL: https://doi.org/10.1134/S1995080220120343.

63. Лазарев А. А., Гафаров Е. Р. Теория расписаний. Задачи и алгоритмы. Москва: Изд-во МГУ, 2011. 224 с. – ISBN 978-5-91450-102-7.

64. Wong A. K. L., Goscinski A. M. Evaluating the EASY-backfill job scheduling of static workloads on clusters // 2007 IEEE International Conference on Cluster Computing. Austin, TX, USA, 2007. pp. 64-73. URL: https://doi.org/10.1109/CLUSTR.2007.4629218.

65. Каляев И.А., Левин И.И. Реконфигурируемые вычислительные системы на основе ПЛИС. – Ростов-на-Дону: Издательство ЮНЦ РАН, 2022. 506 с. ISBN 978-5-4358-0232-0.

66. Aladyshev O. S., Baranov A. V., Ionin R. P. Variants of deployment the high performance computing in clouds // Proceedings of the 2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus). 2018. pp. 1453–1457. URL: https://doi.org/10.1109/EIConRus.2018.8317371.

67. Baranov A. V., Savin G. I., Shabanov B. M. Methods of jobs containerization for supercomputer workload managers // Lobachevskii Journal of Mathematics 2019. Vol. 40, № 5. pp. 525–534. URL: https://doi.org/10.1134/S1995080219050020.

68. Tuli S., Sandhu R., Buyya R. Shared data-aware dynamic resource provisioning and task scheduling for data intensive applications on hybrid clouds using

Aneka // Future Gener. Comput. Syst. 2020. Vol. 106. pp. 595–606. URL: https://doi.org/10.1016/j.future.2020.01.038.

69. Cirne W., Berman F. A model for moldable supercomputer jobs // Proceedings 15th International Parallel and Distributed Processing Symposium. IPDPS 2001 2001. pp. 59-68. URL: https://doi.org/10.1109/IPDPS.2001.925004.

70. Perez-Rua J. Hierarchical Motion-Based Video Analysis with Applications to Video Post-Production // ResearchGate. URL: https://www.researchgate.net/publication/328979329_Hierarchical_motionbased_video_analysis_with_applications_to_video_post-production.

71. Ефимов А. В., Мамойленко С. Н., Перышкова Е. Н. Обработка масштабируемых задач на вычислительных системах с помощью менеджера ресурсов PBS/Torque и планировщика Maui // Программные системы: теория и приложения: 2015. Т. 4, № 3(17).

72. Byun C. et al. Node-based job scheduling for large scale simulations of short running jobs // 2021 IEEE High Performance Extreme Computing Conference (HPEC): 2021. pp. 1–7. URL: https://doi.org/10.1109/HPEC49654.2021.9622870.

73. Sandeep K., Sukhpreet K. Efficient load balancing grouping based job scheduling algorithm in grid computing // International Journal of Emerging Trends in Technology in Computer Science 2013. Vol. 2, № 4. pp. 138–144.

74. Хорошевский В.Г. Распределённые вычислительные системы с программируемой структурой (2010) // Вестник СибГУТИ. 2010. №2 (10). С. 3-41.

75. Brevik J. Eliciting Honest Value Information in a Batch-queue Environment / J. Brevik, A. Mutz, R. Wolski, // 2007 8th IEEE/ACM International Conference on Grid Computing. — 2007. — P. 291–297. DOI: 10.1109/GRID.2007.4354145. Brevik J., Mutz A., Wolski R. Eliciting Honest Value Information in a Batch-queue Environment // 2007 8th IEEE/ACM International Conference on Grid Computing 2007. pp. 291-297. URL: https://doi.org/10.1109/GRID.2007.4354145.

76. Nitro: программный продукт // Adaptive Computing: сайт. URL: http://www.adaptivecomputing.com/products/hpc-products/nitro/.

77. Florian E., Minartz T. Moab evaluation: проектный отчет // University of Hamburg. 2011.

78. Топорков В. В., Бобченков А. В., Емельянов Д. М., Целищев А. С. Методы и эвристики планирования в распределенных вычислениях с неотчуждаемыми ресурсами // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика 2014. № 2. С. 43-62.

79. Gujpti P., Li Y., Lan Z. Fault-aware runtime strategies for high-performance computing // IEEE Transactions on Parallel and Distributed Systems 2009. Vol. 20, No. 4. pp. 460–473. URL: http://doi.org/10.1109/TPDS.2008.128.

80. Etsion Y., Tsafrir D. A short survey of commercial cluster batch schedulers: техн. отчет // The Hebrew University of Jerusalem. Jerusalem, Israel, May 2005. 12 p.

81. Новиков А. Б. Алгоритмы планирования масштабируемых заданий кластерной вычислительной системы // Молодой ученый. 2011. №11. Т.1. С. 74-79. URL https://moluch.ru/archive/34/3884/.

82. Tsafrir D., Etsion Y., Feitelson D. Backfilling using system-generated predictions rather than user runtime estimates // IEEE Transactions on Parallel and Distributed Systems 2007. Vol. 18, No. 6. pp. 789–803. URL: https://doi.org/10.1109/TPDS.2007.70606.

83. Leonenkov S., Zhumatiy S. Introducing New Backfill-based Scheduler for SLURM Resource Manager // Procedia Computer Science 2015. Vol. 66. pp. 661-669. URL: https://doi.org/10.1016/j.procs.2015.11.075.

84. Lelong J., Reis V., Trystram D. Tuning EASY-Backfilling Queues // Job Scheduling Strategies for Parallel Processing : JSSPP 2017 / ed. by W. Cirne, H. Casanova, J. B. Weissman, D. Klusáček. Cham: Springer, 2018. (Lecture Notes in Computer Science; vol. 10773). pp. 43–61. URL: https://doi.org/10.1007/978-3-319-77398-8_3.

85. Gvozdetska N., Globa L., Prokopets V. Energy-efficient backfill-based scheduling approach for SLURM resource manager // 2019 15th International Conference on the Experience of Designing and Application of CAD Systems

(CADSM): Polyana, 2019. pp. 1-5. URL: https://doi.org/10.1109/CADSM.2019.8779312.

86. Feitelson D., Weil A. Utilization and predictability in scheduling the IBM SP2 with backfilling // Proceedings of the First Merged International Parallel Processing Symposium and Symposium on Parallel and Distributed Processing (IPPS/SPDP 1998) 1998. pp. 542–546.

87. Carastan-Santos D., Camargo R., Trystram D. [и др.] One Can Only Gain by Replacing EASY Backfilling: A Simple Scheduling Policies Case Study // 2019 19th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID): 2019. pp. 1-10. URL: https://doi.org/10.1109/CCGRID.2019.00010.

88. Mishra M. K., Mohanty P., Mund G. B. A Time-minimization Dynamic Job Grouping-based Scheduling in Grid Computing // International Journal of Computer Applications 2012. Vol. 40, № 16. pp. 31-38.

89. Byun C. et al. Node-based job scheduling for large scale simulations of short running jobs // 2021 IEEE High Performance Extreme Computing Conference (HPEC): 2021. pp. 1–7. URL: https://doi.org/10.1109/HPEC49654.2021.9622870.

90. Simakov N. A. et al. A Slurm Simulator: Implementation and Parametric Analysis // High Performance Computing Systems. Performance Modeling, Benchmarking, and Simulation. PMBS 2017 / под ред. S. Jarvis, S. Wright, S. Hammond. Cham: Springer, 2018. (Lecture Notes in Computer Science; т. 10724). С. 197–217. URL: https://doi.org/10.1007/978-3-319-72971-8_10.

91. Developing accurate Slurm simulator / N. A. Simakov, R. L. Deleon, Yuqing Lin, Ph. S. Hoffmann, W. R. Mathias // PEARC '22: Practice and Experience in Advanced Research Computing : 2022. pp. 59-1–59-4. URL: https://doi.org/10.1145/3491418.3535178.

92. Гергель В. П., Полежаев П. Н. Исследование алгоритмов планирования параллельных задач для кластерных вычислительных систем с помощью симулятора // Вестн. Нижегор. ун-та им. Н.И. Лобачевского. 2010. № 5-1. С. 201-208.

93. Феоктистов А. Г., Корсуков А. С., Дьядкин Ю. А. Комплексы инструментов для имитационного моделирования предметно-ориентированных распределенных вычислительных систем // Системы управления, связь и безопасность 2016. № 4. С. 30-60.

94. Коваленко Ю. В. Модель с непрерывным представлением времени для задачи составления расписаний с группировкой машин по технологиям // МСиМ. 2013. №1 (27). С.46-55.

95. Flexible job shop scheduling problem with sequence dependent setup time and job splitting: Hospital catering case study / F. Abderrabi, M. Godichaud, A. Yalaoui, F. Yalaoui, L. Amodeo, A. Qerimi, E. Thivet // Applied Sciences 2021. Vol. 11, № 4. pp. 1504. URL: https://doi.org/10.3390/app11041504.

96. Latchoumy P., Khader P. S. Grouping based scheduling with resource failure handling in computational grid // Journal of Theoretical and Applied Information Technology 2014. Vol. 63, № 3. pp. 605–614.

97. Belabid J., Aqil S., Allali K. Solving permutation flow shop scheduling problem with sequence-independent setup time // Journal of Applied Mathematics 2020. Vol. 2020. Ст. 7132469. URL: https://doi.org/10.1155/2020/7132469.

98. Sharma P., Jain A. A review on job shop scheduling with setup times // Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture 2016. Vol. 230, № 3. pp. 517–533. URL: https://doi.org/10.1177/0954405414560617.

99. Гергель В. П., Полежаев П. Н. Исследование алгоритмов планирования параллельных задач для кластерных вычислительных систем с помощью симулятора // Вестник Нижегородского университета им. Н.И. Лобачевского. 2010. № 5-1. С. 201-208.

100. Гнеденко Б.В., Даниелян Э.А., Димитров Б.Н., Климов Г.П., Матвеев В.Ф. Приоритетные системы обслуживания. М.: МГУ. 1973. 448 с.

101. Балыбердин В.А. Методы анализа мультипрограммных систем. М.: Радио и связь, 1982, 152 с.

102. Балыбердин В.А. Оценка и оптимизация характеристик систем обработки данных. М.: Радио и связь, 1987, 176 с.

103. Костогрызов А.И. Исследование эффективности комбинации различных дисциплин приоритетного обслуживания заявок в вычислительных системах // Кибернетика и системный анализ, 1992, Т.28, №1, с. 128-138.

104. Pechinknin A.V., Chaplygin V.V. Stationary Characteristics of the SM/MSP/n/r Queuing System // Automation and Remote Control, 2004, vol. 65, pp. 1429-1443. doi: 10.1023/B:AURC.0000041421.62689.a8.

105. 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.

106. Rumyantsev A., Morozov E. Stability criterion of a multiserver model with simultaneous service // Annals of Operations Research, 2017, Vol. 252, No. 1, pp. 29-39. doi: 10.1007/s10479-015-1917-2.

107. 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.

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

109. AnyLogic, ExtendSim and Simulink Overview Comparison of Structural and Simulation Modelling Systems / I.M. Yakimov, M.V. Trusfus, V.V. Mokshin, A.P. Kirpichnikov // 2018 3rd Russian-Pacific Conference on Computer Technology and Applications (RPC): Vladivostok, Russia, 2018. pp. 1-5. URL: https://doi.org/10.1109/RPC.2018.8482152.

110. Krahl D. ExtendSim 7. 2008 Winter Simulation Conference, Miami, FL, USA, 2008, pp. 215-221. DOI: 10.1109/WSC.2008.4736070

111. Schunk D. Modeling with the Micro Saint simulation package. 2000 Winter Simulation Conference Proceedings (Cat. No.00CH37165), Orlando, FL, USA, 2000, pp. 274-279 vol.1. URL: http://doi.org/10.1109/WSC.2000.899729.

112. Giordano A. A., Levesque A. H. Getting Started with Simulink // Modeling of Digital Communication Systems Using Simulink. John Wiley & Sons, 2015. pp. 1-22. URL: https://doi.org/10.1002/9781119009511.ch1.

113. Optorsim: A grid simulator for studying dynamic data replication strategies / W. H. Bell, D. G. Cameron, F. P. Millar, L. Capozza, K. Stockinger, F. Zini // International Journal of High Performance Computing Applications 2003. Vol. 17, № 4. pp. 403–416. URL: https://doi.org/10.1177/10943420030174005.

114. Chen W., Deelman E. WorkflowSim: A toolkit for simulating scientific workflows in distributed environments // 2012 IEEE 8th International Conference on e-Science: Chicago, IL, USA, 2012. pp. 1–8. URL: https://doi.org/10.1109/eScience.2012.6404430.

115. The MicroGrid: using online simulation to predict application performance in diverse grid network environments / H. Xia, H. Dail, H. Casanova, A.A. Chien // Proceedings of the 2nd International Workshop on Challenges of Large Applications in Distributed Environments (CLADE 2004) : Honolulu, HI, USA, 2004. pp. 52–61. URL: https://doi.org/10.1109/clade.2004.1309092.

116. Benchmarks and Standards for the Evaluation of Parallel Job Schedulers / S. J. Chapin, W. Cirne, D. G. Feitelson, J. P. Jones, S. T. Leutenegger, U. Schwiegelshohn, W. Smith, D. Talby // Job Scheduling Strategies for Parallel Processing / под ред. D. G. Feitelson, L. Rudolph. Berlin, Heidelberg: Springer, 1999. (Lecture Notes in Computer Science; т. 1659). pp. 66–89. URL: https://doi.org/10.1007/3-540-47954-6_4.

117. Lublin U., Feitelson D. G. The Workload on Parallel Supercomputers: Modeling the Characteristics of Rigid Jobs // Journal of Parallel and Distributed Computing. 2003. Vol. 63, № 11. P. 542–546. URL: https://doi.org/10.1016/S0743-7315(03)00108-4.

118. Downey A. A model for speedup of parallel programs: техн. отчет U.C. Berkeley CSD-97-933 // University of California, Berkeley. Dep. of Computer Science. 1997. 25 p.

119. T. H. Le Hai, K. P. Trung and N. Thoai, A Working time Deadline-based Backlling Scheduling solution // 2020 International Conference on Advanced

Computing and Applications (ACOMP), 63-70 (2020). DOI: 10.1109/ACOMP50827.2020.00017.

120. Девятков В. В. Развитие методологии имитационных исследований сложных экономических систем: дис. доктора экономических наук: 08.00.13 / Девятков Владимир Викторович. Москва, 2014. 357 с.

121. Predicting batch queue job wait times for informed scheduling of urgent HPC workloads / N. Brown, G. Gibb, E. Belikov, R. Nash // arXiv.org : электрон. препринт. 2022. URL: https://doi.org/10.48550/arXiv.2204.13543.

122. Shaikhislamov D., Voevodin V. Smart clustering of hpc applications using similar job detection methods // Parallel Processing and Applied Mathematics PPAM 2022 / под ред. R. Wyrzykowski, J. Dongarra, E. Deelman, K. Karczewski. Cham: Springer, 2023. (Lecture Notes in Computer Science; т. 13826). С. 209–223. URL: https://doi.org/10.1007/978-3-031-30442-2_16.