Федеральное государственное бюджетное учреждение науки Институт проблем управления им. В. А. Трапезникова Российской академии наук

На правах рукописи

Тутов Андрей Владимирович

МОДЕЛИ И МЕТОДЫ РАСПРЕДЕЛЕНИЯ ИНФОРМАЦИОННЫХ И ВЫЧИСЛИТЕЛЬНЫХ РЕСУРСОВ ГЕТЕРОГЕННЫХ ЦЕНТРОВ ОБРАБОТКИ ДАННЫХ

Специальность 2.3.8 —
«Информатика и информационные процессы»

Диссертация на соискание учёной степени
кандидата технических наук

Научный руководитель:
доктор технических наук, старший научный сотрудник
Фархадов Маис Паша оглы

Москва — 2026

Оглавление

Стр.

Стр.

Стр.

Стр.

Введение

За последние несколько лет наблюдается беспрецедентный рост использования цифровых сервисов, объемов данных в телекоммуникационных сетях и инфокоммуникационных системах центров обработки данных (ЦОД) в мире. Перед операторами связи и сервис-провайдерами стоят задачи совершенствования собственной ИТ-инфраструктуры с использованием отечественного программного обеспечения для быстрого реагирования на возрастающие потребности общества.

ЦОДы должны предоставлять достаточно ресурсов для обеспечения бесперебойной работы размещенных в них приложений в условиях переменной нагрузки. Помимо традиционных интернет-приложений увеличивается число сервисов, требующих проведения высокопроизводительных вычислений, таких как машинное обучение, обработка больших данных, поддержка приложений инфраструктуры виртуальных рабочих столов и т. д. В связи с высокими требованиями к параллельным вычислениям таких приложений растет спрос на серверы с графическими процессорами (Graphics Processing Unit — GPU). ЦОДы становятся гетерогенными, включающие традиционные серверы и GPU-серверы.

Современные цифровые сервисы базируются на облачных технологиях, в основе которых лежат технологии виртуализации. Без виртуализации невозможно реализовать основные требования облачных сервисов, в том числе динамическое масштабирование (эластичность) ресурсов, использование ресурсов по требованию, оплата только за использованные ресурсы. Данные требования к качеству обслуживания оформляются в виде соглашений об уровне сервиса (Service Level Agreement — SLA).

В связи с разнородностью требований необходимо, чтобы управление ресурсами в облачных ЦОД было эффективным. Для этого используются технологии горизонтального и вертикального масштабирования, живая миграция виртуальных машин, обеспечивающие управление ресурсами ЦОД в реальном времени. Благодаря данным технологиям становится возможным удовлетворять требования к качеству обслуживания при значительно изменяющихся нагрузках на приложения без резерва в виде избыточных ресурсов физических серверов.

Повышение энергоэффективности ЦОД остается одной из главных проблем. На долю цифровых технологий приходится около 12% общемирового потребления электроэнергии. Неэффективное управление ресурсами может привести к неравномерному распределению нагрузки и, как следствие, невыполнению SLA, появлению «горячих» точек в машинных залах, усиленной работе системы охлаждения и, в конечном итоге, к повышенному энергопотреблению.

Системы управления ресурсами в облачных ЦОД должны быть автономными и поддерживать устойчивое состояние с учетом воздействия внешних факторов. Как и в других автономных вычислительных системах в них реализуются функции мониторинга и прогноза нагрузки, планирования и распределения ресурсов.

Необходимо совершенствовать процесс распределения ресурсов, разрабатывать и внедрять эффективные модели, методы и алгоритмы управления ресурсами для нахождения компромисса между множеством противоречивых критериев, таких как обеспечение заданного в SLA-соглашениях качества обслуживания, равномерной загрузки вычислительных ресурсов, минимизации энергопотребления.

На сегодняшний день существует большое количество работ, посвященных моделированию и оценке производительности инфокоммуникационных систем и сетей, повышению эффективности их функционирования, среди которых следует особо отметить отечественные научные группы под руководством Вишневского В.М., Самуйлова К.Е., Воеводина В.В., Смелянского Р.Л., Шабанова Б.М., Каляева И.А., Корнеева В.В., а также групп зарубежных исследователей Buyya R., Menasce D. A. и других.

Число публикаций, в которых отражены результаты исследований таких систем с каждым годом возрастает, что подтверждает актуальность данной проблемы, однако в них рассмотрены преимущественно традиционные и облачные ЦОД, без учета стремительно возросшего спроса на GPU-вычисления. Управление ресурсами на основе совокупности эффективных моделей и методов, которые могут быть положены в основу планировщика вычислительных ресурсов облачного гетерогенного ЦОД остается нерешенной задачей.

Целью диссертационной работы является обеспечение высокой энергоэффективности инфокоммуникационной системы гетерогенного ЦОД и стабильности облачных сервисов путем разработки моделей и методов управления

ресурсами программно-аппаратного комплекса распределенного планировщика ресурсов.

Для достижения поставленной цели в работе решены следующие **задачи**:

1. Анализ проблем повышения эффективности функционирования гетерогенных центров обработки данных.
2. Разработка модели и метода многокритериального первоначального размещения разнородных виртуальных машин с учетом критериев энергопотребления, нарушения SLA-соглашений и загрузки ресурсов.
3. Исследование метода прогнозирования загрузки серверов для принятия решения о необходимости миграции виртуальных машин.
4. Разработка и исследование критерия устойчивости инфокоммуникационной системы ЦОД путем определения оптимального размера окна наблюдения за серверами с учётом возможных помех, вызванных миграциями виртуальных машин.
5. Исследование характеристик процесса миграции виртуальных машин. Разработка метода оценки длительности миграции виртуальных машин.
6. Разработка метода многокритериального выбора целевых серверов для миграции виртуальных машин, обеспечивающий баланс между эффективностью использования вычислительных ресурсов и соблюдением SLA-соглашений.
7. Разработка имитационных моделей. Подтверждение эффективности разработанных моделей и методов.
8. Разработка алгоритмов и рекомендаций по практической реализации моделей и методов программно-аппаратного комплекса распределенного планировщика ресурсов в инфокоммуникационной системе гетерогенного облачного ЦОД, которые могут быть включены в отечественные облачные платформы.

Объектом исследования являются процессы распределения информационных и вычислительных ресурсов в гетерогенных ЦОД.

Предметом исследования являются математические модели и методы эффективного распределения информационных и вычислительных ресурсов в гетерогенных ЦОД.

Методы исследования. Для решения поставленных задач в работе использованы методы теории вероятностей и математической статистики,

многокритериальной оптимизации, прогнозирования и имитационного моделирования.

Научная новизна работы заключается в исследовании и разработке критериев, моделей и методов повышения эффективности программно-аппаратного комплекса планировщика информационных и вычислительных ресурсов гетерогенного ЦОД.

Основными научными результатами являются:

1. Разработаны модель и метод первоначального размещения виртуальных машин в гетерогенном ЦОД, которые, в отличие от существующих аналогов, учитывают ресурсы GPU-серверов и обеспечивают сбалансированное решение по множеству критериев, в то время как применяемые на практике методы обычно ограничиваются единственным критерием (п.1, пнс 2.3.8).
2. Разработаны модель и метод рационального динамического размещения виртуальных машин в ЦОД, направленные на одновременную минимизацию критериев энергопотребления и нарушений SLA-соглашений. Предложенный метод в отличие от используемых на практике, позволяет находить точное решение задачи оптимизации в режиме реального времени и дает в среднем в 3,5 раза лучшие результаты (п.3, пнс 2.3.8).
3. Разработан новый критерий определения рациональной длительности окна наблюдения за серверами. При его расчете используется предложенный метод оценки времени «живой» миграции виртуальных машин, что в совокупности позволяет повысить качество мониторинга и, как следствие, стабильность облачных сервисов (п.3, пнс 2.3.8).

Практическая значимость

1. Исследовано применение метода группового учета аргументов для прогнозирования загрузки физических серверов. Показано, что данный метод позволяет добиться более высокой точности прогноза по сравнению с известными аналогами, что, в свою очередь, ведет к снижению количества необоснованных миграций виртуальных машин.
2. Созданы имитационные модели распределения ресурсов ЦОД, которые позволяют проверить эффективность предложенных методов и алгоритмов динамического распределения ресурсов.

3. Предложены алгоритмы, рекомендации, архитектура и программные модули для практической интеграции моделей и методов программно-аппаратного комплекса распределенного планировщика информационных и вычислительных ресурсов в гетерогенных ЦОД.

Предложенные в диссертации математические модели и методы были внедрены в центрах обработки данных ПАО «МТС», ПАО «Вымпелком», ООО «АЭР - Технологические Решения» (AliExpress Россия), АО «БВТ БАРЬЕР РУС» (международный холдинг BWT).

Результаты диссертационной работы используются в дисциплине «Вычислительные системы» для направления магистратуры 09.04.01 «Информатика и вычислительная техника» (программа «Архитектура информационных систем»).

Соответствие паспорту специальности 2.3.8 Работа выполнена в соответствии со следующими пунктами паспорта специальности 2.3.8 «Информатика и информационные процессы».

- П.1. Разработка компьютерных методов и моделей описания, оценки и оптимизации информационных процессов и ресурсов, а также средств анализа и выявления закономерностей на основе обмена информацией пользователями и возможностей используемого программно-аппаратного обеспечения.
- П.9. Разработка архитектур программно-аппаратных комплексов поддержки цифровых технологий сбора, хранения и передачи информации в инфокоммуникационных системах, в том числе, с использованием «облачных» интернет-технологий и оценка их эффективности.
- П.14. Разработка и исследование принципов организации и функционирования распределенных информационных систем и баз данных, прикладных протоколов информационных сетей, форматов представления данных и языков информационного поиска в распределенных информационных ресурсах.

Основные положения, выносимые на защиту:

1) Модель и метод многокритериального первоначального размещения разнородных виртуальных машин позволяет достичь баланса между множеством противоречивых критериев, таких как энергопотребление, качество обслуживания, использование ресурсов по сравнению с рас-

пространенными на практике алгоритмами «первый подходящий» и «наиболее подходящий».

2) Метод прогнозирования перегрузки и недогрузки серверов на основе метода группового учета аргументов позволяет минимизировать число прямых и обратных миграций виртуальных машин, тем самым повысить стабильность облачных сервисов.
3) Метод расчета длительности живой миграции виртуальных машин для нахождения приближенного аналитического выражения плотности распределения вероятностей позволяет наиболее точно определить размер скользящего окна наблюдения за серверами.
4) Метод многокритериального динамического размещения виртуальных машин по критериям нарушения SLA-соглашений и эффективности использования вычислительных ресурсов, по сравнению с распространенным на практике эвристическим алгоритмом "первый подходящий с упорядочиванием по убыванию позволяет получить в среднем в 3,5 раза лучшие результаты по комбинированному критерию энергопотребления и нарушения SLA-соглашений.

Достоверность результатов обеспечивается за счет корректного применения математического аппарата, программного обеспечения и подтверждается результатами расчетов и имитационного моделирования.

Апробация работы. Основные результаты диссертационной работы докладывались и обсуждались на следующих конференциях: международной научно-технической конференции «Распределенные компьютерные и телекоммуникационные сети: управление, вычисление, связь» (Москва, DCCN-2016, 2017, 2018, 2025); международной отраслевой научно-технической конференции «Технологии информационного общества» (Москва, 2017 гг., 2023 г.); конференции «Телекоммуникационные и вычислительные системы» (Москва, 2017, 2018, 2021 гг.), на международной научно–практическая конференции «Менеджмент качества, транспортная и информационная безопасность, информационные технологии» (Москва, IT&MQ&IS–2018), на конференции 2018 Systems of Signals Generating and Processing in the Field of on Board Communications (Москва, 2018); на 24-й и 28-й международных конференциях Ассоциации открытых инноваций FRUCT (Москва, 2019, 2021 гг.); International Workshop on Model-Driven Organizational and Business Agility (Москва, 2022); 2022 Intelligent Technologies and Electronic Devices in Vehicle and Road Transport Complex

(TIRVED) (Москва, 2023); XXI Международной научно-практической конференции «Новые информационные технологии в образовании (Технологии 1С в цифровой трансформации экономики и социальной сферы)» (Москва, 2022).

Публикации. Основные результаты по теме диссертации изложены в 30 печатных изданиях, 13 из которых изданы в журналах, рекомендованных ВАК, 8 — проиндексированы в базах данных Scopus и Web of Science.

Объем и структура работы. Диссертация состоит из введения, пяти глав, заключения и пяти приложений. Полный объём диссертации составляет 210 страниц, включая 67 рисунков и 25 таблиц. Список литературы содержит 170 наименований.

Глава 1. Анализ современных подходов к распределению ресурсов в гетерогенных центрах обработки данных

1.1 Роль центров обработки данных в цифровой экономике РФ

Центр обработки данных (ЦОД) – это специализированное помещение или здание, предназначенное для размещения сетевого оборудования, хранилищ данных и других компонентов информационной инфраструктуры организаций.

В архитектуре любого ЦОД можно выделить инфокоммуникационную и инженерные подсистемы. Инфокоммуникационная подсистема включает в себя вычислительную и телекоммуникационную подсистемы.

Вычислительная подсистема–это комплекс аппаратного и программного обеспечения, предназначенный для выполнения вычислительных задач.

Телекоммуникационная подсистема позволяет осуществлять взаимосвязь элементов ЦОД, обеспечивать доступ пользователей к ресурсам ЦОД, в случае распределенных структур объединять удаленные ЦОДы в единую систему.

Во всем мире расширяется использование центров обработки данных для предоставления различного рода инфокоммуникационных услуг. Наблюдается рост числа центров обработки данных, модернизация существующих ЦОД, увеличение числа стоек в существующих ЦОД (рис. 1.1).

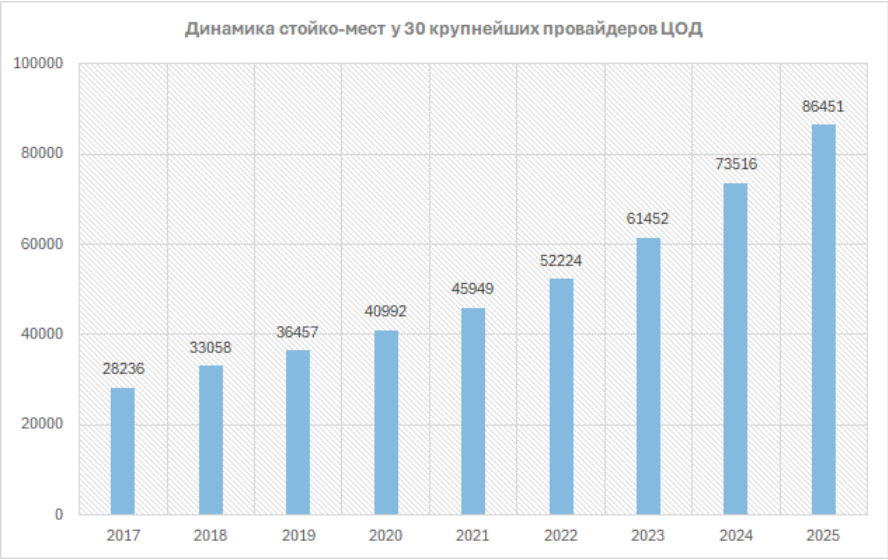

Рисунок 1.1 — Количество стоек у топ-30 провайдеров в 2017–2024 гг. и прогноз CNEWS Analytics на 2025 год

Данная тенденция объясняется ростом популярности различных облачных сервисов, на которых основывается современная цифровая экономика [1]. Государственная программа «Цифровая экономика Российской Федерации» принята в июле 2017 года, где одной из задач является создание инфраструктуры цифровой экономики, в том числе развитие сетей связи и центров обработки данных, без которых невозможно реализовать основные цифровые технологии, перечисленные в программе, а также конкурировать в этих областях на мировом рынке [2]. На период с 2025 по 2030 годы был принят национальный проект «Экономика данных и цифровая трансформация государства», который продолжает национальный проект «Цифровая экономика» [3].

Импортозамещение является первоочередной задачей в современных условиях санкционного давления. Проекты масштабной цифровой трансформации промышленности и социальной сферы, объявленные Правительством РФ, должны реализовываться с применением отечественных программно-аппаратных средств. В этих условиях операторы связи и сервис-провайдеры вынуждены решать задачи совершенствования собственной ИТ-инфраструктуры, что требует разработки новых или адаптации существующих методов управления ресурсами, способных эффективно работать на гетерогенном отечественном оборудовании..

1.2 Облачные вычисления

Облачные вычисления, предоставляющие доступ к ИТ-ресурсам как к коммунальной услуге, позволили создавать новые предприятия в более короткие сроки, облегчили расширение предприятий по всему миру, ускорили темпы научно-технического прогресса и привели к созданию новых видов приложений. Кроме этого, облачные вычисления предоставляют возможность получения более высокого уровня готовности и масштабируемости ИТ-сервисов, в том числе с реализацией схем катастрофоустойчивости в сети распределенных ЦОД.

Облачные вычисления, по определению NIST (Национального института стандартов и технологий США), являются моделью предоставления удобного сетевого доступа к общему пулу вычислительных ресурсов, которые могут быть быстро выделены и освобождены с минимальными усилиями по администриро-

ванию или взаимодействием с провайдером [4]. В соответствии с определением отличительными особенностями облачных вычислений от, например, традиционного хостинга, являются:

– наличие пула виртуализированных вычислительных ресурсов;
– эластичность ресурсов — возможность динамического масштабирования ресурсов в зависимости от нагрузки;
– самообслуживание (создание, удаление, масштабирование виртуальных машин можно проводить без участия администратора ЦОД);
– оплата за ресурсы по факту использования, как оплата за коммунальные услуги.

Существует три модели облачных услуг: SaaS (программное обеспечение как услуга), PaaS (платформа как услуга) и IaaS (инфраструктура как услуга). Их различие определяется тем, какой уровень облачного стека берет на себя провайдер (рис. 1.2).

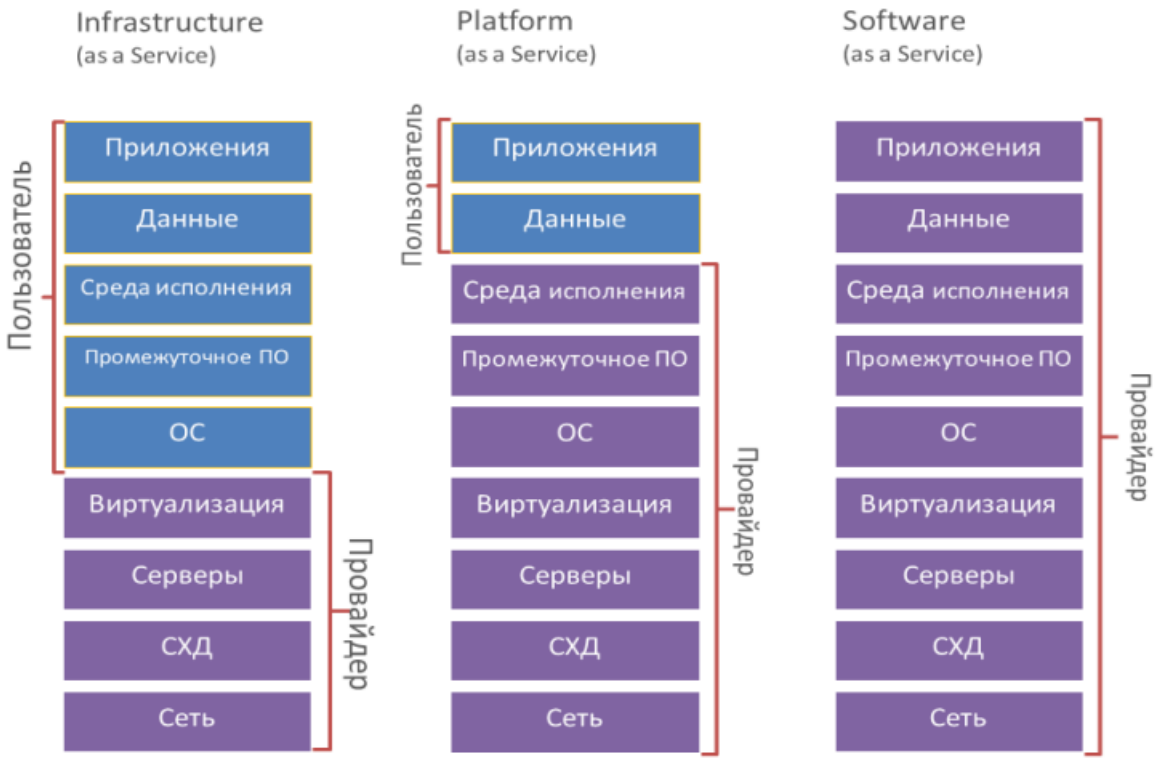

Рисунок 1.2 — Модели облачных вычислений

В модели обслуживания SaaS конечным пользователям предлагается доступ к готовым приложениям, размещенным в облачных ЦОД. Такая модель позволила предприятиям и индивидуальным пользователям получить гибкий доступ к множеству видов программного обеспечения через Интернет, без

необходимости обслуживания собственной ИТ-инфраструктуры и покупки лицензий.

В модели *«платформа как услуга»* (PaaS) пользователям предлагаются платформы для разработки облачных сервисов, включая инструменты для поддержки всего жизненного цикла разрабатываемого ПО. Примерами платформ являются VK Cloud Solutions, Yandex Cloud, SberCloud, Google App Engine, Microsoft Azure и др.

В модели *«инфраструктура как услуга»* (IaaS) провайдер предоставляет клиентам виртуальные машины, ресурсы хранилищ данных и сети. Для управления виртуальными ресурсами дополнительно предоставляются панели управления и интерфейсы прикладного программирования (API). Типичными примерами IaaS являются сервисы VK Cloud Solutions, Yandex Cloud, SberCloud, Selectel, Ростелеком-Cloud, Amazon EC2, Google Compute Engine и др. Уровень IaaS является основой для SaaS и PaaS.

Перечисленные три модели являются базовыми. На их основе появилось ряд других облачных услуг. Перечислим некоторые из них.

GPU as a Service (GPUaaS) представляет собой облачную модель, которая предоставляет пользователям доступ к графическим процессорам (GPU) через Интернет. Эта услуга позволяет компаниям и индивидуальным разработчикам использовать мощные вычислительные ресурсы, которые необходимы для выполнения сложных задач, таких как машинное обучение, рендеринг графики, обработка больших данных и другие ресурсоемкие приложения, без необходимости инвестировать в дорогостоящее оборудование.

В последние годы наблюдается значительное расширение предложения GPUaaS со стороны крупных облачных провайдеров, в том числе в России. В 2023 г. GPUaaS аналитики Global Market Insights оценили в $6,4 млрд. Ожидается, что в период с 2024 по 2032 гг. этот рынок будет расти более чем на 30% в год [5].

Основными факторами, с которым сталкиваются организации, способствующими росту рынка GPUaaS, являются растущий объем данных и требования к более сложным вычислительным задачам. Требования к быстрой обработке и анализу данных делает GPU-процессоры более предпочтительными по сравнению с традиционными CPU. Графические процессоры способны обрабатывать сотни и тысячи потоков параллельно, что позволяет значительно ускорить выполнение задач машинного обучения и аналитики.

Таблица 1 — Сравнение моделей обслуживания CaaS, IaaS, PaaS и FaaS

Характеристика	**CaaS**	**IaaS**	**PaaS**	**FaaS**
Уровень управления	Контейнеры	Виртуальные машины	Платформа	Функции
Масштабирование	Авто/ручное	Ручное	Авто	Авто
Переносимость	Высокая	Средняя	Очень низкая	Очень низкая
Использование	Микросервисы	Любые приложения	Веб-приложения	Событийные задачи, микросервисы

Функция как услуга или бессерверные вычисления FaaS (Function as a Service / Serverless). Это модель облачных вычислений, позволяющая разработчикам запускать отдельные функции (блоки кода) в ответ на события, без необходимости управлять серверами или инфраструктурой.

Контейнер как услуга CaaS (Containers as a Service) – это облачная модель, в которой провайдер предоставляет платформу для развертывания, управления и масштабирования контейнеров (Docker, Kubernetes и др.) без необходимости администрирования инфраструктуры. Сравнение моделей обслуживания CaaS, IaaS, PaaS, FaaS приведено в таблице 1.

Существуют и другие модели предоставления облачных услуг, например, база данных как услуга DBaaS (Database as a Service), хранилище как услуга STaaS (Storage as a Service), виртуальный рабочий стол как услуга DaaS (Desktop as a Service) и др. Каждое направление требует соответствующих исследований. В данной работе будет рассматриваться преимущественно модель IaaS.

По типу собственности различают публичные, частные и гибридные облачные решения.

Частное облако — это облачная инфраструктура, которая принадлежит одной компании, реализованная на своем или на арендованном оборудовании.

Если виртуальная IT-инфраструктура облака принадлежит провайдеру и предоставляется компании-клиенту в аренду, то такое облако называют *публичным.*

Гибридное облако представляет собой сочетание как публичных, так и частных облачных решений. Гибридные облака позволяют использовать преимущества двух моделей, распределяя данные между разными облачными средами. Компания, выбирающая гибридное облако, сама несет ответственность за управление взаимодействием этих двух служб, особенно за безопасность передачи данных между общедоступными и частными облачными сетями.

Использование облачных вычислений по прогнозам будет продолжать расти. По данным IKS Consulting по итогам 2024 года объем российского рынка

Таблица 2 — Сравнение отечественных облачных платформ

Провайдер	**IaaS**	**PaaS**	**AI/ML**	**Kubernetes**	**Аренда GPU**	**Гос. сертификация**
VK Cloud	+	+	+	+	NVIDIA A100, V100	+
Yandex Cloud	+	+	+	+	NVIDIA A100, T4	частично
SberCloud	+	+	+	+	NVIDIA A100, V100	+
Rostelecom Cloud	+	+/—	—	+/—	+	+
Selectel	+	-	-	+/—	NVIDIA A100	+/—
Cloud.ru	+	+/—	+	+	+	+

сервисов IaaS и PaaS увеличился по отношению к 2023-му — на 31,7% и 59% соответственно [6].

Основной тенденцией развития рынка облачных услуг в России эксперты назвали увеличение потребности в гибридных и мультиоблачных средах. Помимо этого, компании видят большой интерес к облачным сервисам искусственного интеллекта [7].

Сравнение отечественных облачных платформ по типу сервиса по данным на 2024 год приведено в таблице 2. В таблице «+» означает полноценную поддержку; «+/—» — ограниченный функционал или требуется дополнительная настройка; «—» означает, что сервис отсутствует или не является ключевым. Данные взяты с официальных сайтов и документации. Как видно из таблицы, провайдеры имеют в своем распоряжении графические процессоры NVIDIA.

Такое широкое использование облачных вычислений в различных областях, а также распространение больших данных, машинного обучения и блокчейна, создает новые проблемы и заставляет пересмотреть модели, которые были разработаны для решения задач масштабируемости, управления ресурсами, надежности и безопасности для реализации облачных вычислительных сред нового поколения [8].

1.3 Современные вызовы и проблемы

1.3.1 Интернет вещей, большие данные и машинное обучение

Доступность Интернета, а также смартфонов, компьютеров и планшетов привело к популярности ИТ-сервисов, таких как социальных сетей, файлооб-

менных сервисов, мессенджеров, хранилищ файлов, электронная коммерция и др., что сказалось на росте Интернет-трафика в мире.

По данным отчета We Are Social and Hootsuite (2025) численность населения мира по состоянию на 2025 год составляет 8,2 миллиарда человек. Мобильными телефонами пользуются 5,78 миллиарда человек, что составляет 70,5% мирового населения. Соответственно наблюдается рост числа пользователей Интернет-сервисов около 5,56 миллиарда человек во всём мире. Уровень проникновения интернета сейчас составляет 67,9% [9].

Рост трафика приводит к росту объемов данных, которые нужно хранить. По данным Минцифры, трафик в российских фиксированных и мобильных сетях за 2024 год вырос на 24,4% — с 151 до 188 Эбайт (рис. 1.3) [10].

Рисунок 1.3 — Динамика роста объемов трафика в РФ

Возрастает генерация потоковых данных с сенсоров и датчиков по мере развертывания Интернета вещей (IoT) и киберфизических систем. Объемы таких данных по прогнозам возрастут во много раз и могут превзойти объемы данных, представленных в общедоступной сети WWW, на предприятиях и в мобильных облаках [8].

Некоторые приложения Интернета вещей и умного города требуют обратной связи, такие как видеоданные с миллионов камер городского наблюдения,

беспилотных автомобилей и дронов. Например, число камер видеонаблюдения в Москве на 2025 г. уже составляет более 255 тысяч [11]. Это делает задержку и пропускную способность между границей сети и облаком для выполнения аналитики существенным ограничением. Поэтому появились *граничные* или *туманные* вычисления, которые начинают дополнять облачные вычисления на граничных устройствах.

Приложения смартфонов, работающие совместно с облачными сервисами, создают, так называемые, мобильные облака. Необходимость выполнения одноранговых вычислений на периферии с облачными сервисами в центрах обработки данных может вызвать потребность в большем количестве региональных центров обработки данных для уменьшения задержки сети и стимулировать рост туманных вычислений [8].

Необходимость обработки «больших данных», генерируемых в последние годы, и увеличение вычислительной мощности привели к развитию области искусственного интеллекта. Разрабатываются новые модели и алгоритмы машинного обучения в различных приложениях. Облачный ЦОД является платформой для размещения сервисов машинного обучения.

Использование графических процессоров (GPU) вместо традиционных CPU стало стандартом в машинном обучении и Big Data благодаря архитектурным преимуществам, позволяющим реализовывать параллельные вычисления. В традиционных CPU обычно имеются от 4 до 64 ядер, оптимизированных для решения последовательных задач. В GPU имеются тысячи ядер (например, NVIDIA A100 — 6912 ядер), работающих параллельно. Также GPU предназначены для матричных вычислений, а обучение нейросетей как раз требует операций с огромными матрицами. Для задач обработки больших данных GPU ускоряют SQL-запросы с помощью GPU-оптимизированных СУБД и графовые вычисления и др. По данным Global Market Insights спрос на GPU-серверы за период 2024-2032 будет ежегодно расти на 30% [12].

1.3.2 Энергопотребление

Рост числа дата-центров (ЦОД) и их мощности делает энергопотребление одной из ключевых проблем отрасли. Международное энергетическое агентство

(IEA) и другие аналитики кардинально пересматривают свои прогнозы по энергопотреблению ЦОДов в сторону резкого увеличения из-за бума искусственного интеллекта, расширение облачных сервисов и роста Интернет-трафика (+30% в год, Cisco [13]. Ожидается, что к 2026 году их совокупное энергопотребление может достигнуть более 1000 ТВт·ч. Это примерно столько же, сколько потребляет вся Япония — одна из крупнейших экономик мира [14].

Для решения проблемы высокого потребления энергии необходимо улучшать физическую инфраструктуру ЦОД и повышать эффективность алгоритмов распределения и управления ресурсами. Для этого необходимы дополнительные исследования [15; 16].

Инфраструктура ЦОД должна быть построена так, чтобы справиться с потенциальной пиковой нагрузкой, которая редко встречается на практике. Это приводит к недостаточной загрузке части ресурсов ЦОД. Серверы имеют узкий диапазон энергопотребления, незагруженный сервер потребляет 70% электроэнергии, как если бы он работал на полную мощность [17]. В целях экономии незагруженные серверы необходимо выключать или переводить их в спящий режим, так как затраты на избыточные мощности весьма значительны и включают в себя расходы на дополнительную мощность системы охлаждения, распределения питания, генераторов, источников бесперебойного питания и так далее.

Инженерные системы потребляют значительную долю электрической энергии ЦОД (рис. 1.4). Большая часть приходится на систему охлаждения, составляющая в среднем 40 % всех затрат на электроэнергию [18].

Для оценки эффективности использования электроэнергии центрами обработки данных используется PUE (Power Usage Effectiveness) — показатель, введенный Green Grid (2007) [19] и включенный в стандарты ISO. Рассчитывается как отношение общего энергопотребления ЦОД к энергии, потребляемой ИТ-оборудованием.

Последние достижения в области проектирования и строительства ЦОД показали значительное улучшение PUE за счет снижения потерь в инженерных подсистемах. Однако повышение энергоэффективности ЦОД требует не только совершенствования тепловых характеристик самого оборудования, но и оптимизации работы системы охлаждения.

Проблему потребления энергии облаками нельзя рассматривать независимо от обеспечиваемого ими качества обслуживания (QoS), поскольку снижение

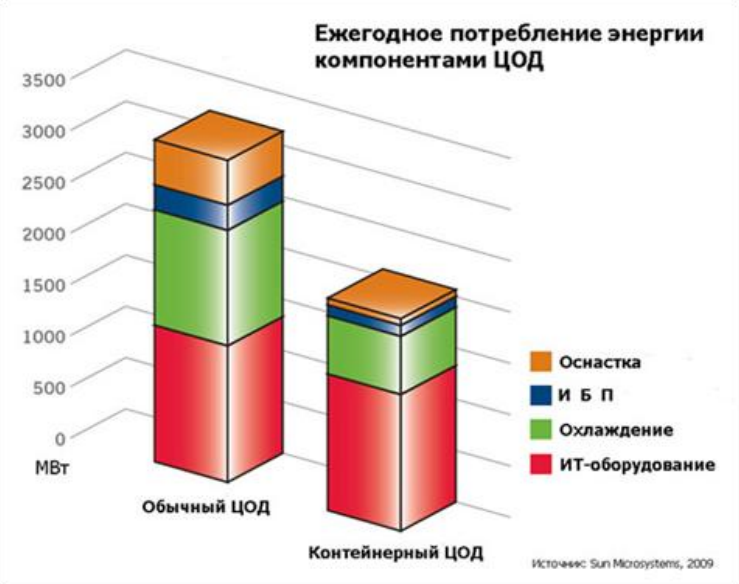

Рисунок 1.4 — Энергопотребление компонентов ЦОД

энергопотребления часто сопровождается снижением качества обслуживания пользователей. Управление ресурсами должно в комплексе учитывать как затраты на энергопотребление, так и потери за нарушение соглашений об уровне сервиса (SLA–Service Level Agreement).

1.3.3 Масштабирование, эластичность и качество обслуживания

Облачные вычисления гарантируют практически неограниченные вычислительные ресурсы по запросу. В этом их отличие от более ранних моделей распределенных вычислений, таких как грид-системы и кластеры. Это дает два преимущества: во-первых, появилась возможность выполнять SLA-соглашения на заданном уровне несмотря на случайные всплески нагрузки, и, во-вторых, пользователям облачных вычислений не нужно делать значительных предварительных вложений в вычислительную инфраструктуру, которая может расти по мере увеличения потребностей в вычислительных ресурсах, оплачивать ресурсы по мере использования.

Эти преимущества облачных вычислений могут быть реализованы благодаря виртуализированной инфраструктуре с эластичными ресурсами и масштабируемыми сервисами.

Виртуализация позволяет гибко масштабировать ресурсы виртуальных машин в соответствии с потребностями приложений. Существуют два подхода к масштабированию: вертикальное и горизонтальное масштабирование.

Вертикальное масштабирование — увеличение производительности каждого компонента системы с целью повышения общей производительности, т.е. можно сделать виртуальную машину более мощной, добавив ей процессорное ядро, память и дисковое пространство. Это самый простой способ масштабирования, так как не требует никаких изменений в прикладных программах, работающих на таких системах, однако в данном случае нельзя добиться многократного масштабирования.

Горизонтальное масштабирование в облачных вычислениях — это объединение виртуальных машин, параллельно выполняющих одну и ту же функцию, в кластер и изменение числа машин в кластере. Этот способ масштабирования требует поддержки со стороны программ. Преимуществом горизонтального масштабирования является то, что можно наращивать ресурсы вычислительной системы в десятки раз.

Кластеризация. Для того чтобы можно было осуществлять миграцию виртуальных машин, большинство платформ виртуализации накладывают ограничение—виртуальные машины должны принадлежать одному кластеру.

На практике на основе платформ виртуализации реализуются две разновидности кластеров: отказоустойчивый кластер и кластер распределенных ресурсов. В терминологии VMware vSphere две разновидности кластеров называются: High-availability кластер (HA) и Distributed Resource Scheduler кластер (DRS).

В конечном итоге масштабируемость облака ограничена степенью масштабирования отдельных компонентов, а именно вычислительного комплекса, хранилищ данных и сетевых соединений.

Таким образом, масштабируемость и эластичность дают возможность повышения производительности приложений облачных вычислений экономически эффективным способом, которые еще предстоит использовать в полной мере. Для этого требуются эффективные механизмы управления ресурсами и планирования.

1.3.4 Надежность и безопасность

Надежность еще одна важная проблема в облачных вычислениях. Облачные ЦОД состоят из взаимосвязанных и взаимозависимых систем. Из-за своего масштаба, сложности и взаимозависимости системы облачных вычислений сталкиваются с различными угрозами, связанными с надежностью, такими как отказы оборудования, сети, отказы в обслуживании из-за отсутствия ресурсов по истечении времени ожидания, ошибок в программном обеспечении и др. Некоторые из этих отказов могут разрушительно повлиять на работу системы, вызывая, таким образом, критические отказы. Более того, может запускаться каскад отказов, приводящий к крупномасштабным сбоям в обслуживании с далеко идущими последствиями [20]. Разработка облачных сервисов с гарантированной производительностью и устойчивостью для борьбы со всеми типами независимых и коррелированных отказов с учетом энергоэффективности облачных систем является одной из актуальных исследовательских задач.

Безопасность – главная проблема в системах ИКТ, и облачные вычисления не являются исключением. Наблюдается ежегодный прирост хакерских атак. По данным Check Point Research (январь-июнь 2024) глобальное число атак выросло на 15-20% по сравнению с тем же периодом 2023 года.

Методы и средства обеспечения безопасности облака могут различаться в зависимости от типа облачной среды–частные, публичные, гибридные облака, мультиоблачные среды. В целом, все меры безопасности облака направлены на защиту следующих компонентов: физические сети и инженерные системы, носители данных, серверы, платформы виртуализации, операционные системы, промежуточное программное обеспечение, среды выполнения, данные, приложения, оборудование конечного пользователя.

В настоящее время сформировалась тенденция к использованию методов машинного обучения в информационной безопасности для анализа угроз, анализа атак, распространения вирусов и корреляции данных [21].

1.4 Архитектура гетерогенных облачных центров обработки данных

1.4.1 Требования к гетерогенным облачным центрам обработки данных

Особенности облачных вычислений, такие как высокая надежность, эластичность ресурсов и предоставление ресурсов по требованию предъявляют высокие требования к инфокоммуникационной инфраструктуре центров обработки данных, начиная от физических серверов, сетей и систем хранения данных и заканчивая технологиями виртуализации и платформами разработки приложений.

Инфраструктура и архитектурные решения центров обработки данных, ориентированных на облачные услуги, принципиально не отличаются от традиционных виртуализированных ЦОД. К характерным чертам облачных ЦОД следует отнести использование типовых серверов стандартной архитектуры, систем хранения с горизонтальной масштабируемостью, применение технологий виртуализации ресурсов и специальных программных средств управления «облаками», однако на техническом уровне состав подсистем остается идентичным. Эволюция к облачным ЦОД в то же время выдвигает более жесткие требования к производительности большинства инфраструктурных элементов.

Все решения для облачных ЦОД должны поддерживать работоспособность в условиях высокоплотной IT-нагрузки. Например, виртуальные машины содержат, помимо информации и приложений, еще и технические данные, что приводит к увеличению нагрузки на системы хранения. Кратно растет объем информационных потоков внутри ЦОД — от миграции ВМ до взаимного обмена данными между оборудованием облачного ЦОД, что приводит к «перегруженности» внутренней кабельной системы при недостаточном быстродействии ее элементов.

На инженерном уровне решаются задачи обеспечения функционирования основных систем ЦОД в штатном режиме. В него входят подсистемы бесперебойного энергоснабжения, поддержания температуры и влажности в помещении, кондиционирования, пожарной сигнализации и пожаротушения.

Автоматизированные системы мониторинга и управления – важный и необходимый элемент для работы ЦОД. Внедрение облачных технологий требует автоматизации традиционных ЦОД, причем это касается разных уровней и задач: сетей, серверов, систем хранения, безопасности, настроек конфигурации, работы с активами и т.д. С точки зрения аппаратуры в облачных ЦОД могут использоваться стандартные современные компоненты и решения, хотя на рынке и существует отдельное «облачное» направление. Все это оборудование должно быть полностью автоматизировано – необходимо соответствующее программное обеспечение, которое позволяет регулировать все операции и согласованно управлять ими с рабочего места администратора.

Облачный подход к архитектуре ЦОД, в сочетании с развитыми программными средствами управления и широким использованием типовых конфигураций, с одной стороны снижает себестоимость услуг, а с другой стороны позволяет быстро наращивать ресурсы без увеличения сложности эксплуатации.

На концептуальном уровне требования к гетерогенным облачным ЦОД можно обобщить следующим образом:

- Гибкая, масштабируемая, программно-определяемая архитектура.
- Ориентация на сервисы.
- Динамическое распределение ресурсов и балансировка нагрузки.
- Построение на основе SLA.
- Наличие доступного интерфейса прикладного программирования (API) для управления гетерогенными ресурсами.
- Энергоэффективность.
- Единая система мониторинга.

1.4.2 Телекоммуникационная подсистема

Телекоммуникационная подсистема облачного ЦОД — это комплекс сетевых и коммуникационных решений, обеспечивающих высокоскоростное, надежное и безопасное взаимодействие между серверами, системами хранения и внешними сетями.

Телекоммуникационная подсистема позволяет осуществлять взаимосвязь элементов ЦОД, обеспечивать доступ пользователей к ресурсам ЦОД, в случае распределенных структур объединять удаленные ЦОДы в единую систему. Таким образом, телекоммуникационная подсистема является связующим звеном ЦОД и включает в себя структурированные кабельные системы, активное сетевое оборудование, каналы связи, в том числе магистральные.

Широкое применение в облачных ЦОД получили «чистый» Ethernet (особенно для IP-трафика) и InfiniBand (для HPC/AI-кластеров).

ЦОД традиционно имеет трехуровневую архитектуру (рис. 1.5), где каждый уровень имеет определенные функции по распределению и управлению трафиком [22].

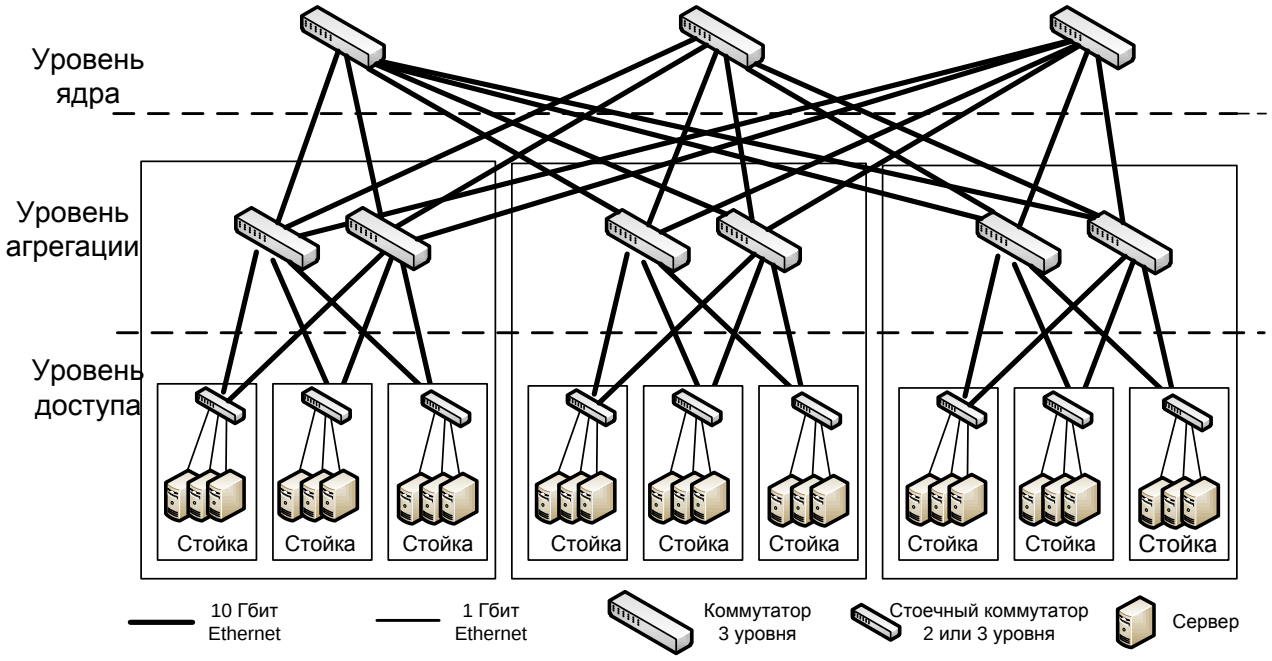

Рисунок 1.5 — Трехуровневая сетевая архитектура ЦОД

На нижнем уровне, известном как уровень доступа, каждый физический сервер, подключен к одному или двум (для повышения надежности) коммутаторам доступа. На уровне доступа каждый коммутатор доступа соединен с одним или двумя коммутаторами уровня агрегации. В свою очередь каждый коммутатор уровня агрегации связан с более чем одним коммутатором ядра. На данном уровне поддерживаются технологии виртуализации (VXLAN, EVPN).

Коммутаторы доступа обеспечивают связь между серверами одной стойки и возможность выхода на более высокий уровень агрегации. Коммутаторы

уровня агрегации соединяют коммутаторы уровня доступа и локализуют трафик между серверами в пределах нескольких соседних стоек. На коммутаторах настраиваются межсетевые экраны, балансировщики нагрузки.

Коммутаторы ядра соединяют коммутаторы уровня агрегации таким образом, что существует связь между каждым сервером в ЦОД и возможность выхода за пределы центра обработки данных.

Для всех облачных приложений, в частности для тех, которые используют параллельные вычисления, коммуникационные задержки, налагаемые сетевыми технологиями и протоколами в ЦОД, ограничивают общую производительность.

Существуют два критичных аспекта. Первым является сетевая топология, которая значительно влияет на производительность и на отказоустойчивость. Второй ключевой аспект ближе непосредственно к управлению ресурсами: как обеспечивать предсказуемую задержку и пропускную способность в сетях ЦОД с учетом различных видов трафика.

Недостатком традиционной трехуровневой архитектуры является то, что она больше предназначена для интенсивного обмена серверов ЦОД с внешними клиентами/интернетом. Однако с распространение распределенных архитектур приложений, таких как микросервисы, распределенные СУБД, машинное обучение, анализ данных в реальном времени создают большой внутренний трафик между серверами, стойками или модулями внутри ЦОД. Это приводит у появлению узких мест на уровне агрегации.

Поэтому на смену трехуровневой топологии пришла плоская топология Spine-Leaf, включающая в себя высокоскоростные маршрутизаторы Spine (400/800 Гбит/с), образующие ядро сети, и Leaf-коммутаторы, к которым подключаются серверы (аплинки к Spine — 100/200 Гбит/с) [23].

Внутри самого ЦОД был осуществлен шаг в сторону внедрения дифференциации сервисов с помощью политик QoS. Реализуется это использованием технологий виртуализации сетей, в частности, программно-конфигурируемых сетей (SDN).

Помимо SDN разработана концепция NFV, которая предполагает замещение сетевых устройств стандартизированными высокопроизводительными серверами, коммутаторами и системами хранения данных с реализацией сетевых функций (служб) программным обеспечением. Основное преимущество

использования NFV заключается в том, что поставщики облачных услуг могут запускать новые услуги сетевых функций более гибким и гибким способом.

SDN и NFV могут служить строительными блоками облаков следующего поколения, облегчая решение таких проблем, как устойчивость, взаимосвязанные облака и безопасность.

1.4.3 Виртуализация серверов

Облака используют различные технологии виртуализации (виртуализацию серверов, сетей и хранилищ), чтобы предоставить пользователям дополнительный уровень абстракции, в виде единой вычислительной платформы, скрывая неоднородность оборудования, фактические места размещения и внутренние сложности управления. В целом, виртуализация позволяет повысить коэффициент использования ресурсов, организовать динамическое распределение ресурсов, минимизировать энергопотребление, а также повысить масштабируемость, доступность и надежность облачных ресурсов и сервисов. За счёт широкого использования технологий виртуализации облачные ЦОД требуют меньшей площади. С архитектурной точки зрения, можно выделить следующие подходы к виртуализации, используемые в ЦОД:

- **Архитектура на основе гипервизора**: уровень виртуализации, называемый гипервизором, устанавливается и запускается на «голом» оборудовании, как, например, VMWare ESXi, что позволяет сохранить полный контроль над вычислительной системой. Программное обеспечение, которое размещает и управляет виртуальными машинами называется монитором виртуальных машин (MBM).
- **Контейнерная виртуализация**. С появлением Docker [24] контейнерные технологии вызвали широкий интерес как в академических кругах, так и в промышленности. Контейнеры обеспечивают легкую среду для развертывания приложений; они представляют собой автономные модули, которые вместе упаковывают программное обеспечение и его зависимости. Как и в случае с виртуальными машинами, контейнеры позволяют совместно использовать ресурсы одного вычислительного

Таблица 3 — Сравнение виртуальных машин и контейнеров

Параметр	Виртуальная машина	Контейнер
Гостевая ОС	Полноценная ОС	Общее ядро ОС хоста
Размер	ГБ	МБ
Запуск	Минуты	Секунды
Потребление ресурсов	Высокое	Значительно ниже
Плотность	Мало экземпляров на хост	Много экземпляров на хост
Производительность	Накладные расходы от гипервизора	Практически нативная
Уровень изоляции	Сильная (аппаратная)	Слабая (на уровне процесса)
Поверхность атаки	Гипервизор (менее уязвим)	Ядро хоста (больше и уязвимее).
Последствия взлома	Возможно, одна ВМ	Один контейнер

узла, позволяя приложениям работать как изолированные процессы пользовательского пространства.

Рост популярности контейнеров можно объяснить двумя основными особенностями. Во-первых, они запускаются очень быстро, и время их запуска составляет менее секунды. Во-вторых, контейнеры занимают мало памяти и потребляют очень мало ресурсов. По сравнению с виртуальными машинами, использование контейнеров не только улучшает производительность приложений, но также позволяет хосту поддерживать больше экземпляров одновременно. Сравнение виртуальных машин и контейнеров приведено в таблице 3. Несмотря на эти преимущества, все еще есть недостатки и проблемы, которые необходимо решать. Во-первых, из-за совместного использования ядра, изоляция и безопасность контейнеров слабее, чем в виртуальных машинах [25].

Во-вторых, это оптимизация производительности контейнеров в зависимости от видов приложений. В-третьих, это управление кластерами контейнеров на основе требований пользователей к QoS. Системы управления кластером контейнеров, такие как Kubernetes, Mesos и Swarm, становятся основным программным обеспечением платформы облачных вычислений.

Виртуализация графического процессора GPU — это технология, обеспечивающая разделение физических ресурсов одного физического GPU между несколькими ВМ или контейнерами. Традиционно каждая виртуальная машина была привязана к отдельному физическому GPU, что приводило к ограничению масштабируемости и неэффективному использованию мощностей. Однако с развитием технологий виртуализации появилась возможность делить ресурсы одного GPU между множеством виртуальных машин, предо-

ставляя каждой ВМ лишь необходимую долю ресурсов видеокарты. Различают следующие виды виртуализации [26]:

- **PCIe passthrough**: физическое подключение GPU непосредственно к виртуальной машине, без дополнительной виртуализации. Этот метод предлагает наивысшую производительность, однако миграция возможна только на аналогичные хосты с аналогичным оборудованием.
- **Virtualized GPUs (vGPU)**: логическая виртуализация GPU, при которой физическое устройство делится на несколько независимых виртуальных GPU. Данный метод наиболее универсален и лучше всего подходит для массовых миграций, хотя и обладает меньшей производительностью по сравнению с PCIe passthrough.
- **SR-IOV – Single Root I/O Virtualization**: деление одного физического GPU на несколько виртуальных экземпляров, каждый из которых доступен отдельной виртуальной машине. Метод менее распространён для миграции, поскольку требует тщательной настройки совместимости.

Примерами реализаций виртуализации GPU являются

- **NVIDIA vGPU**: технология от NVIDIA, предназначенная для разделения ресурсов GPU между множеством виртуальных машин. Включает поддержку для виртуализации рабочих станций и HPC-кластеров.
- **NVIDIA MIG (Multi-Instance GPU)**: аппаратное разделение GPU A100/H100 на изолированные экземпляры. Виртуальным машинам гарантированы пропускная способность и память.
- **AMD MxGPU**: технология аналогичная NVIDIA vGPU от AMD, позволяющая разделить физические ресурсы GPU между виртуальными машинами.
- **Intel AMT GPU**: решение Intel для виртуализации интегрированных графических карт, которое поддерживает совместное использование ресурсов одним физическим устройством между несколькими виртуальными системами.

1.4.4 Миграции виртуальных машин

Важнейшей функцией платформ виртуализации является поддержка миграции виртуальных машин.

Под живой миграцией (Live Migration) понимают перемещение активной ВМ с одного физического сервера на другой без остановки ее работы. Сервисы, выполняемые внутри ВМ, испытывают лишь кратковременную задержку, как правило, незаметную для пользователей. В процессе миграции по сети копируется состояние процессора, содержимое памяти и другая контекстная информация, при этом все характеристики ВМ (сетевые настройки, данные на дисках, состояние ОС и приложений) остаются полностью сохраненными.

Другим вариантом является статическая миграция (Static Migration), называемая также «холодной» (Cold Migration) [27]. Ее отличительная особенность — необходимость полного отключения виртуальной машины на время переноса. После завершения работы ВМ ее образ, состояние и конфигурация перемещаются на целевой сервер. Это гораздо более быстрый и простой способ переноса виртуальной машины с незначительным увеличением трафика в сети; однако статическая миграция виртуальной машины требует значительно большего времени простоя сервиса по сравнению с живой миграцией. В связи с этим использование живой миграции более распространено в коммерческих ЦОД.

Живая миграция в виртуализированных ЦОД используется для консолидации ВМ, балансировки нагрузки, предотвращения перегрева серверов, а также для проведения обслуживания (обновление ПО, ремонт) без остановки сервисов.

В первом случае, когда есть большое количество недозагруженных физических серверов, миграция позволяет уменьшить фрагментацию серверов для возможного размещения на освободившихся «тяжелых» вычислительных задач, а также выключить неиспользуемые серверы в целях уменьшения энергопотребления. Для балансировки нагрузки миграция будет срабатывать при неравномерной нагрузке на физические серверы, перемещая ВМ с высоконагруженных серверов на менее нагруженные. Это позволяет поддерживать одинаковый уровень сервиса для виртуальных машин одного класса облуживания, а также избежать перегрева сервера, поддерживать температуру в машинном зале на одном уровне, что сокращает расходы на охлаждение [28; 29]

Процесс живой миграции виртуальной машины значительно сложнее, чем просто перенос страниц памяти виртуальной машины с исходного сервера на хост назначения. Во время процесса копирования на запущенной виртуальной машине могут выполняться команды записи в страницы памяти, поэтому новые «грязные» страницы памяти также должны быть скопированы на хост назначения. Таким образом, в целях обеспечения согласованного состояния виртуальных машин, участвующих в миграции, процесс копирования всех грязных страниц должен быть осуществлен до завершения процесса миграции. Кроме того, каждая активная ВМ имеет собственную долю доступных ей физических ресурсов, таких как системы хранения данных, сеть, и устройства ввода/вывода. Поэтому в процессе миграции должно гарантироваться, что соответствующие физические ресурсы на хосте назначения присоединены к перенесенной виртуальной машине.

Даже при использовании технологии живой миграции полностью избежать простоев не удаётся. Как показано в работе [30], время недоступности сервиса варьируется от 60 миллисекунд до 3 секунд и определяется интенсивностью работы приложения с памятью. Следовательно, миграция виртуальных машин может оказывать влияние на выполнение условий SLA, что особенно важно учитывать в облачных и гибридных инфраструктурах.

Миграция виртуальных машин с GPU имеет свои особенности и сложности, отличающих этот процесс от стандартной миграции ВМ с CPU. Это связано со следующим:

- **Большой объём передаваемых данных**. Одним из главных факторов, влияющих на миграцию ВМ с GPU, является большой объем данных, необходимых для передачи состояния устройства. Современный GPU способен хранить значительное количество информации внутреннего состояния (контекст исполнения, данные в видеопамяти, дескрипторы ресурсов). В зависимости от конфигурации состояние GPU может занимать десятки гигабайт памяти. Это существенно увеличивает продолжительность процесса миграции и требует высокоскоростных каналов связи между хостами.
- **Высокие требования к синхронизации**. Особенно важным аспектом миграции ВМ с GPU являются высокие требования к синхронизации. Графический процессор активно взаимодействует с центральным процессором, памятью и приложениями, работающими на виртуальной

машине. Например, в случае вычислений общего назначения точность результатов зависит от точного согласования состояний обоих устройств. Любые рассинхронизации могут привести к потере точности или даже отказу всей вычислительной среды.

– **Тип виртуализации GPU**. Тип используемого метода виртуализации GPU определяет возможные ограничения и сложности при переносе ВМ.

1.5 Обзор работ по повышению эффективности управления ресурсами инфокоммуникационной системы центров обработки данных

1.5.1 Методы первоначального размещения виртуальных машин в гетерогенных центрах обработки данных

Масштаб современных облачных ЦОД растет быстро, и на сегодняшний день в них содержатся от десятков до сотен тысяч вычислительных устройств и устройств хранения.

Современные провайдеры инфраструктурных услуг (IaaS) используют либо статическое выделение ресурсов (ВМ получают фиксированные квоты физических мощностей), либо динамические стратегии управления, позволяющие перераспределять нагрузку между серверами в ответ на ее колебания с помощью миграции ВМ и методов балансировки [8].

Эти политики могут быть либо реактивными, либо упреждающими и обычно основываются на знании требований к ресурсам виртуальных машин, предоставляемых пользователем или оцениваемых с использованием данных мониторинга и прогнозирования.

Статическое (первоначальное) размещение предполагает, что в исходном состоянии все физические серверы свободны и обладают известной ресурсной емкостью, а размещению подлежит множество виртуальных машин с заданными требованиями к ресурсам. Данный подход требует априорной информации обо всех рабочих нагрузках и их потребностях. Такие методы особенно востре-

бованы в сценариях первичного развертывания ВМ либо при миграции целой группы виртуальных машин между центрами обработки данных.

Традиционные стратегии размещения виртуальных машин часто основаны на эвристических алгоритмах, цель которых — обеспечить удовлетворительные решения за приемлемое время вычислений. Примерами служат алгоритмы «самый подходящий» (Best Fit – BF), «первый подходящий» (First Fit – FF), «самый неподходящий» (Worst Fit – WF). Алгоритм BF размещает виртуальные машины на хостах с наименьшим количеством оставшихся ресурсов, тем самым минимизируя неиспользуемое пространство. Алгоритм FF быстро размещает виртуальные машины на первом доступном хосте с достаточным количеством ресурсов, жертвуя потенциальной оптимизацией ресурсов ради скорости. Напротив, WF стремится к равномерному распределению нагрузки, размещая виртуальные машины на хостах с наибольшим доступным ресурсом.

В настоящее время опубликовано значительное число работ, в которых предлагаются более эффективные по сравнению с классическими методы решения задач первичного и оперативного размещения ВМ в облачной инфраструктуре [31; 32]. Во многих работах задача размещения ВМ сводится к задаче об упаковке в контейнеры. Примерами таких работ являются [29; 33—37].

В ряде работ для решения задач статического размещения виртуальных машин используются генетические и муравьиные алгоритмы (ACO) [35; 36; 38; 39]. Такие подходы не гарантируют оптимальное решение, но позволяют достаточно быстро получить приемлемое решение.

Стратегии, учитывающие энергосбережение, направлены на снижение энергопотребления ЦОД путем размещения ВМ на меньшем количестве хостов, что позволяет отключать неиспользуемые серверы. Такие методы, как динамическое масштабирование напряжения и частоты (DVFS), регулируют энергопотребление в соответствии с нагрузкой сервера [40; 41]. Стратегии консолидации ВМ мигрируют виртуальные машины для уменьшения количества активных серверов, обеспечивая экономию энергии [35; 38; 39].

Другим распространенным критерием является балансировка нагрузки, что позволяет предотвратить перегрузку отдельных хостов. Распространенные методы включают в себя циклический алгоритм (Round Robin), взвешенный циклический алгоритм (Weighted Round Robin) и алгоритм наименьшего количества подключений (Least Connections). «Продвинутые» методы отслеживают

производительность хостов и динамически корректируют размещение виртуальных машин для поддержания сбалансированной нагрузки [32; 42].

Стратегии размещения виртуальных машин, основанные на соглашении об уровне обслуживания (SLA), учитывают конкретные требования к качеству обслуживания (QoS) каждой виртуальной машины, такие как задержка, пропускная способность и надежность, размещая виртуальные машины соответствующим образом для удовлетворения этих критериев [43; 44].

Методы многокритериальной оптимизации позволяют сбалансировать эти критерии, стремясь как к производительности, так и к соблюдению SLA. Проблемы многокритериального размещения рассматривались в [45—47]. Для решения многокритериальных задач чаще всего используются методы формирования обобщенного критерия. Однако встречаются и другие методы многокритериального принятия решений, например, метод анализа иерархий [48; 49], который обычно не подходит для использования в реальном времени. Подробный обзор существующих алгоритмов многокритериального размещения виртуальных машин можно найти в работе [50].

Данные работы не учитывают гетерогенность ЦОД. Предоставление графического процессора и памяти ВМ требует учета дополнительных параметров и ограничений [51].

На раннем этапе GPU использовались в режиме прямого проброса (passthrough), что не позволяло делить ресурсы одной карты между несколькими ВМ. Появление технологий, таких как NVIDIA GRID, позволило организовать совместный доступ, однако они имели ограничения, например, по мультиплексированию видеопамяти [52]. С течением времени ситуация изменилась: появились технологии, позволяющие организовать совместное использование GPU несколькими ВМ [53]. Одним из таких решений стала технология NVIDIA Multi-Instance GPU (MIG), которая предоставляет возможность одновременной работы нескольких ВМ с разделением ресурсов GPU, включая изолированные разделы памяти [54].

Простая эвристика First Fit, реализованная, в частности, в VMware vSphere, широко применяется для выбора сервера назначения при размещении ВМ с GPU. Тем не менее, как показано в [51], в условиях активного создания и удаления виртуальных машин такой подход теряет эффективность, приводя к нерациональному использованию GPU и видеопамяти [55]. Согласно выводам [56], фрагментация GPU-ресурсов представляет собой одну из главных проблем

при эксплуатации крупномасштабных GPU-кластеров, задействованных в обучении нейросетей.

В работе [57] представлен метод размещения виртуальных машин для GPU NVIDIA GRID, который учитывает тип рабочей нагрузки на ВМ, надежность и доступность памяти видеокарты (GDDRAM). В [58] предложен симулятор для сравнения различных вариантов размещения ВМ по стоимости. В [59] была сформулирована целочисленная линейная задача для минимизации физической загрузки GPU в облаке. Плотная нейронная сеть DNN использовалась для размещения виртуальных машин с поддержкой GPU в NVIDIA GRID в работе [60].

Оптимизация разделов NVIDIA MIG также рассматривается в [60—63]. Один из подходов к дефрагментации памяти был предложен в [55]. В данной статье также предложена формулировка многокритериальной задачи оптимизации для размещения виртуальных машин в виде задачи целочисленного линейного программирования. Однако формулировка данной задачи слишком сложна для прямого решения.

Таким образом, необходимо разработать специализированный метод многокритериального размещения виртуальных машин, позволяющий минимизировать количество задействованных физических узлов для снижения энергопотребления, обеспечить равномерную загрузку ресурсов серверов, учесть аппаратные ограничения серверов: наличие GPU, специфичные архитектуры, получив решение за приемлемой время.

1.5.2 Динамическое управление виртуальными машинами в гетерогенных средах

Модель предоставления ресурсов по требованию предполагает возможность динамического создания, удаления и масштабирования виртуальных машин по мере необходимости. Однако следствием такой гибкости становится фрагментация ресурсов на физических серверах и, как результат, снижение эффективности их использования.

Методы управления ресурсами важны не только для провайдеров IaaS, но и для поставщиков PaaS и SaaS, поскольку они помогают управлять типом

и количеством ресурсов, выделенных для распределенных приложений, контейнеров, веб-сервисов и микросервисов.

Размещение ВМ, учитывающее текущее состояние инфраструктуры, относится к классу задач динамического управления. Для перемещения виртуальных машин между физическими серверами применяются технологии статической или живой миграции.

Теоретически для этих целей возможно периодическое использование алгоритмов статического размещения. Однако такой подход игнорирует накладные расходы, связанные с самим процессом перераспределения ресурсов.

На практике процесс динамического управления ресурсами, как правило, декомпозируется на последовательность этапов [64]: мониторинг состояния серверов для выявления критических ситуаций (перегрузка/недогрузка); выбор виртуальных машин, подлежащих миграции; определение целевых физических серверов для размещения отобранных ВМ.

Поскольку современные приложения демонстрируют высокую изменчивость нагрузки, то для обеспечения качества обслуживания (QoS), определяемое SLA, необходимо тщательно прогнозировать нагрузку. В противном случае постоянное перераспределение ресурсов может повлиять на стабильность работы сервисов и облака в целом.

В первых работах, посвященных динамическому размещению виртуальных машин [65; 66], основная цель формулировалась как минимизация количества используемых физических серверов при одновременном сокращении числа миграций.

Среди практических реализаций следует отметить решения компании VMware, описанные в [67]. Первый продукт — Distributed Resource Scheduler (DRS) — обеспечивает балансировку нагрузки в кластере физических хостов за счет автоматической миграции виртуальных машин. Второй продукт — Distributed Power Management (DPM) — расширяет функциональность DRS, добавляя механизмы управления энергопотреблением. DPM анализирует загрузку CPU и памяти и принимает решения о выключении хостов при низкой нагрузке или их включении при возрастании нагрузки для поддержания требуемого качества обслуживания приложений.

Данные программные продукты используют пороговые алгоритмы для определения того, когда инициировать миграцию. Для каждого сервера задаются два порога:

- **Верхний порог**. Если загрузка сервера превышает это значение, он считается перегруженным. С него нужно мигрировать часть ВМ, чтобы избежать нарушения SLA.
- **Нижний порог**. Если загрузка сервера падает ниже этого значения, он считается недогруженным. Все его ВМ нужно мигрировать на другие серверы, чтобы выключить сам сервер.

Также пороги могут быть единичными фиксированными или множественными, что позволяет использовать разные пороги для разных серверов и точнее управлять консолидацией.

Со временем исследования выявили ограничения этого подхода, и фокус сместился на учет следующих критериев [68]:

- **Качество обслуживания (QoS) и SLA**: простая «упаковка» ВМ на меньшее число серверов ведет к конкуренции за ресурсы (CPU, кэш, память, устройства ввода/вывода). Это вызывает нарушение SLA-соглашений. Современные алгоритмы учитывают не только загрузку CPU, но и другие метрики [69; 70].
- **Устойчивость**: простые пороговые алгоритмы могли вызывать обратные миграции. ВМ мигрирует с сервера A на B, из-за чего сервер A становится недогруженным, и следующая миграция пытается вернуть ВМ обратно. Современные методы используют прогнозирование нагрузки, чтобы избежать этого (см. параграф 1.5.4).
- **Гетерогенность**: ранние методы часто предполагали однородные серверы. В современных дата-центрах серверы сильно различаются по поколениям CPU, объему памяти и т.д.
- **Многокритериальная оптимизация**: помимо энергопотребления и миграций, также учитываются:
  - **Связанность ВМ**: Размещение интенсивно обменивающихся друг с другом ВМ должно быть ближе, желательно на одном сервере или в одном коммутаторе агрегации, чтобы снизить нагрузку на сеть [71; 72].
  - **Надежность**: не следует размещать все критичные ВМ на одном сервере.
  - **Тепловыделение**: учет температуры и эффективности охлаждения в разных частях ЦОД [28; 29].

Таким образом, учет нескольких критериев, таких как энергопотребление, выполнение SLA-соглашений, равномерная температура процессоров и более полное использование ресурсов серверов, при динамическом распределении является актуальной задачей. Учет данных критериев и детальная проработка последовательности действий по оптимальному размещению ВМ позволит повысить эффективность управления ресурсами и качество облачных сервисов.

Время, необходимое для решения оптимизационной задачи, является одним из основных факторов, влияющих на качество принятия решений в реальном времени. Один цикл работы контроллера длится несколько минут (2-6 минут). Задержки в принятии решений могут привести к значительным штрафам за нарушение соглашений SLA и дополнительным эксплуатационным расходам. Нерегулируемое увеличение задержек сделает невозможным внедрение инновационного высокодоходного облака.

Проблема планирования и управления должна решаться комплексно, необходимо разработать методику управления ресурсами и выработать рекомендации по практической реализации данной методики.

1.5.3 Оптимизация процесса миграции виртуальных машин

Появление технологии миграции виртуальных машин (ВМ) решает проблемы перегрузки серверов и снижения производительности, позволяя перемещать ВМ между серверами в пределах одного ЦОД или между различными центрами обработки данных. При миграции ВМ гипервизор разгружает перегруженные серверы, перемещая их рабочие нагрузки на недозагруженные серверы или серверы с нормальной нагрузкой.

Однако миграция ВМ может потребовать дополнительных ресурсов (таких как электроэнергия, пропускная способность сети и вычислительные ресурсы) и может негативно влиять на приложения внутри перемещаемой ВМ до завершения процесса миграции. Поэтому для поддержания производительности приложений крайне важно завершать процесс миграции за минимальное время, используя при этом минимум сетевых и серверных ресурсов [73].

Миграция ВМ в пределах локальной сети управляется значительно проще по сравнению с миграцией в глобальной сети. Причина в том, что в локальной

сети не требуется миграция хранилищ данных благодаря сетевым хранилищам (NAS). В то время как миграция ВМ через глобальную сеть сталкивается с рядом проблем, таких как миграция хранилищ данных, сетевая перегрузка, ограниченная пропускная способность, ненадёжные каналы связи. Все эти факторы затягивают общий процесс миграции ВМ.

Для решения данных проблем миграции в ЦОД миграция исследовалась в различных работах. В обзорной статье [74] выделены следующие схемы миграции ВМ:

– традиционные схемы миграции ВМ;
– схемы миграции на основе искусственного интеллекта,

в контексте балансировки нагрузки, энергоэффективности, снижения нарушений SLA-соглашений, а также использования сети и пропускной способности.

В работе [75] представлен комплексный обзор по живой миграции виртуальных машин. В работе [76] предоставлен обзор методов прогнозирования, основанных на эффективных способах миграции ВМ, где авторы выявили ключевые проблемы в миграциях на основе прогноза и классифицировали по типу используемых алгоритмов прогнозирования.

Авторы работы [77] исследовали схемы консолидации ВМ и представили различные метрики производительности, методы оптимизации, их цели и подходы к оценке в облачных вычислениях. Кроме того, они также рассмотрели программные и аппаратные метрики, а также архитектуру консолидации ВМ в системах облачных вычислений.

В целом, можно выявить следующие проблемы миграции ВМ для эффективного управления ресурсами ЦОД.

– Применение методов ИИ (машинное, глубокое, федеративное обучение) позволяет прогнозировать загрузку ресурсов (ЦПУ, ОЗУ, сеть) и принимать эффективные решения о миграции: выбор ВМ, момент миграции и целевой хост. Однако алгоритмы ИИ требуют значительных вычислительных и энергетических ресурсов, что при их нехватке может привести к задержкам и нарушению SLA. Требуется разработать схемы миграции на основе ИИ, которые обеспечивают точное прогнозирование в реальном времени с минимальным потреблением ресурсов.
– Проблемы миграции ВМ через глобальные сети: большие задержки, ограниченная пропускная способность, неоднородность сетей и потери пакетов увеличивают время миграции и риск нарушения SLA.

– Проблема доступности ресурсов: для начала миграции необходимы свободные ресурсы на исходном хосте (ЦПУ, I/O, сеть), а для эффективной работы после миграции — на целевом. Заблаговременное прогнозирование доступности ресурсов с учётом динамически меняющихся требований ВМ является сложной задачей. Нехватка ресурсов на перегруженном хосте может привести к его отказу. По тому необходимо эффективное управление ресурсами на обоих хостах для успешной миграции.
– Множественная и коррелированная миграция. При миграции между ЦОД ограничения пропускной способности могут нарушить связь между коррелированными ВМ, что приведёт к снижению производительности и сбоям. Поэтому необходимы схемы, учитывающие коммуникацию между самими мигрирующими ВМ.

1.5.4 Прогнозирование нагрузки

Одними из важнейших исследовательских задач, связанных с облачными сервисами, является способность точного прогнозирования вычислительных потребностей и производительности приложений при различных схемах распределения ресурсов, использование моделей рабочей нагрузки при принятии решений по управлению ресурсами [78]. Перегрузка сервера — это не просто загрузка CPU >90%. Это состояние, при котором сервер не может обеспечить заявленные параметры производительности (SLA) для размещенных на нем рабочих нагрузок (ВМ, контейнеров). Проявляться она может по-разному:

– высокая загрузка CPU: очереди за процессорное время;
– нехватка памяти: своппинг, который снижает производительность;
– ожидание ввода/вывода: высокое время ожидания операций диска или сети;
– конкуренция за ресурсы между ВМ.

Пороговые методы являются самыми простыми. Для каждого типа серверов или даже для каждого отдельного сервера вручную задаются верхние пороги для метрик (CPU, RAM, ввод/вывод). Однако они не учитывают динамичность и нестационарность нагрузки, не предсказывает будущее, а лишь

реагирует на уже наступившее событие, требуют ручной настройки и глубокого знания специфики оборудования и нагрузок, часто приводит к ложным срабатываниям или, наоборот, к пропуску перегрузок.

В гетерогенной среде нельзя использовать единые абсолютные пороги (например, 80% CPU) для всех серверов. Процент загрузки на мощном сервере нового поколения и на устаревшем сервере — это совершенно разные по воздействию на производительность величины. Если для всех серверов в гетерогенном кластере установить общее правило: мигрировать нагрузку, если загрузка CPU $>$ 85%, то старые серверы будут непреднамеренно перегружены. Необходимо вводить абстрактную единицу измерения вычислительной мощности (например, vCPU или свои внутренние единицы, основанные на бенчмарках). И считать нагрузку в этих абсолютных единицах.

Статистические методы и анализ временных рядов основаны на анализе истории загрузки серверов для прогнозирования будущего.

Используются методы скользящего среднего, экспоненциальное сглаживание, локальная регрессия [79], ARIMA [80].

Данные методы в отличие от пороговых методов могут давать краткосрочный прогноз, но требуют настройки параметров и могут плохо работать с быстро меняющимися нагрузками со случайными всплесками.

Методы машинного обучения также широко используются для прогнозирования перегрузок серверов [81—83]. Прогнозирование производительности ВМ с GPU на конкретной конфигурации приведено в работах [84; 85]. Недостатками таких моделей являются высокие требования к качеству данных, сложность интерпретации результатов, а также необходимость переобучения нейронных сетей с изменением паттернов нагрузки, например, с запуском нового приложения. Кроме этого, вся цепочка «сбор метрик -$>$ прогноз -$>$ миграция» должна занимать время, существенно меньшее времени наступления перегрузки.

1.6 Выводы и постановка задачи диссертационного исследования

Рассмотренные выше тенденции и проблемы совершенствования распределения ресурсов инфокоммуникационной системы ЦОД позволили сформулировать цель и задачи работы.

Целью диссертационной работы является обеспечение высокой энергоэффективности инфокоммуникационной системы гетерогенного ЦОД и стабильности облачных сервисов путем разработки моделей и методов управления ресурсами программно-аппаратного комплекса распределенного планировщика ресурсов.

Поставленная в диссертационной работе цель достигается решением следующих задач:

1. Анализ проблем повышения эффективности функционирования гетерогенных центров обработки данных.
2. Разработка модели и метода многокритериального первоначального размещения разнородных виртуальных машин с учетом критериев энергопотребления, нарушения SLA-соглашений и загрузки ресурсов.
3. Обосновать выбор и выполнить адаптацию метода прогнозирования краткосрочной загрузки серверов для принятия своевременных решений об инициации миграции виртуальных машин.
4. Разработка и исследование критерия устойчивости инфокоммуникационной системы ЦОД путем определения оптимального размера окна наблюдения за серверами с учётом возможных помех, вызванных миграциями виртуальных машин.
5. Исследование характеристик процесса миграции виртуальных машин. Разработка метода оценки длительности миграции виртуальных машин.
6. Разработка метода многокритериального выбора целевых серверов для миграции виртуальных машин, обеспечивающий баланс между эффективностью использования вычислительных ресурсов и соблюдением SLA-соглашений.
7. Разработка имитационных моделей. Подтверждение эффективности разработанных моделей и методов.

8. Разработка алгоритмов и практических рекомендаций по внедрению результатов исследования в программно-аппаратные комплексы распределенного планировщика ресурсов отечественных облачных платформ.

Глава 2. Первоначальное размещение виртуальных машин в облачных центрах обработки данных

2.1 Постановка задачи

Одним из аспектов планирования ресурсов в облачной среде является первоначальное или статическое размещение виртуальных машин на группе физических серверов. Когда приходит запрос от пользователя на размещение одной или нескольких виртуальных машин, система управления ресурсами облачного ЦОД должна разместить эти ВМ на физических серверах – это решение должно согласовываться с целями управления облачного провайдера, т.е. задача размещения виртуальных машин заключается в их оптимальном закреплении за физическими серверами. Например, оптимальным может считаться такое размещение ВМ, при котором число физических серверов будет минимальным. Один из таких вариантов размещения виртуальных машин представлен на рис. 2.1. На рисунке видно, что виртуальные машины были

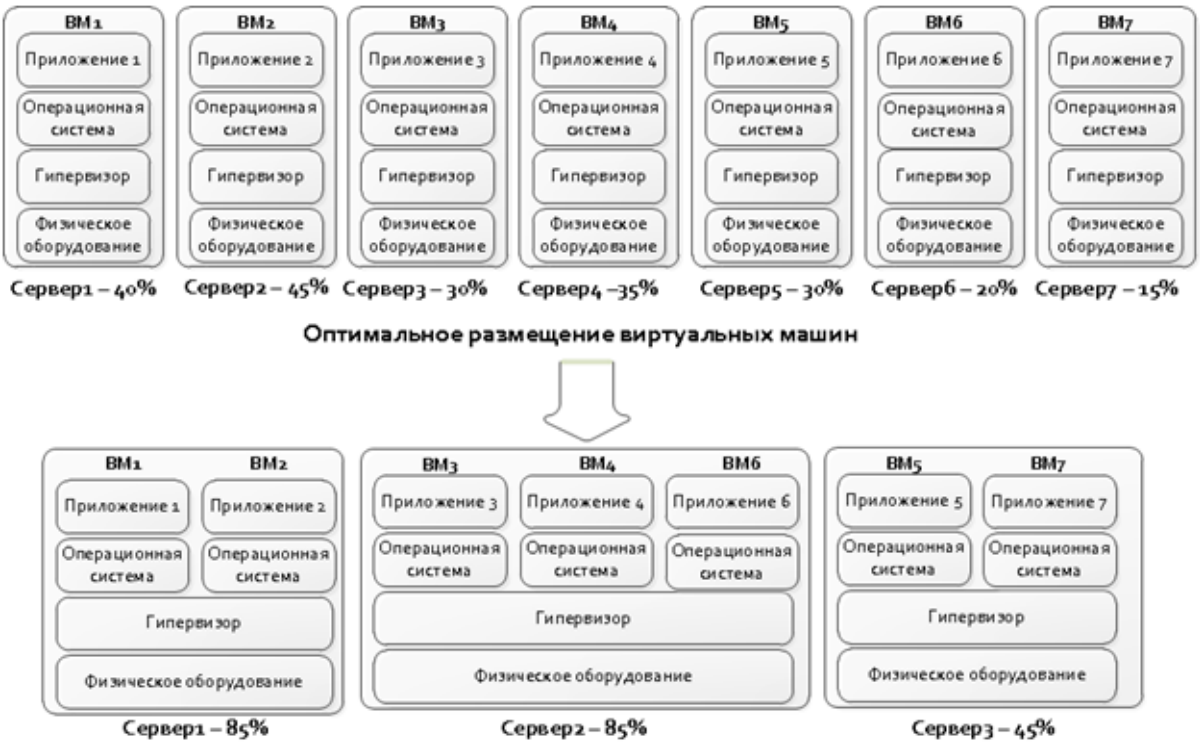

Рисунок 2.1 — Пример размещения виртуальных машин на меньшем числе хостов

размещены на трех физических серверах с более высокой загрузкой, а не на семи низкозагруженных, как было вначале.

Такая задача оптимизации может быть сведена к задаче об упаковке в контейнеры, если принять, что виртуальной машине присваивается статическая доля физических ресурсов (обычно процессор и память) и доли этих ресурсов можно складывать. Ключевая сложность заключается в комбинаторной природе задачи: необходимо найти такое назначение ВМ на серверы, чтобы:

- минимизировать количество задействованных физических узлов (для снижения энергопотребления);
- обеспечить равномерную загрузку ресурсов (CPU/RAM баланс);
- учесть аппаратные ограничения серверов (наличие GPU, специфичные архитектуры).

В данной главе приведено описание разработанного метода первоначального размещения виртуальных машин в гетерогенных ЦОДах для повышения их энергоэффективности, обеспечивая при этом минимальный уровень нарушений соглашений об уровне обслуживания (SLA) и равномерную загрузку ресурсов. В развитие анализа, проведенного в разделе 1.5.1, в данной главе предлагается метод, непосредственно учитывающий гетерогенность ресурсов и особенности GPU. Отличием данной работы от предыдущих исследований является сведение задачи размещения ВМ с GPU к известной задаче об упаковке в контейнеры, что возможно благодаря технологии MIG. Кроме этого, если реализовать метод дефрагментации памяти, предложенный в [55], то становится возможным использовать предложенный метод для динамического размещения. Также отличием является то, что в предложенной постановке задачи учитываются три противоречивых критерия: энергопотребление, остаток неиспользуемых ресурсов и нарушения SLA-соглашений.

Пусть имеется M различных физических серверов, каждый из которых характеризуется производительностью процессора, объемом оперативной памяти, объемом GDDRAM и производительностью графического процессора. Если сервер имеет несколько ядер процессора и графических процессоров, их число суммируется. Кроме того, заданы N виртуальных машин с собственными характеристиками vCPU, vRAM, vGDDRAM и vGPU, которые упорядочены пользователями. Виртуальные машины, размещенные на одном сервере, совместно используют его ресурсы. Каждой ВМ назначена своя доля ресурсов физического сервера, которая считается независимой от долей ресурсов других

ВМ. Конфигурация каждого сервера представлена в виде k-мерного вектора, где каждый компонент соответствует одному из ресурсов. Под оптимальным размещением ВМ понимается такое их назначение на физические серверы, которое минимизирует энергопотребление, нарушения SLA-соглашений и потери ресурсов.

2.2 Выбор критериев оптимизации

Энергопотребление

Как было показано в первой главе проблемы энергопотребления и тепловыделения являются очень существенными в современных ЦОД.

Значительные потери электроэнергии кроются в неэффективном использовании вычислительных ресурсов. Серверы в среднем используются лишь на 12-18% от их потенциальной вычислительной мощности. При этом простаивающее, но включенное оборудование потребляет до 60% от своего пикового энергопотребления. Это является основной причиной перерасхода электроэнергии, что в итоге приводит к повышенной совокупной стоимости владения (TCO) [16]. Таким образом, держать незагруженные серверы включенными является крайне невыгодным с точки зрения энергоэффективности.

Энергопотребление сервера с графическим процессором значительно выше, чем у обычного сервера, поскольку включает в себя и энергопотребление видеокарты, и может быть представлено в виде следующей модели: $f_{power} = f_{pow\_cpu} + f_{pow\_gpu}$. Модель энергопотребления процессора, полученная в ряде работ экспериментально [86], имеет следующий вид:

$$f_{pow\_cpu} = \begin{cases} p_0 + (p_1 - p_0) \cdot u_{cpu}, & u_{cpu} > 0; \\ 0, & u_{cpu} = 0. \end{cases} \tag{2.1}$$

где p_0—потребление электроэнергии незагруженным сервером;

p_1—потребление электроэнергии полностью загруженным сервером.

Значения параметров p_0 и p_1 для каждого типа процессоров различны, получить их можно из публикуемых некоммерческой организацией SPEC тестов [87].

Модель энергопотребления графического процессора по аналогии имеет следующий вид:

$$f_{pow\_gpu}(u_{gpu}) = \begin{cases} g_0 + (g_1 - g_0) \cdot u_{gpu}, & u_{gpu} > 0; \\ 0, & u_{gpu} = 0. \end{cases} \quad (2.2)$$

где g_0 —потребление электроэнергии незагруженной видеокартой; g_1 — потребление электроэнергии полностью загруженной видеокартой.

Общее энергопотребление сервера тогда является суммой энергопотребления графического процессора и сервера без GPU.

Неиспользуемые ресурсы

Остаток неиспользуемых на каждом хосте может существенно различаться в зависимости от размещенных на него виртуальных машин. Остаток ресурсов на каждом сервере должен быть сбалансирован по нескольким измерениям. На рис. 2.2 приведена иллюстрация использования памяти и процессора тремя виртуальными машинами. При размещении каждой ВМ тратятся ресурсы процессора и памяти. Остаток ресурсов заштрихован и его желательно минимизировать.

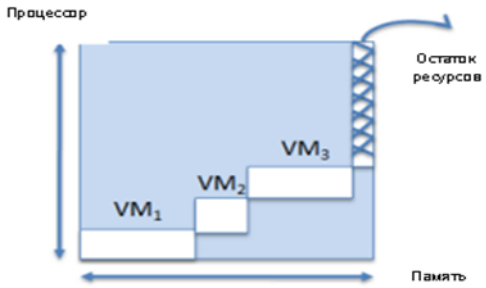

Рисунок 2.2 — Иллюстрация остатка неиспользуемых ресурсов

Как видно на рис. 2.2 ресурсы используются несбалансированно: память использована почти полностью, тогда как процессор – меньше, чем наполовину. Для получения модели неиспользуемых ресурсов введем следующие обозначения.

Пусть u_{CPU} — загрузка процессора, u_{RAM} — загрузка памяти, u_{GPU} и u_{GDDRAM} — загрузка процессоров и памяти видеокарты. Тогда критерий неиспользуемых ресурсов можно сформулировать следующим образом:

$$f_{res}(u_{CPU},u_{RAM},u_{GPU},u_{GDDRAM}) = 1 - \frac{(u_{CPU} \cdot u_{RAM} + u_{GPU} \cdot u_{GDDRAM})}{2} \quad (2.3)$$

Поскольку для работы ВМ критична одновременная доступность CPU и RAM (как и GPU и GDDRAM), вклад каждого типа ресурсов оценивается как произведение их загрузки. Общий критерий дисбаланса есть среднее арифметическое этих двух произведений, вычтенное из единицы, чтобы его можно было минимизировать.

Нарушения SLA-соглашений

Облачные ЦОД должны гарантировать соблюдение соответствующего качества обслуживания QoS, обычно задокументированное в форме SLA-соглашений. В случае предоставления SLA вопрос обеспечения ресурсов приобретает весомое значение. В настоящее время в облачных ЦОД размещают приложения тысячи клиентов со всего мира. Эти клиенты имеют различные требования, которые к тому же могут меняться время от времени. Например, предприятию-клиенту облачного сервиса может требоваться меньшее время ответа в дневное время, чем ночью. Для решения такой проблемы и снижения числа необходимых физических серверов потребуется разработка алгоритмов, координирующих работу сервисов с условиями выполнениями SLA-соглашений.

Пока, для упрощения средств управления облачными ЦОД, ресурсы каждому клиенту распределяются в соответствии с его индивидуальным SLA, без учета SLA других пользователей. Внутренние отличия между многочисленными SLA могут иметь огромное влияние на количество необходимых каждому пользователю ресурсов. Хотя гетерогенные требования клиентов делают алгоритмы планирования и консолидации ВМ сложными, они могут быть применены для повышения энергоэффективности. Важно разрабатывать и анализировать алгоритмы с использованием такой гетерогенности. Кроме того, чтобы соответствовать требованиям крупных облачных ЦОД, необходимо разработать масштабируемое решение, способное обслуживать тысячи пользователей.

Критерий нарушения SLA-соглашений можно сформулировать следующим образом:

$$f_{SLA}(u_{CPU}, u_{GDDRAM}) = 1 - (\frac{1}{1 + \mathrm{e}^{u_{CPU} \cdot 100 - a}} + \frac{1}{1 + \mathrm{e}^{u_{GDDRAM} \cdot 100 - b}})/2, \qquad (2.4)$$

где a — порог загрузки процессора, при превышении которого начинается деградация производительности; b — порог загрузки памяти видеокарты, при превышении которого начинается деградация производительности.

2.3 Математическая модель задачи

Математическая модель задачи оптимального размещения виртуальных машин имеет следующий вид [88; 89]:

$$\min_{x_{ij}} \frac{1}{M} \sum_{j=1}^{M} f^{j}_{power}(x_{ij}); \tag{2.5}$$

$$\min_{x_{ij}} \frac{1}{M} \sum_{j=1}^{M} f^{j}_{resource}(x_{ij}); \tag{2.6}$$

$$\min_{x_{ij}} \frac{1}{M} \sum_{j=1}^{M} f_{SLA}(x_{ij}) \tag{2.7}$$

при ограничениях

$$\sum_{i=1}^{N} r_i^{CPU} x_{ij} < c_j^{CPU}, j = 1, \cdots, M, \tag{2.8}$$

$$\sum_{i=1}^{N} r_i^{RAM} x_{ij} < c_j^{RAM}, j = 1, \cdots, M, \tag{2.9}$$

$$\sum_{i=1}^{N} r_i^{GPU} x_{ij} < c_j^{GPU}, j = 1, \cdots, M, \tag{2.10}$$

$$\sum_{i=1}^{N} r_i^{GDDRAM} x_{ij} < c_j^{GDDRAM}, j = 1, \cdots, M, \tag{2.11}$$

$$\sum_{j=1}^{M} x_{ij} = 1, i = 1, \cdots, N, \tag{2.12}$$

где x_{ij}—переменные, которые принимают значения 1, если виртуальная машина i принадлежит физическому серверу j и 0, если иначе;

$[r_i^{CPU}, r_i^{RAM}, r_i^{GPU}, r_i^{GDDRAM}]$—вектор потребности в ресурсах i-го облачного сервера;

$[c_j^{CPU}, c_j^{RAM}, c_j^{GPU}, c_j^{GDDRAM}]$—вектор имеющихся ресурсов на физическом сервере j.

Таблица 4 — Сравнение методов решения задач размещения виртуальных машин

Метод	Точность	Время	Риски
Эвристики (FFD)	Низкая	<1 сек	Дисбаланс ресурсов
Генетические (GA)	Средняя	Минуты	Локальные оптимумы
Линейное программир.	Высокая	Часы	Неприменимость в реальном времени
Муравьиный алгоритм	Средняя	Секунды	Требуется тонкая настройка параметров
Обучение с подкреплением	Высокая	Секунды	Неустойчивость при новых типах ВМ
Графовые нейросети	Высокая	5–10 сек	Черный ящик, сложность отладки

2.4 Метод решения

Задача об упаковке в контейнеры относится к классу NP-трудных задач. Точным методом решения задачи является метод ветвей и границ. Для задачи большой размерности используются приближенные эвристические алгоритмы, например, жадные и генетические.

На практике в качестве метода консолидации виртуальных машин получили распространение эвристические алгоритмы как FFD (First Fit Decreasing — первый подходящий по убыванию) и BFD (Best Fit Decreasing — наилучший подходящий по убыванию), а также их модификации [90—93]. Одной из таких модификаций является алгоритм Sercon, реализованный в OpenStack, который пытается минимизировать число хостов с минимальным числом миграций [94]. Хорошие результаты с точки зрения близости к оптимальному решению и времени вычислений показали группирующие генетические алгоритмы [39; 95] и муравьиные алгоритмы [45; 96]. Сравнение алгоритмов приведено в таблице 4.

Метаэвристика муравьиной колонии (ACO) доказала свою эффективность в различных областях, включая задачи дискретной оптимизации, и превосходит некоторые проверенные алгоритмы [97]. Среди недавних работ, в которых ACO используется для размещения виртуальных машин, можно отметить работы [35; 96].

Алгоритмы ACO были первоначально разработаны на основе наблюдения за уникальной способностью реальных муравьев определять кратчайший путь между своим домом и источником пищи. Особенностью является то, что муравьи выделяют особое вещество, называемое феромоном, вдоль пройденного ими маршрута. Запах феромона позволяет другим особям распознавать выбранный ими путь и следовать по нему. Когда колония сталкивается с вы-

бором из двух маршрутов, каждое насекомое сначала делает выбор случайным образом, распределяя равные шансы между путями. Но поскольку насекомые, выбравшие кратчайший путь, возвращаются раньше остальных, концентрация феромонов на нём быстро увеличивается. Это стимулирует остальную часть колонии выбирать этот путь. Со временем почти все муравьи начинают двигаться исключительно по оптимальному маршруту. Применительно к задаче о размещении виртуальных машин каждый искусственный муравей создаёт решение, состоящее из списка назначений $VM_i - PM_j$ (виртуальная машина i на физический сервер j). Уровни феромонов связаны с каждым назначением $VM_i - PM_j$ отражая желательность такого назначения. Эвристические значения динамически вычисляются для каждого назначения $VM_i - PM_j$, представляя собой предпочтение при назначении. Количество муравьёв является входным параметром алгоритма. Каждый муравей независимо распределяет ВМ по серверам на основе случайно переставленного списка ВМ и создаёт своё уникальное распределение, формируя таким образом потенциальное решение. Затем муравьи проверяют возможность размещения каждой ВМ на выбранном сервере, оценивая доступность необходимого vCPU, vRAM, vGPU, vGDDRAM в соответствии с ограничениями (2.8)–(2.10). Блок-схема алгоритма приведена на рис. 2.3.

Рассмотрим основные блоки схемы алгоритма определения маршрута методом муравьиной колонии, приведенной на рис. 3.

- **Блок «Начало»**. Старт алгоритма.
- **Блок «Инициализация параметров»**. На этом этапе задаются начальные значения параметров для работы алгоритма, такие как количество муравьёв, количество феромона и другие настройки. Инициализируется матрица феромонов, которая представляет вероятность размещения ВМ на физическом сервере. Начальное значение феромона для каждого размещения одинаково $\tau_0 = 1$. Это гарантирует, что все феромонные следы начинаются с одинаковой начальной концентрации.
- **Цикл «Превышено число итераций *Iter*?»**. Проверяется, достигнуто ли заданное в качестве параметра число итераций алгоритма. Если да, то алгоритм завершается. За время одной итерации каждый муравей строит одно решение.
- **Цикл «Превышено число муравьев *nAnts*?»**. Каждый муравей формирует решение. Если достигнуто максимальное число муравьев,

то счетчик итераций увеличивается на единицу и происходит переход к проверке условия предыдущего цикла.

– **Блок «Построение решений каждым муравьем».** Каждый муравей последовательно строит решение, назначая виртуальные машины на физические серверы. Выбор сервера j для текущей виртуальной машины i осуществляется на основе **правила псевдослучайного пропорционального выбора**, которое учитывает текущий уровень феромона τ_{ij} и эвристическую информацию η_{ij}. Эвристическая информация η_{ij} определяется как величина, обратная ожидаемому дисбалансу ресурсов сервера после размещения ВМ (или его загрузке), что стимулирует выбор менее загруженных серверов.

  Правило выбора формулируется следующим образом. Генерируется случайное число q, равномерно распределенное на отрезке $[0, 1]$:

  – если $q \leqslant q_0$ (режим *эксплуатации*), муравей выбирает сервер j^*, обеспечивающий наилучшее локальное решение:

$$j^* = \arg \max_{k \in \text{доступные серверы}} \left\{ \tau_{ik} \cdot [\eta_{ik}]^\beta \right\}; \tag{2.13}$$

  – если $q > q_0$ (режим *исследования*), сервер j выбирается случайным образом с вероятностью, пропорциональной тому же произведению:

$$p_{ij} = \frac{\tau_{ij} \cdot [\eta_{ij}]^\beta}{\sum\limits_{k \in \text{доступные серверы}} \tau_{ik} \cdot [\eta_{ik}]^\beta}. \tag{2.14}$$

  Параметр $q_0 \in [0{,}1]$ определяет баланс между эксплуатацией (использованием лучшего известного пути) и исследованием (поиском новых решений). Параметр β задает относительную важность эвристической информации по сравнению с интенсивностью феромонного следа.

– **Блок «Решение допустимо?»** является проверкой, что все ограничения на ресурсы CPU, RAM, GPU и GDDRAM выполнены. При выполнении всех ограничений происходит переход к следующему блоку, иначе счетчик муравьев увеличивается на единицу и происходит переход на блок проверки числа муравьев.

– **Блок «Вычисление целевой функции fitness».** Целевая функция рассчитывается как взвешенная сумма трёх критериев (2.5) – (2.7):
  $f = \alpha_1 \cdot f_{power} + \alpha_2 \cdot f_{res} + \alpha_3 \cdot f_{sla} \to min$

$\alpha_1 + \alpha_2 + \alpha_3 = 1,\ \alpha_1, \alpha_2, \alpha_3 \leqslant 0$

– **Блок «fitness $<$ best_fitness?»**. Проверка улучшения целевой функции. Если да, то переход к следующему блоку, иначе счетчик муравьев увеличивается на единицу и происходит переход на блок проверки числа муравьев.
– **Блок «best_fitness = fitness»**. Обновление значения целевой функции.
– **Блок «Обновление феромонов»**. После того как все муравьи построили все свои решения, обновляется информация о феромонах.

Операция локального обновления феромона применяется всякий раз, когда муравей получает успешное решение: это предотвращает повторение предыдущих действий и стимулирует множество решений.

Для каждой пары $VM_i - PM_j$ обновленное значение уровня феромона τ_{ij}^{t+1}:

$$\tau_{ij}^{t+1} = (1 - \rho) \cdot \tau_{ij}^{t} + \rho \cdot \tau_0. \tag{2.15}$$

Параметр испарения феромонов, обозначаемый символом ρ, определяет скорость уменьшения интенсивности феромонных следов с течением времени. Начальная концентрация феромонов задается значением τ_0 и отражает начальную интенсивность следа.

Глобальное обновление феромонов. В конце каждой итерации наилучшее решение, обнаруженное на тот момент, обозначаемое как $S*$, подвергается процессу глобального обновления феромонов. Следует отметить, что феромон испаряется на всех путях. Затем наилучшие решения усиливаются добавлением феромона. Этот шаг усиливает феромонные метки наилучшего решения, стимулируя последующие поколения муравьёв следовать по этому пути. Обновлённые концентрации феромона помогают новым муравьям выбирать перспективные варианты размещения серверов, облегчая поиск оптимальных решений.

$$\tau_{ij}^{t+1} = (1 - \rho) \cdot \tau_{ij}^{t} + \Delta\tau_{ij}. \tag{2.16}$$

$$\Delta\tau_{ij} = \begin{cases} q/f(S^*), & \text{если} \quad ij \in S^*; \\ 0, & \text{иначе.} \end{cases} \tag{2.17}$$

q — параметр обновления феромона.

– **Блок «Конец»**. Завершение работы алгоритма.

2.5 Вычислительные эксперименты

Была проведена серия экспериментов имитационного моделирования, в ходе которой предложенный алгоритм сравнивался с политиками FF и BF, используемых на практике для размещения виртуальных машин с графическими процессорами.

Политика FF: алгоритм методично проверяет доступные серверы и установленные на них видеокарты, выявляя ресурсы, подходящие для выделения необходимого объема виртуальной памяти. Как только подходящий вариант найден, виртуальная машина немедленно размещается. Этот подход получил широкое распространение благодаря своей наглядности и высокой производительности.

Политика BF: политика BF определяет полный список GPU в ЦОД, которые соответствуют требованиям конкретного запроса виртуальной машины. Затем выбирается GPU, который минимизирует количество оставшихся свободных блоков, обеспечивая оптимальное использование ресурсов видеокарты.

Моделируемый центр обработки данных состоит из разнородных физических серверов и ВМ. Потребность в ресурсах выражается в общей ресурсной емкости физических серверов. Требования к ресурсам ВМ генерируются случайным образом. Производительность vCPU в ядрах равномерно распределена из следующего набора значений {0,25, 0,5, 1, 1,5, 2, 2,5, 3, 4}, а объем vRAM в ГБ распределен из того же набора значений. vGPU и vGDDRAM генерируются как экземпляры MIG со следующим набором значений (1,5), (2,10), (3,20), (7,40). Количество серверов и ВМ задается в качестве начальных значений для моделирования центров обработки данных различных размеров. Наборы случайных входных данных генерируются для каждой серии экспериментов. Каждый эксперимент был выполнен 20 раз. Результаты усреднены.

На рисунке 3 показаны средние нормализованные значения критериев с одинаковыми весами для 100 ВМ и 60 физических серверов. Для ясности показано среднее использование ресурсов. Алгоритмы FF и BF, как правило, размещают ВМ на меньшем количестве серверов, что приводит к более высо-

Таблица 5 — Настройки параметров

Число муравьев	**Число итераций**	β	ρ	q
6	30	2	0.3	1

кой загрузке ресурсов и нарушениям SLA. По сравнению с ними предложенный алгоритм имеет наибольшее энергопотребление, но уровень потерь ресурсов и нарушений SLA-соглашений значительно ниже, поскольку этот алгоритм пытается сбалансировать ресурсы во всех измерениях и избежать их чрезмерного использования (рис. 2.4).

Для определения временной сложности предложенного алгоритма было проведено моделирование с увеличением числа ВМ и серверов. Число серверов составляло 60% от числа ВМ. График зависимости времени вычислений от размера задачи представлен на рис. 2.5. Алгоритм реализован на языке программирования *R* и был запущен на компьютере Dell, оснащенном четырехъядерным процессором Intel Core i5-10210U с тактовой частотой 1,6 ГГц и объемом оперативной памяти 8 ГБ. Экспериментально показано, что время решения растет нелинейно, и может быть аппроксимировано квадратичной зависимостью от числа ВМ, что подтверждает применимость алгоритма для задач реальной размерности.

2.6 Параметры муравьиного алгоритма

Параметры муравьиного алгоритма определялись анализом чувствительности. Моделирование проводилось посредством 100 независимых прогонов. Результаты были усреднены. Значения параметров установлены следующим образом (таблица 5).

На рисунке 2.6 представлены изменения значения целевой функции при различных параметрах муравьиного алгоритма. Как видно из рисунка 4, количество агентов-муравьев (nAnts) постепенно увеличивается от 1 до максимального значения 10, а минимальное значение целевой функции впервые достигается при nAnts = 6, и дальнейшее увеличение количества муравьев не улучшает качество решения.

На рисунке 2.7 показано, как влияет на целевую функцию относительная важность эвристического значения β по сравнению с уровнем феромона в вероятностном решающем правиле (2.14). При $\beta < 2$ уровень феромона оказывает большее влияние по сравнению с эвристическим значением. Целевая функция достигает лучшего значения при $\beta = 2$, а при более высоких значениях эвристическая часть доминирует в вероятностном решающем правиле, и качество решения ухудшается.

Целевая функция сходится за разное число итераций, что приведено на рис. 2.8. В целом, достаточно 30 итераций.

На рис. 2.9 приведены значения целевой функции при различных значениях параметра испарения феромона. При ρ=0,3 достигается минимальное значение.

2.7 Выводы

1. Впервые задача размещения ВМ с GPU сформулирована как многомерная задача упаковки контейнеров с учетом аппаратных возможностей разделения GPU (технология MIG), что позволило более точно моделировать использование данного вида ресурсов.
2. Показана возможность эффективного использования предложенного подхода не только для статического (первоначального) размещения, но и для динамического перераспределения виртуальных машин в условиях изменяющейся нагрузки. Ключевым фактором стало комплексное применение технологии NVIDIA Multi-Instance GPU (MIG) совместно с методом дефрагментации памяти, описанным в литературе, что обеспечивает необходимую гибкость управления ресурсами.
3. Разработана математическая модель задачи многокритериальной оптимизации первоначального размещения ВМ в гетерогенной инфраструктуре. Модель формализует и позволяет находить баланс между тремя конфликтующими целевыми функциями: минимизацией энергопотребления, снижением дисбаланса в использовании разнородных ресурсов и предотвращением нарушений соглашений об уровне сервиса (SLA).

4. Для решения сформулированной задачи успешно адаптирован муравьиный алгоритм. Экспериментально подтверждено, что данный алгоритм позволяет находить сбалансированные по заданным критериям решения за приемлемое вычислительное время, что делает его пригодным для практического использования.
5. Разработанная модель и алгоритм готовы к интеграции в модуль планировщика ресурсов гетерогенных облачных ЦОД. Их внедрение позволит формировать научно обоснованные и эффективные планы первоначального размещения виртуальных машин для эффективной работы всей инфраструктуры.

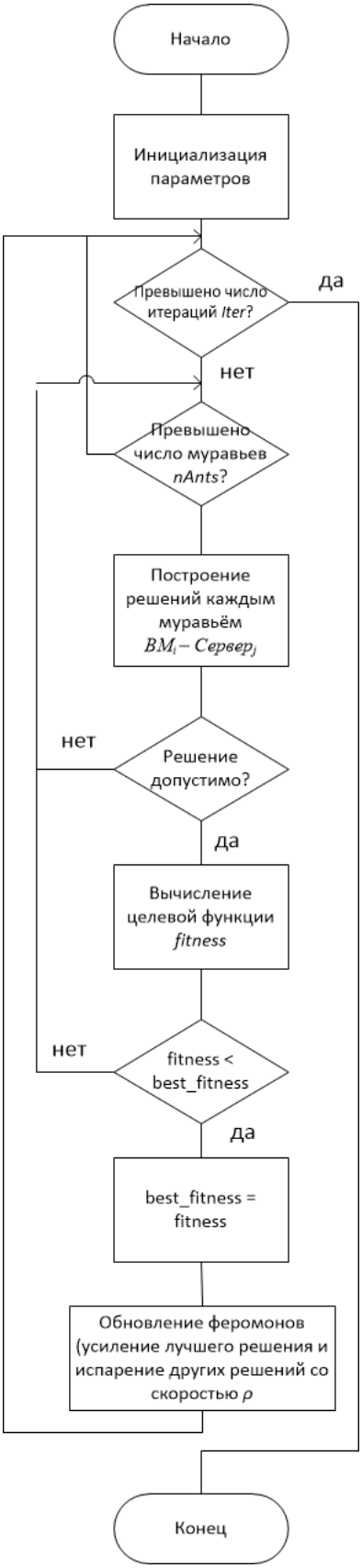

Рисунок 2.3 — Блок-схема муравьиного алгоритма

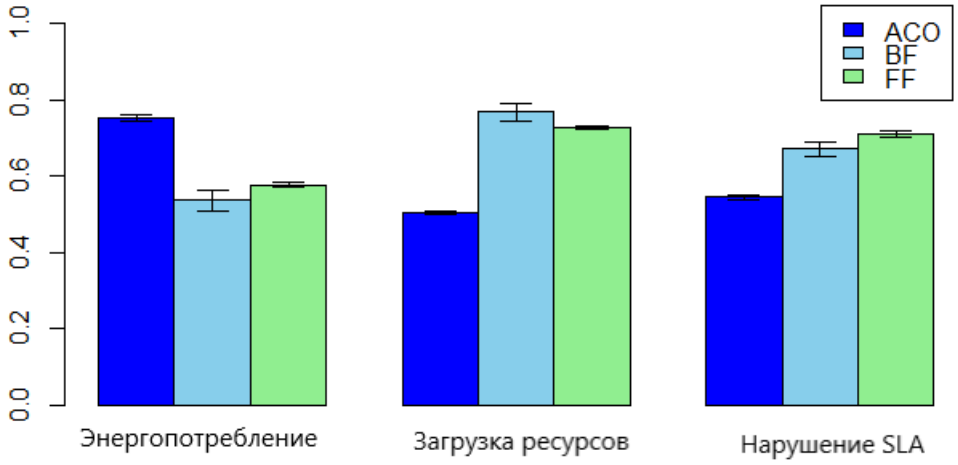

Рисунок 2.4 — Сравнение предлагаемого алгоритма с известными эвристиками FF и BF

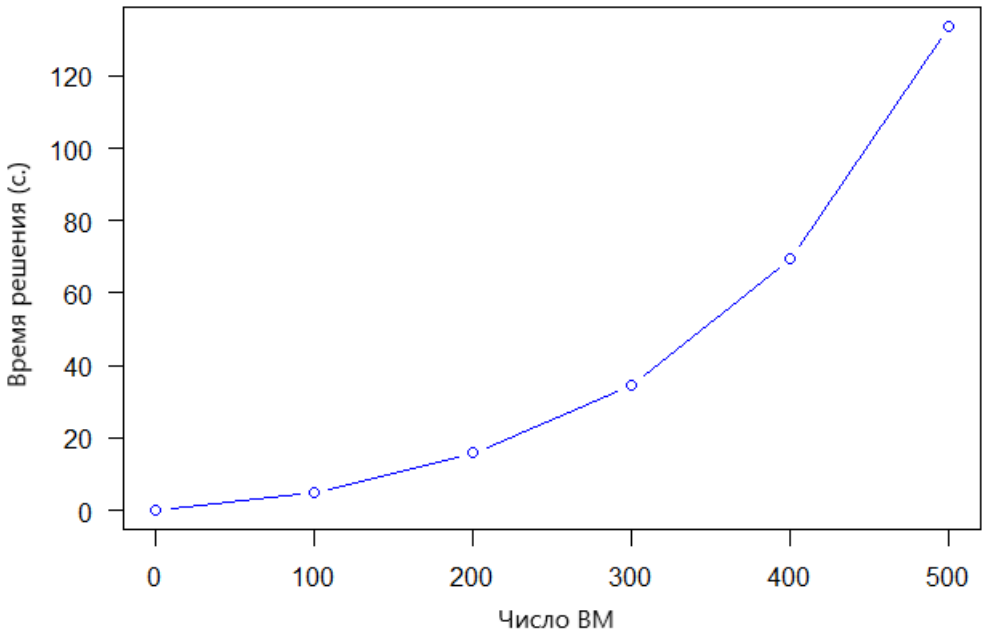

Рисунок 2.5 — Время решения задач большой размерности

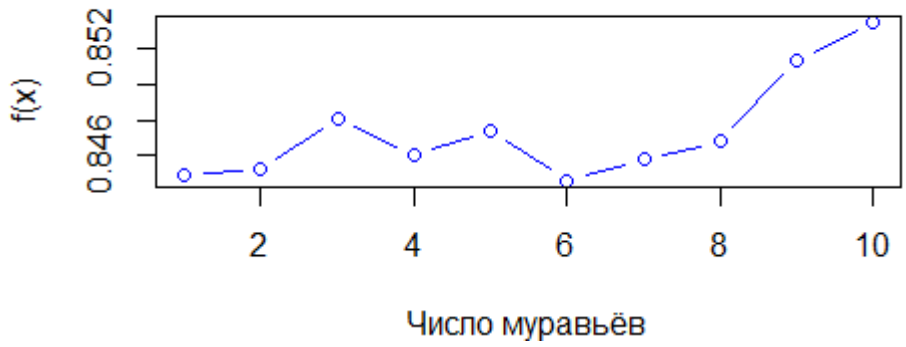

Рисунок 2.6 — Изменение значения целевой функции при различном числе муравьев

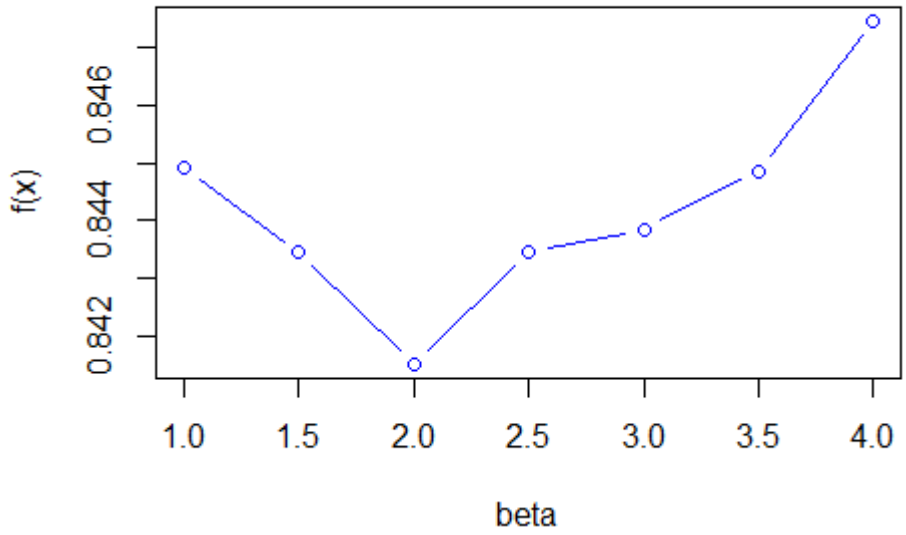

Рисунок 2.7 — Влияние на целевую функцию относительной важности эвристического значения β

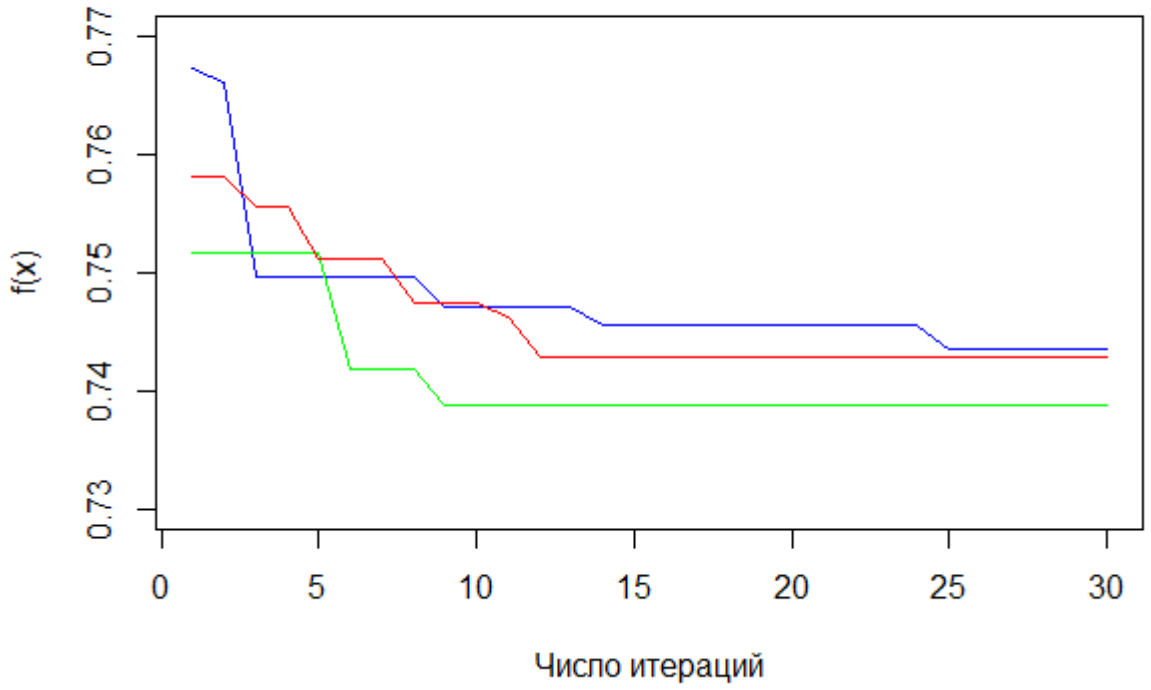

Рисунок 2.8 — Сходимость целевой функции при различном числе итераций

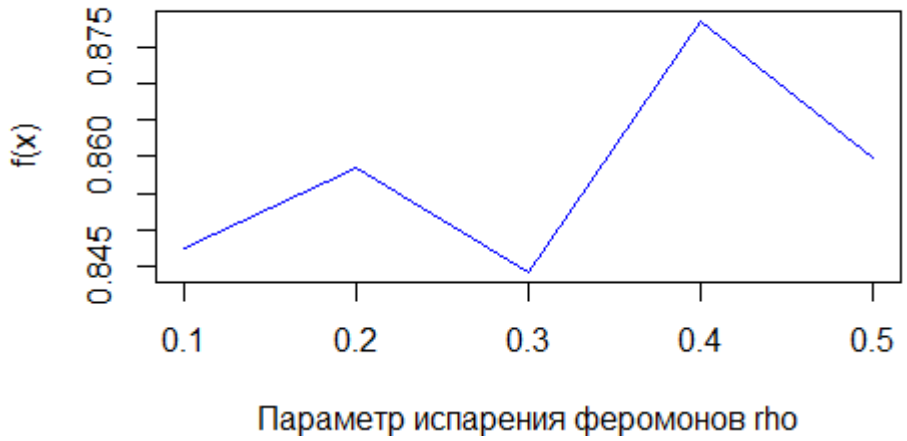

Рисунок 2.9 — Значения целевой функции при различных значениях параметра испарения феромона

Глава 3. Мониторинг и прогнозирование характеристик инфокоммуникационной системы при динамическом распределении ресурсов

3.1 Проблема динамического распределения ресурсов

Центры обработки данных должны предоставлять достаточное количество ресурсов для бесперебойной работы приложений с заданным качеством обслуживания. При этом нагрузка на приложения может существенно изменяться в течение суток и иметь случайные всплески. Динамическое распределение ресурсов на уровне IaaS основывается на использовании миграции виртуальных машин, перемещаемых в зависимости от текущих условий работы системы. Система управления ресурсами должна отвечать на следующие вопросы.

1. Когда нужно производить перемещение ВМ?
2. Какие ВМ выбрать для миграции?
3. Куда перемещать ВМ?

Современные гипервизоры VMware ESX, Microsoft Hyper-V или Xen позволяют администратору в ручном режиме осуществлять динамическое распределение ВМ. Однако в условиях ЦОД критически важным является оперативное принятие решения с учетом текущих и прогнозируемых значений состояния системы. Необходимо обеспечить такой режим управления, при котором: исключаются неоправданные перемещения ВМ; минимизируются ошибки при выборе целевых серверов; своевременно выявляются и устраняются состояния как перегрузки (угроза отказа из-за нехватки ресурсов), так и недогрузки (снижение энергоэффективности) физических серверов.

Данные проблемы решаются посредством перераспределения виртуальных машин между серверами. Если сервер испытывает повышенную нагрузку, то ВМ мигрируют на менее загруженные серверы. Для повышения энергоэффективности рекомендуется перенести все виртуальные машины с проблемного сервера на другие устройства, после чего сам сервер переводится в состояние сна либо отключается.

Важно подчеркнуть, что устранение указанных проблем представляет собой сложную и двусмысленную задачу. Практика показывает, что политика,

ориентированная исключительно на минимизацию энергопотребления путем концентрации ВМ на ограниченном числе хостов, способна привести к перегрузке активных серверов и, как следствие, к нарушению соглашений об уровне обслуживания. Возникает необходимость одновременного учета двух противоречивых целей. В связи с этим в диссертационной работе разработан многокритериальный метод динамического распределению ресурсов ЦОД, интегрирующий рассмотренные выше факторы.

3.2 Архитектура системы

Система динамического распределения ресурсов на уровне IaaS также строится в соответствии с архитектурой автономных вычислительных систем (MAPE-K) «Мониторинг, анализ, планирование и выполнение» на основе централизованного или децентрализованного контроллера [98].

Рассмотрим децентрализованную систему управления ресурсами, имеющую двухуровневую архитектуру, состоящую из глобального и локальных контроллеров (рис. 3.1) [99]. Двухуровневая архитектура управления ресурсами оказывается более эффективной по сравнению с централизованным подходом, при котором все управляющие функции выполняются единым центром. Поскольку виртуальные машины независимы друг от друга и могут реализовываться на различных платформах, то возможны различные реализации локальных контроллеров. Все внутренние сложности функций управления в виртуальных машинах и контейнерах переводятся локальными и глобальным контроллерами в простые запросы на объем необходимых ресурсов к гипервизору.

Локальные контроллеры анализируют состояние физических серверов, на которых они расположены и определяют возможные состояния недогрузки и перегрузки на основании прогноза для следующего окна наблюдения. В случае выявления одного из перечисленных состояний, локальный контроллер сообщает об этом глобальному контроллеру, который выбирает сервер назначения, на который будет производиться миграция ВМ.

В данной системе локальный контроллер отвечает на следующий вопрос:

– Когда нужно производить перемещение ВМ?

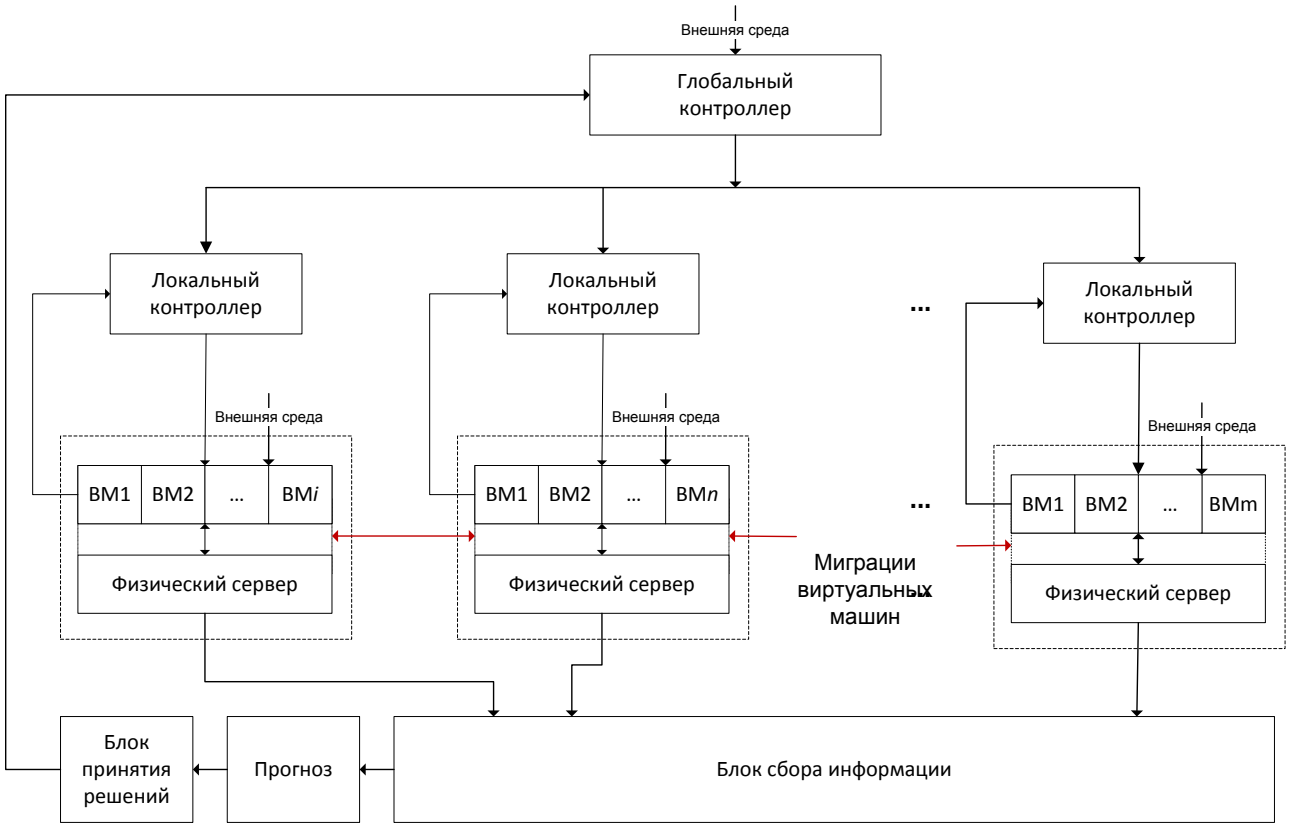

Рисунок 3.1 — Двухуровневая система управления ресурсами облачного ЦОД

Тогда как глобальный

– Какие ВМ выбрать для миграции?
– Куда перемещать ВМ?

Для этого локальные контроллеры постоянно анализирует данные системы мониторинга, и в случае выхода показателей за допустимые пределы сообщают об этом глобальному контроллеру, который активизирует процесс миграции виртуальных машин.

Проверка показателей системы осуществляется последовательно, в соответствии с важностью критериев (перегрузка, недогрузка сервера), при этом процесс наблюдения осуществляется непрерывно, в том числе в случае выполнения действий по перемещению ВМ. Приоритет критериев может быть изменен администратором системы, однако в силу их противоречивости этап проверки должен оставаться последовательным (рис. 3.2). При возникновении условий, инициирующих миграцию, глобальный контроллер идентифицирует виртуальные машины, размещенные на проблемных серверах, и запускает процедуру поиска целевого сервера. После выбора сервера назначения и начала переноса целевой физический сервер временно выводится из пула доступных ресурсов

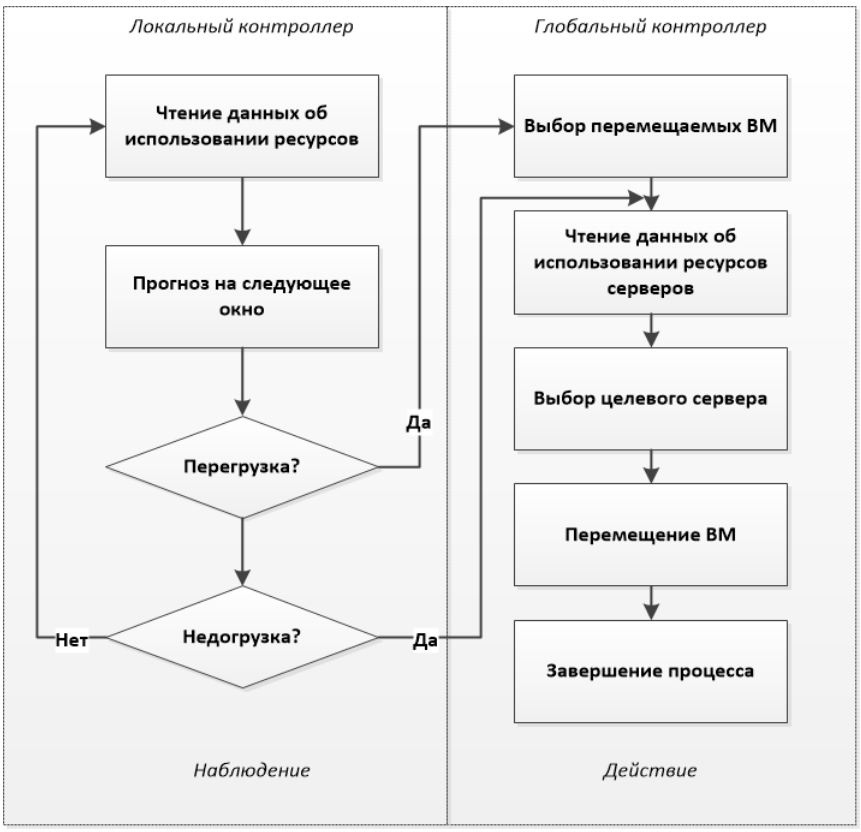

Рисунок 3.2 — Алгоритмы работы локальных и глобального контроллеров

до завершения миграции и подтверждения стабильного функционирования системы.

3.3 Мониторинг и прогноз нагрузки

3.3.1 Метод скользящего окна

Система мониторинга собирает для блока прогнозирования данные о текущем состоянии виртуальных машин и физических хостов, такие как загрузка процессора, памяти, сети. Блок прогнозирования определяет возможную критическую ситуацию в следующем окне наблюдения.

Подходы к определению момента, когда нужно начинать миграцию ВМ можно разбить на три группы [31; 100]:

1. Периодическое применение статических алгоритмов оптимального размещения ВМ.

2. Эвристические алгоритмы на основе пороговых уровней.
3. Использование методов прогнозирования.

Первый вариант не является эффективным, так как статические алгоритмы не учитывают предысторию размещения виртуальных машин и предназначены в основном только для первоначального размещения.

Наиболее простым и распространенным решением считается установка пороговых значений для ключевых метрик, таких как температура, загруженность сервера и количество нарушений SLA. Этот подход широко применяется в промышленности благодаря своей простоте и эффективности. Обычно на серверах устанавливаются верхние и нижние границы загрузки CPU, например, верхний предел составляет 80%, а нижний — 20%.

При превышении верхнего порога система определяет сервер как перегруженный, что требует переноса одной или нескольких ВМ. Когда же загрузка опускается ниже нижней границы, сервер считается недостаточно нагруженным, и тогда целесообразно перенести все ВМ на другие узлы и отключить его для экономии ресурсов.

Тем не менее, применение статичных пороговых значений оказывается неэффективным в системах с динамической нагрузкой, потому что такие правила не способны адаптироваться к изменениям нагрузки и игнорируют среднесрочную динамику поведения системы, приводя к принятию неоптимальных решений. Ключевая задача состоит в различии временных колебаний от устойчивых изменений нагрузки, что достижимо лишь при объединении пороговых алгоритмов с методами прогнозирования. Именно такой комплексный подход представляется наиболее актуальным и многообещающим на сегодняшний день.

Преимуществом такого подхода является то, что можно начать миграцию до наступления перегрузки, упреждающе. Можно принять более взвешенное решение, учитывая не текущий «мгновенный» скачок, а долгосрочный тренд. Например, если модель предсказывает, что нагрузка на сервер через 10 минут превысит порог, система уже сейчас может запустить миграцию наименее критичной ВМ с этого сервера, предотвращая сбой.

Формально постановку задачи прогнозирования можно сформулировать следующим образом.

Пусть задан временной ряд загрузки процессора $Y = \{y_1, y_2, ..., y_T\}$. Необходимо найти функцию f, которая по предшествующим значениям ряда $\{y_{t-h}, y_{t-h-1}, ..., y_{t-1}\}$ позволяет получить прогноз $\hat{y}_t$ на горизонт h.

Обозначим:

– $\tau_1, \tau_2 \in \mathbb{R}$ — критические пороги недогрузки и перегрузки соответственно;

– f — модель (алгоритм) прогнозирования с параметрами θ, подлежащими оптимизации.

Цель обучения состоит в нахождении оптимальных параметров θ^*, минимизирующих ожидаемые потери на новых данных:

$$\theta^* = \arg\min_{\theta} \mathbb{E}_{(X,Y)\sim P_{data}} \left[L\big(Y, f(X;\theta)\big) \right], \tag{3.1}$$

где P_{data} — неизвестное распределение данных, порождающее пары «признаки — целевая переменная» ($X = \{y_{t-h}, ..., y_{t-1}\}$, $Y = y_t$), а L — функция потерь, например, среднеквадратическая ошибка.

Для прогнозирования необходимо организовать сбор данных, обычно для этого используется метод скользящего окна.

Метод скользящего окна является способом, позволяющим применять модель стационарного потока для нестационарного [101]. Основной идеей является аппроксимация нестационарной нагрузки последовательностью стационарных нагрузок $U = (u_1, u_2, \cdots, u_{Nu})$, которые поступают одна после другой. В данной модели матрица вероятностей переходов P становится функцией от текущей стационарной нагрузки $P(u)$.

Основная трудность использования данного метода заключается в зависимости детектирования от длины окна [102].

В работе [101] авторы назвали политику, которая принимает «идеальные» решения для текущей стационарной нагрузки «лучшей адаптивной политикой». Однако данная политика требует полного знания о всей последовательности нагрузок и времени их смены. Поскольку в реальных условиях модель нагрузки u_i строится на основе накопленных статистических данных, а моменты смены состояний заранее неизвестны, возникает необходимость применения эвристических методов, позволяющих достичь эффективности, близкой к «лучшей адаптивной политике».

Применяется метод скользящего окна, охватывающего последние l_w событий. Пусть c_{ij} — наблюдаемая частота переходов из состояния i в j в

пределах текущего окна. Тогда вероятности переходов оцениваются по формуле: $\hat{p_{ij}} = \frac{c_{ij}}{(\sum_{k \in S} c_{ik})}$.

При $l_w \to \infty$ оценка $\hat{p_{ij}}$ сходится к истинной p_{ij}, при длине нагрузки u_i [101].

Недостатки метода:

– систематическая ошибка при l_w меньше длины последовательности;

– ошибка разрешения (точность $1/l_w$.) [103];

– время адаптации при смене нагрузки.

В работе [103] был расширен метод скользящего окна с использованием окон различного размера, где размер используемого окна динамически выбирается с использованием информации о предыдущем состоянии системы и отклонений в оценках, полученных в различных окнах. Алгоритм динамически выбирает размер окна, снижая систематическую ошибку и сочетая преимущества малой ошибки выборки (большие окна) и малой ошибки идентификации (малые окна).

Для того чтобы эффективно сочетать метод скользящего окна для слежения за серверами и использование пороговых уровней необходимо, чтобы статистика, рассчитываемая на основе окон, была робастной (устойчивой). Если при резком изменении малой доли количества данных (вследствие выбросов или других случайных искажений) значение статистической характеристики претерпевает существенные изменения, то такая статистика неустойчива. Так в работе для определения порогового уровня загрузки процессора [79] предложено использование нескольких адаптивных эвристических алгоритмов с использованием таких робастных статистик, как медианное абсолютное отклонение и межквартильный размах. Недостатком таких алгоритмов является отсутствие прогноза, что не позволяет превентивно управлять ресурсами ЦОД.

Метод скользящего окна обладает ограничением: он нечувствителен к трендовым изменениям и реагирует только на уже произошедшие события. Для повышения качества управления крайне желательно прогнозировать значения показателей на ближайшую перспективу. Это позволит избежать необоснованных миграций в ситуациях, когда превышение порога вызвано кратковременными явлениями (например, случайным всплеском нагрузки). Если прогноз указывает на выход загрузки процессора за пределы допустимых значений, локальный контроллер инициирует процесс миграции.

3.3.2 Методы прогнозирования состояния хоста

Для выявления трендовой составляющей временного ряда используются регрессионные методы. Так в работе [104] для прогнозирования перегрузок серверов использовался метод наименьших квадратов для поиска параметров линейной регрессии загрузки процессора. Использование метода локальной регрессии обосновано в работе [79]. В данной работе эффективность алгоритма локальной регрессии сравнивалась с пороговыми алгоритмами на основе таких робастных статистик, как абсолютное медианное отклонение и межквартильный размах.

Основной идеей метода локальной регрессии, предложенного Кливлендом [105] является подбор простых моделей на локализованном наборе данных для построения кривой, которая аппроксимирует исходные данные. Наблюдениям (x_i, y_i) присваиваются веса соседей с использованием трикубической функции весов, причем чем дальше находится наблюдение от x_i, тем меньше должен быть его вес. После того как веса заданы, взвешенным методом наименьших квадратов рассчитываются оценки коэффициентов либо локально-линейной, либо локально-полиномиальной регрессии. Далее происходит переход к следующей точке: расчет весов, расчет коэффициентов новой модели, и так далее, до тех пор, пока не будут получены $\hat{y}_i$ для всех наблюдений, после чего в распоряжении исследователя оказывается сглаженный ряд. Более подробно алгоритм метода локальной регрессии приведен в [106].

В предложенном в работе [79] алгоритме перегрузки сервера данный подход применяется для определения линейного тренда k последних точек наблюдения за загрузкой процессора, где $k = q/2$, а q – количество наблюдений в подмножестве данных, локализованных вокруг x. Новая линия тренда находится для каждого наблюдения. Эта линия тренда используется для оценки следующего наблюдения. Если прогнозное значение загрузки процессора $\hat{g}(x_{k+1})$, умноженное на параметр безопасности s, $s \in \mathbb{R}^+$ больше 1, т.е. неравенство 3.2 удовлетворяется, то хост перегружен и требуется убрать некоторые ВМ с хоста.

$$s \cdot \hat{g}(x_{k+1}) \geqslant 1. \tag{3.2}$$

Кроме регрессионных моделей для прогнозирования нагрузки использовалась интегрированная модель авторегрессии — скользящего среднего (ARIMA) [107], авторегрессионная модель использовалась также в [108]. Для избавления от шумов и сглаживания данных в работах [109; 110] применялся фильтр Калмана. Нейронная сеть для прогнозирования перегрузок серверов использовались в работе [110; 111].

Однако для эффективного обучения нейронных сетей необходимо множество наблюдений. Для задачи прогнозирования перегрузки серверов с переменным числом ВМ в режиме реального времени такое количество статистических данных может оказаться недоступным. Другим недостатком моделей на основе нейронных сетей являются значительные временные затраты на обучение модели для достижения удовлетворительного результата. Эта проблема не столь существенна, если исследуется небольшое число временных последовательностей, однако крупный ЦОД включает тысячи серверов и ВМ, а это тысячи временных последовательностей.

Традиционные статистические методы прогнозирования, будучи ориентированными на упрощенные модели, часто не позволяют выявить неявные зависимости в данных. Как следствие, точность их прогнозов при исследовании сложных систем, таких как облачные ЦОД, оказывается недостаточно высокой. Сложность облачной среды, характеризующейся большим числом разнородных показателей, требует применения методов, способных учитывать всю совокупность имеющихся параметров. По этой причине в данной работе предлагается использовать метод группового учета аргументов (МГУА). Метод был разработан в 1968 году академиком А.Г. Ивахненко [112]. Классический МГУА принадлежит к числу эволюционных алгоритмов искусственного интеллекта. МГУА обладает преимуществами, когда отсутствует или почти отсутствует априорная информация о структуре модели и распределении ее параметров. Кроме этого, по сравнению с нейронными сетями, МГУА может использоваться, когда данных наблюдений крайне мало, вплоть до того, что параметров модели больше, чем число наблюдений [113].

3.3.3 Метод группового учета аргументов

Постановка задачи моделирования следующая. Пусть имеется выборка из N наблюдений

$$\{X(1),Y(1)\}$$
$$\{X(2),Y(2)\}$$
$$\cdots$$
$$\{X(N),Y(N)\}$$

Наиболее полная зависимость между входами $X(i)$ и выходами $Y(i)$ может быть представлена с помощью обобщенного полинома Колмогорова-Габора. Пусть $X = \{x_1, \cdots, x_M\}$ — вектор входных переменных (аргументов), тогда такой полином имеет вид:

$$Y(x_1,...,x_M) = a_0 + \sum_{i=1}^{M} a_i x_i + \sum_{i=1}^{M}\sum_{j=1}^{M} a_{ij} x_i x_j + \sum_{i=1}^{M}\sum_{j=i}^{M}\sum_{k=j}^{M} a_{ijk} x_i x_j x_k + ..., \quad (3.3)$$

где все коэффициенты a неизвестны.

В процессе идентификации модели (определения числовых значений коэффициентов) в качестве критерия селекции наиболее часто применяется минимум критерия регулярности, который записывается следующим образом:

$$\bar{\varepsilon^2} = \frac{1}{N}\sum_{i=1}^{N}(y_i - f_i(x_i))^2, \ \bar{\varepsilon^2} \to min. \quad (3.4)$$

На одной и той же выборке можно построить множество моделей, обеспечивающих нулевую ошибку на обучающих данных. Для этого достаточно повышать степень полинома: при наличии N узлов интерполяции существует целое семейство полиномиальных моделей, каждая из которых будет точно проходить через все экспериментальные точки, давая нулевую ошибку $\bar{\varepsilon^2} = 0$.

Обычно степень нелинейности берут не выше $N - 1$.

Обозначим S—сложность модели, которая определяется числом членов полинома Колмогорова-Габора.

Ошибка $\bar{\varepsilon^2}$ нелинейно зависит от сложности модели: она снижается при усложнении модели до некоторой точки, после которой начинает расти. Это обуславливает существование оптимальной сложности, обеспечивающей минимальную ошибку.

В условиях зашумленных данных проявляются следующие эффекты (рис. 3.3):

1. общая форма зависимости (падение, затем рост) сохраняется при любом уровне шума;
2. величина минимальной ошибки $\min_S \bar{\varepsilon}^2$ монотонно возрастает с увеличением уровня помех;
3. оптимальная сложность S_0 сдвигается влево (уменьшается) по мере роста помех, причем $S_0 = \arg\min \bar{\varepsilon}^2$ остается положительной при ненулевом шуме.

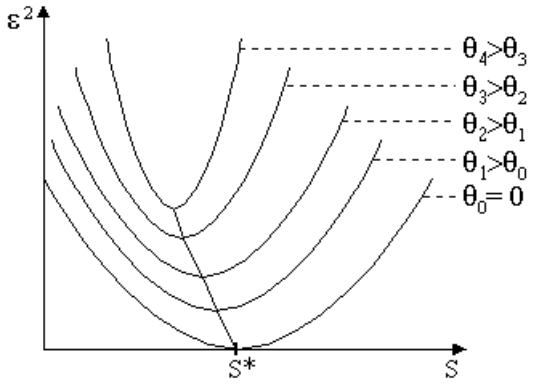

Рисунок 3.3 — Поиск оптимальной сложности модели

Выборка по определению всегда ограничена и не отражает всей полноты генеральной совокупности. Для компенсации этого недостатка в методе группового учета аргументов используется внешнее дополнение — дополнительная (проверочная) выборка, точки которой не участвуют в оценивании коэффициентов полинома Колмогорова–Габора. Процедура поиска оптимальной модели включает следующие шаги:

1. Исходная выборка объема N разбивается на три части: обучающую (N_A), тестовую (N_B) и экзаменационную (N_C).
2. По обучающей выборке N_A оцениваются коэффициенты модели $\hat{a_0}, \hat{a_i}, \hat{a_{ij}}$.
3. На тестовой выборке N_B проводится селекция — отбирается несколько лучших моделей-претендентов.
4. Для окончательного выбора на экзаменационной выборке N_C вычисляется значение внешнего критерия (например, 3.4), и модель с наилучшим значением признается оптимальной.

В некоторых модификациях алгоритма этап проверки на экзаменационной выборке может отсутствовать.

В настоящее время разработано и исследовано много разновидностей алгоритмов МГУА переборного [112] и итерационного [114] типов. Комбинаторные алгоритмы перебирают все модели из заданного базиса, выбирая лучшую по критерию селекции. Они эффективны для структурной идентификации, но только при малом числе аргументов из-за взрывного роста числа вариантов. Итерационные алгоритмы работоспособны при достаточно большом количестве аргументов, но специфика их архитектуры не гарантирует построения модели истинной структуры, поскольку они базируются на неполных индуктивных процедурах иерархического усложнения моделей. Существуют также гибридные алгоритмы МГУА [113].

Для цели прогнозирования перегрузки серверов выбран комбинаторный алгоритм МГУА, поскольку он имеет особенности, позволяющие улучшить прогнозирование моделей сложных объектов и придать им объективный характер:

1. вид модели алгоритм выбирает сам, а не исследователь, т.е. осуществляется селекция модели с использованием внешнего критерия;
2. возможность прогнозирования при неполном информационном базисе;
3. самоорганизация физической и прогнозирующих моделей возможна при сильно зашумленных исходных данных.

3.4 Сравнение алгоритмов прогнозирования

3.4.1 Критерии эффективности

В силу специфики облачной модели, предоставляющей пользователям абстракцию неограниченных ресурсов, алгоритмы распределения должны тестироваться в условиях крупномасштабных виртуализированных сред. Однако проведение повторяемых экспериментов на реальных ЦОД такого масштаба сопряжено с серьезными трудностями.

Решением является использование имитационного моделирования. Платформа CloudSim де-факто стала стандартным инструментом для сравнительного анализа алгоритмов управления ресурсами в облачных ЦОД. Это

подтверждается многочисленными обзорами, которые опираются на CloudSim и применяют его классификационную схему.

Библиотека CloudSim предоставляет собой набор Java-классов для описания облачного ЦОД и проведения самого имитационного моделирования. Моделирование проводилось на компьютере Intel Atom N2800 (1.8 ГГц), 2 ГБ ОЗУ с операционной системой Windows 7 и установленной средой разработки Eclipse MARS 2 и виртуальной машиной Java. Для построения модели были дополнены ряд классов. Детали построения имитационных моделей на данной платформе приведены в главе 5. В данном параграфе основное внимание будет уделено основным моментам, связанным с моделированием и анализом результатов.

Необходимо определить критерии, по которым будут сравниваться результаты моделирования с различными исходными данными. Логично предположить, что это должны быть показатели качества, задаваемые в SLA-соглашениях, а также показатели важные для облачного провайдера, такие как энергопотребление ЦОД, число миграций виртуальных машин.

Требования на качество обслуживания, формализуемые в виде SLA-соглашений, могут меняться для различных приложений, поэтому важно определить метрики, не зависящие от приложения, которые могут быть использованы для оценки качества обслуживания для любой ВМ в инфраструктуре IaaS.

В работе [79] были предложены две метрики нарушения SLA в среде IaaS.

1. Доля времени, в течение которого активные хосты испытывают загрузку 100 % (Overload Time Fraction—OTF):

$$OTF = \frac{1}{N}\sum_{i=1}^{N}\frac{T_{si}}{T_{ai}}, \tag{3.5}$$

где N—общее количество физических серверов в кластере;

T_{si}—суммарная продолжительность периодов, в течение которых хост i функционировал в режиме полной (100%) загрузки, что приводило к нарушениям SLA-соглашений;

T_{ai}—общее время нахождения хоста i в активном состоянии (т.е. время, в течение которого на нем выполнялись виртуальные машины).

2. Общее ухудшение производительности виртуальных машин из-за миграций (Performance Degradation due to Migrations–PDM).

$$PDM = \frac{1}{M}\sum_{j=1}^{M}\frac{C_{rj}}{C_{dj}}, \tag{3.6}$$

где M—общее количество виртуальных машин в системе;

C_{dj} — показатель снижения производительности j виртуальной машины, обусловленного процессами миграции;

C_{rj} —суммарный объем процессорных ресурсов (в MIPS), потребленных j-й ВМ за все время ее существования.

В рамках данной работы величина C_{dj} принимается равной 10 % от загрузки CPU (в MIPS), зафиксированной в периоды выполнения всех миграций виртуальной машины j.

Предполагается, что SLA-соглашения выполняются, когда ВМ предоставляется вся запрашиваемая производительность в соответствии с параметрами самой ВМ.

Если хост, обслуживающий приложения, испытывает 100-процентную загрузку, то производительность приложений ограничена возможностями хоста, поэтому ВМ не предоставляется требуемый уровень производительности.

Оба показателя (OTF и PDM) независимо характеризуют уровень нарушений SLA в системе, поэтому предложен комбинированный критерий, учитывающий оба этих показателя нарушения SLA, обозначенный (SLA Violation–SLAV). Данный критерий рассчитывается следующим образом:

$$SLAV = OTF \cdot PDM. \tag{3.7}$$

Поскольку потребление энергии E и SLAV обычно являются отрицательно коррелированными показателями, авторы работы [79] предложили комбинированный критерий: ESV (Energy and SLA Violations), который рассчитывается как

$$ESV = E \cdot SLAV. \tag{3.8}$$

3.4.2 Параметры имитационной модели

Моделирование проводилось для ЦОД, состоящего из 800 физических серверов, половина из которых HP ProLiant ML110 G4 и другая половина HP ProLiant ML110 G5. Тактовая частота процессоров измерялась в миллионах инструкций в секунду (MIPS), 1860 MIPS каждое ядро для сервера HP ProLiant ML110 G4 и 2660 MIPS для сервера HP ProLiant ML110 G5. Каждый сервер имеет пропускную способность 1 Гбит/с.

Характеристики используемых в моделировании виртуальных машин соответствуют типам Amazon EC2, однако с одним отличием: все ВМ являются одноядерными. Это обусловлено тем, что исходные данные о нагрузке, применявшиеся в имитационных экспериментах, были собраны с виртуальных машин, имеющих одно ядро.

При моделировании использовались ВМ следующих типов:

– высокоскоростной средний экземпляр (2500 MIPS, 0,85 Гб ОЗУ);
– большой экземпляр (2000 MIPS; 3,75 Гб ОЗУ);
– малый экземпляр (1000 MIPS; 1,7 Гб ОЗУ);
– микро экземпляр (500 MIPS; 613 Гб ОЗУ).

Первоначально все ВМ были размещены в соответствии со своими потребностями в ресурсах, определенные типом ВМ. Однако с течением времени ВМ используют меньше ресурсов в соответствии с нагрузкой, что дает возможность их динамического размещения на меньшем числе серверов. Энергопотребление серверов при различных уровнях рабочей нагрузки приведено в таблице 15.

Были использованы файлы, содержащие рабочую нагрузку, имеющуюся в CloudSim, собранную с более чем 1000 виртуальных машин проекта PlanetLab в течение случайно выбранных 10 дней. Интервал измерения нагрузки составляет 5 минут. Параметры рабочей нагрузки приведены в таблице 14. Также для эксперимента была взята нагрузка серверов Alibaba [115].

Для проверки эффективности прогнозирования перегрузки серверов методом МГУА была разработана имитационная модель, в основу которой положена модель энергоэффективного ЦОД [79]. В данный пакет был включен МГУА, который использовался для прогнозирования состояния сервера на следующем шаге наблюдения. Метод МГУА для прогнозирования перегрузки хоста был зарегистрирован в государственном реестре программного обеспечения [116].

3.4.3 Анализ результатов

В начале процесса моделирования виртуальные машины на каждом сервере свободно размещались 1-2 виртуальные машины. В процессе моделирования осуществлялось уплотнение виртуальных машин на меньшем числе серверов и незагруженные серверы для экономии электроэнергии выключались.

В экспериментах размер окна варьировался от 7 до 15 точек. При прогнозировании загрузки хоста свыше 85% принимается решение о миграции виртуальной машины. Размер обучающей выборки в экспериментах составил 2/3 от всего объема. Размер экзаменационной выборки составил две последние точки. Метод группового учета аргументов сравнивался с линейными моделями, параметры которых оценивались методом локальной регрессии и методом наименьших квадратов. Виртуальные машины для миграции выбирались по минимуму оперативной памяти в соответствии с алгоритмом «минимальное время на миграцию» (Minimum Migration Time — MMT) [79].

Результаты экспериментов для размера окна из 10 точек приведены в таблице 6.

Таблица 6 — Сравнение методов прогнозирования перегрузки виртуальных машин

Политика	**ESV** ($\times 10^{-3}$)	**Энергопотр.** (кВт×ч)	**SLAV** ($\times 10^{-5}$)	**OTF**	**PDM**	**Число миграций** ($\times 10^3$)	**Число обр. миграций** ($\times 10^3$)
Лок.регр.	0,17	163,15	0,11	0,14	0,08	27,63	4,6
МНК	0,31	185,17	0,25	0,15	0,17	33,86	5,46
МГУА	0,16	165,25	0,1	0,1	0,1	29,04	4,6

Таким образом, использование комбинаторного алгоритма МГУА для прогнозирования перегрузки хостов на большинстве исследованных выборок превосходит метод локальной регрессии и метод наименьших квадратов.

Анализ изменения трендов загрузки серверов назначения после размещения ВМ позволяет внести дополнительное условие для выбора лучшего сервера для миграции, что может значительно повысит устойчивость процесса миграции виртуальных машин.

3.4.4 Выбор длины скользящего окна

Для повышения точности прогнозирования при мониторинге состояния серверов применяется метод скользящего окна. В границах каждого окна фиксируются значения загрузки процессора, которые затем усредняются. Такой подход позволяет снизить частоту ложных срабатываний, обусловленных кратковременными случайными колебаниями нагрузки.

Как уже упоминалось важным вопросом при использовании метода скользящего окна является выбор длины окна.

Окно небольшого размера приводит к быстрому и агрессивному наблюдению, в то время как большие значения вызывают медленное наблюдение, но в то же время является фильтром к случайному кратковременному нарушению порогов, что позволяет избежать ненужных миграций. Поэтому выбор размера окна оказывает влияние на эффективность наблюдений.

В связи с некачественным прогнозом может также возникнуть ситуация, когда сервер, с которого будет мигрирована виртуальная машина, вновь примет ВМ на следующем шаге наблюдения, т.е. возникнет обратная миграция.

В общем случае размер окна наблюдения должен быть больше длительности миграции, поскольку миграция вносит помехи в процесс наблюдения, дополнительно загружая процессор сервера и сеть.

В ряде работ рассмотрена проблема прогнозирования длительности миграции, в том числе путем определения ее плотности вероятности.

Используя функцию плотности распределения длительности миграции, введем коэффициент устойчивости. Он определяется как отношение длительности миграции к размеру скользящего окна наблюдения: $t_{\text{migration}}/t_{\text{window}}$, где $t_{\text{migration}}$ — 90-процентный квантиль распределения времени миграции, т.е. такое значение, что

$$P[t \leqslant t_{\text{migration}}] = 0{,}9.$$

Это означает, что в 90% случаев реальная длительность миграции будет меньше или равна $t_{\text{migration}}$.

Для обоснования размера скользящего окна в имитационную модель энергоэффективного ЦОД были внесены следующие изменения.

Таблица 7 — Влияние длительности миграции на качество прогноза

Метод	Длительность миграции (мин.)	Число миграций	Число обратных миграций	Общее энергопотр.	Нарушение SLA
ЛР	1	8048	1200	50,93	9,89
ЛР	2	9123	2497	62,35	12,76
ЛР	3	11486	3578	67,35	14,87
ЛР	4	15527	5679	72,63	16,42
МГУА	1	8100	1225	51,93	10,54
МГУА	2	9065	2365	60,5	11,54
МГУА	3	11786	3778	69,62	15,35
МГУА	4	15588	5643	73,77	16,11

1. Возможность отслеживания числа обратных миграций. При этом понимание обратной миграции было шире, чем просто возврат одной и той же машины на тот же сервер. Пусть с сервера А на сервер Б в окне w_1 была перенесена ВМ1, тогда обратная миграция будет зафиксирована, если в следующем окне с сервера Б приходится переносить любую ВМ.
2. Для моделирования влияния миграции на загрузку серверов период наблюдения был разбит на 10 временных интервалов (точек), по которым впоследствии рассчитывалась средняя загрузка процессора за окно. В каждый из этих интервалов вносились случайные возмущения на уровне 5–15%, имитирующие флуктуации загрузки CPU, обусловленные процессом миграции.

С учетом того, что окно наблюдения в реальной системе имеет фиксированную длину 5 минут, моделирование выполнялось для четырех значений 90-процентного квантиля длительности миграции: 1, 2, 3 и 4 минуты. Полученные данные о влиянии миграции на точность прогнозов и результирующие показатели работы ЦОД сведены в таблицу 7. Для прогноза методом локальной регрессии (ЛР) и МГУА.

Таким образом, продолжительность миграции, составляющая более 1/3 окна наблюдения в эксперименте, отрицательно влияет на качество прогноза. Поэтому целесообразно поддерживать коэффициент устойчивости на данном уровне.

Отношение средней длительности миграции к ширине скользящего окна может служить показателем устойчивости процесса живой миграции.

3.5 Метода расчета длительности миграции и длительности простоя виртуальных машин

3.5.1 Виды миграции виртуальных машин

Остановка и копирование (Stop and Copy)

Данный вид миграции отличается простотой реализации, однако требует полной остановки виртуальной машины на время переноса. Процесс заключается в копировании всех страниц памяти на целевой сервер, после чего ВМ возобновляет работу. Длительность такой миграции относительно невелика по сравнению с другими типами и прямо пропорциональна объему оперативной памяти переносимой ВМ.

Основным недостатком является простой сервиса, продолжительность которого может быть значительной при большом объеме памяти. Как отмечается в [117], это делает данный подход неприемлемым для большинства критичных к доступности приложений.

Предварительное копирование (Pre-copy)

Данный вид миграции используется в большинстве гипервизоров, таких как Xen, VMware и KVM. Гипервизор изначально копирует все страницы памяти исходной виртуальной машины на виртуальную машину сервера назначения, при этом операционная система и все сервисы исходной ВМ остаются в рабочем состоянии. После этого докопируются измененные станицы памяти за несколько итераций. Исходная ВМ останавливается и все оставшиеся страницы передаются на сервер назначения. Затем все службы на сервере назначения возобновляются. Этот подход имеет преимущество перед полной остановкой ВМ, поскольку уменьшает время её простоя. Однако механизм докопирования измененных страниц памяти, лежащий в основе живой миграции, создает

дополнительную нагрузку на сетевую инфраструктуру и увеличивает общую продолжительность переноса. Платой за это является максимально возможная непрерывность работы приложения с точки зрения пользователя.

Для повышения производительности миграции с предварительным копированием могут применяться дросселирование процессора, дельта-сжатие и сжатие данных.

Дросселирование процессора [118] – это метод намеренного сокращения выделяемого виртуальной машине процессорного времени для того, чтобы понизить скорость изменения страниц памяти и тем самым ускорить сходимость процесса итерационного копирования.

Дельта-сжатие (DLTC) [119] является еще одной модификацией миграции с предварительным копированием, применяющее во время живой миграции сжатие методом RLE к измененной части страниц памяти, называемой дельтой. Этот метод может потребовать значительного объема дополнительной памяти для хранения страниц памяти для вычисления их дельты.

Оптимизация сжатия данных (DTC) [120] сжимает страницы памяти перед передачей. Этот метод требует значительного объема дополнительных вычислительных ресурсов и, таким образом, может не являться жизнеспособным вариантом, если загрузка процессора на сервере высока.

Пост-копирование (Post-copy)

При данном подходе значения регистров процессора немедленно переносятся на виртуальную машину на сервере назначения. ВМ на сервере сразу же стартует. Остальная память будет скопирована либо в фоновом режиме или по требованию. В отличие от предварительного копирования, при пост-миграции ВМ запускается на целевом сервере практически сразу, а страницы памяти передаются в фоновом режиме. Если приложению требуется страница, еще не успевшая скопироваться, возникает исключительная ситуация, и гипервизор загружает эту страницу с исходного узла в приоритетном порядке. Такая стратегия позволяет добиться минимального времени недоступности ВМ, однако расплатой становятся: увеличение суммарной длительности миграции;

потенциальное возрастание задержек при доступе к памяти в процессе работы приложения. Этапы миграции с пост-копированием приведены на рис. 3.4.

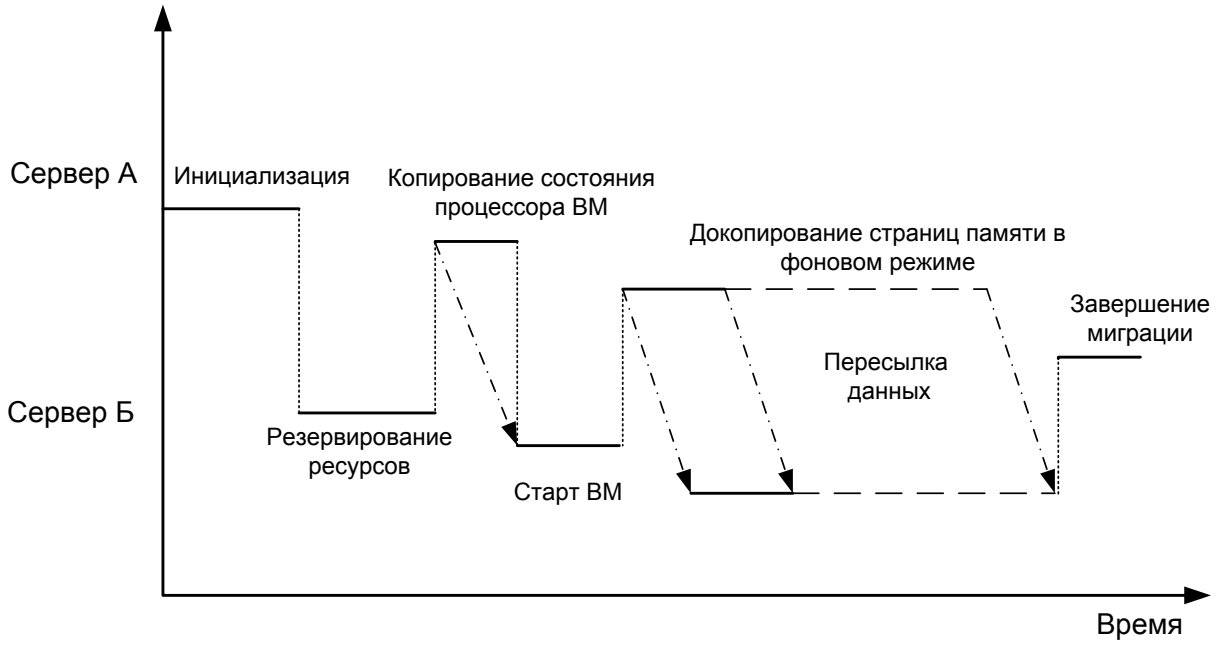

Рисунок 3.4 — Длительность миграции с пост копированием

Гибридные миграции

Поскольку различные виды миграции имеют свои преимущества и недостатки, то за последнее время появилось множество работ, в которых исследовались различные гибридные виды миграции [121—123].

3.5.2 Характеристики миграции виртуальных машин

Производительность живой миграции оценивается следующими показателями [30]:

– **Время подготовки** — время между началом миграции и передачи состояния процессора виртуальной машины на узел назначения, в течение которого ВМ продолжает выполняться и модифицировать страницы памяти. Для миграции с предварительным копированием это время

включает в себя весь этап итерационного копирования памяти, тогда как для пост-копирования это время несущественно.

- **Время простоя** — это время, в течение которого мигрирующая виртуальная машина останавливается. Как минимум, это время включает в себя передачу регистров процессора. Для миграции с предварительным копированием за это время передаются оставшиеся измененные страницы памяти. Для пост-копирования передаются некоторые другие параметры, если таковые имеются, для запуска ВМ на сервере назначения.
- **Время возобновления** — это время на возобновление работы виртуальной машины на целевом сервере. Можно назвать этот промежуток времени окончанием миграции: все зависимости от источника должны быть удалены. Для предварительного копирования нужно только переназначить целевую ВМ и уничтожить источник копирования. С другой стороны, при пост-копировании в этот период продолжается работа по копированию страниц памяти с исходной ВМ в фоновом режиме.
- **Число перемещенных страниц** — это общее число перемещенных страниц памяти, в том числе повторяющихся за все время миграции. При подходе с предварительным копированием большинство страниц перемещается во время подготовки, а при пост-копировании большинство страниц перемещается во время возобновления.
- **Общая длительность миграции**—это суммарная длительность всех перечисленных этапов миграции от начала до конца. Общее время является важным, поскольку оно влияет на освобождение ресурсов на обоих участвующих серверах, а также внутри самих ВМ на обоих узлах. До завершения миграции нельзя освободить память виртуальной машины на узле-источнике.
- **Ухудшение производительности приложения**—показатель, характеризующий степень замедления работы приложения, выполняющегося на виртуальной машине, в процессе миграции. В методе предварительного копирования причиной деградации является необходимость отслеживания модифицируемых (загрязненных) страниц памяти, что существенно сказывается на производительности приложений с интенсивной записью. В методе пост-копирования замедление обусловлено подкачкой недостающих страниц по требованию.

Для полноты оценки системы динамического управления ресурсами целесообразно учитывать:

– **Общее число миграций**. Экспериментально подтверждена корреляция между точностью прогнозирования перегрузок и количеством выполняемых миграций: ошибки прогноза провоцируют избыточные перемещения. Кроме того, значение порога перегрузки обратно пропорционально числу миграций, но прямо пропорционально риску нарушения SLA [79].
– **Частота возвратных миграций («пинг-понг эффект»)**. Миграция считается нерезультативной, если принявший сервер вскоре после переноса ВМ сам оказывается перегруженным и требует разгрузки. Высокий уровень возвратов не только увеличивает общее число миграций, но и свидетельствует о нестабильности системы, которая начинает «переливать» ВМ между хостами без достижения устойчивого состояния. Ключевым способом предотвращения таких ситуаций является прогнозирование загрузки целевого сервера после размещения [124].

3.5.3 Этапы миграции с предварительным копированием

Наиболее распространенным видом миграции, реализуемым в большинстве гипервизоров, таких как Xen, VMware и KVM является подход с предварительным копированием (pre-copy migration). Длительность такой миграции состоит из нескольких этапов (рис. 3.5) [30].

Этап 0. Инициализация. На сервере А находится активная виртуальная машина. Выбирается сервер назначения Б для миграции. Блочные устройства на сервер Б зеркалируются и свободные ресурсы резервируются.

Этап 1. Резервирование. Инициализация контейнера на хосте назначения.

Этап 2. Итерационное копирование. За первую итерацию копируются все страницы памяти из сервера А на сервер Б. В последующих итерациях передаются только копии тех страниц, которые были изменены во время предыдущей фазы передачи. Число итераций ограничивается в настройках

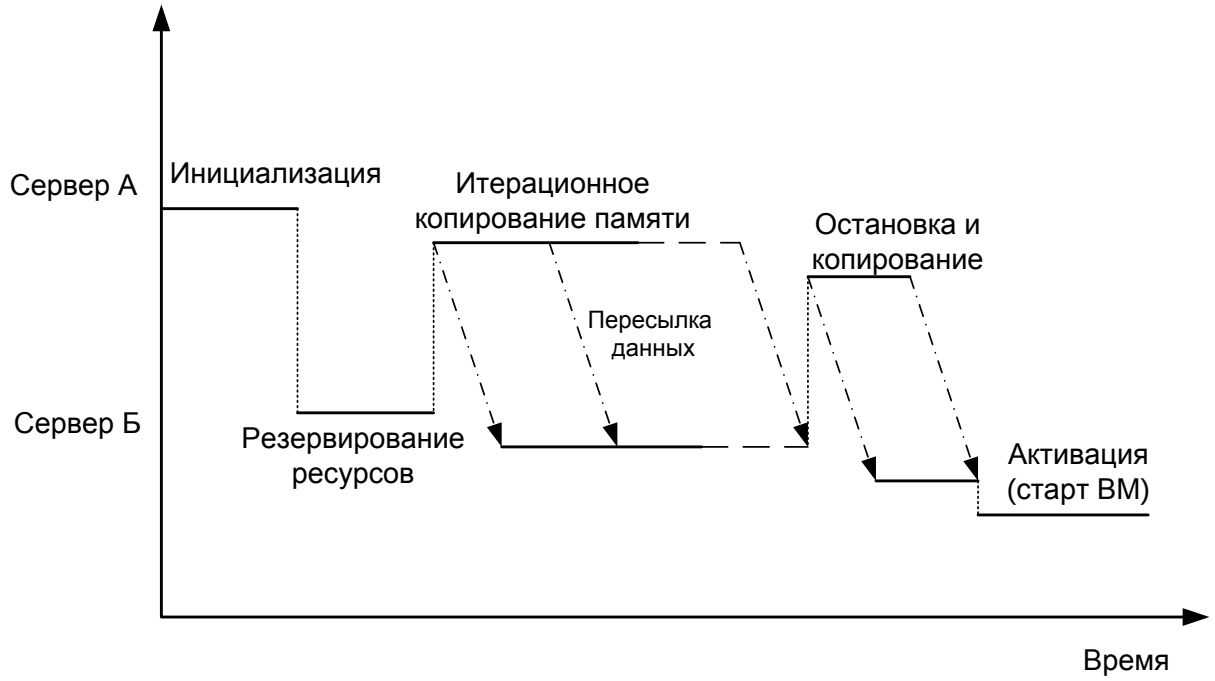

Рисунок 3.5 — Длительность миграции с предварительным копированием

гипервизора. Например, в гипервизоре Xen максимальное число итераций составляет 29 раз.

Этап 3. Остановка и копирование. На сервере А приостанавливается запущенный экземпляр ОС и весь сетевой трафик переключается на сервер Б. Передаются регистры и кэш-память процессора, а также все оставшиеся несогласованные страницы памяти. В конце этой стадии существуют согласованные приостановленные копии виртуальных машин на хостах А и Б. Копия А по-прежнему считается основной и возобновляется в случае сбоя.

Этап 4. Завершение. Сервер Б сообщает серверу А на то, что он успешно принял согласованный образ ОС. Сервер А теперь может отказаться от исходной виртуальной машины, а сервер Б становится основным хостом.

Этап 5. Активация. Перемещенная виртуальная машина активируется. Происходит подключение к локальным устройствам, а также конфигурация сетевого окружения.

Общая длительность миграции является суммарным временем всех этапов миграции:

$$\tau = T_P + T_R + \sum_i T_{PCi} + T_{SC} = T_C + T_A, \tag{3.9}$$

где T_P—время на инициализацию;

T_R—время на резервирование ресурсов;

T_{PCi}—время на предварительное копирование в i-ой итерации;

T_{SC}—время на предварительное копирование в i-ой итерации;

T_C—время на завершение;

T_A—время на активацию.

В данной формуле есть детерминированные и случайные составляющие миграции. К детерминированным относятся все операции кроме итерационного копирования, их длительность может незначительно варьироваться, но они всегда присутствуют. Длительность итерационного копирования (случайной составляющей) зависит от пропускной способности сети, объема оперативной памяти и скорости модификации страниц памяти.

3.5.4 Обзор существующих методов оценки длительности миграции

Вопросам прогнозирования длительности миграции посвящено ряд работ. В частности, в работе [125] предложен метод, позволяющий минимизировать длительность простоя виртуальной машины в течение миграции за счет нахождения оптимального числа итераций докопирования. Метод основывается на реализации двух алгоритмов: прогнозирования числа итераций докопирования модифицированных страниц памяти и уменьшения размера требуемой оперативной памяти при выполнении миграции.

Модель, предложенная в работе [126], позволяет оценить ключевые показатели живой миграции ВМ: длительность простоя, объем передаваемых данных и утилизацию сети. На ее основе авторы разработали нелинейную оптимизационную модель, нацеленную на минимизацию времени простоя при ограничениях на пропускную способность канала. В отличие от подхода [126], где акцент сделан на простое, исследование [127] фокусируется на минимизации полного времени миграции.

Для прогнозирования характеристик миграции (общее время, простой, сетевой трафик, деградация приложений, загрузка CPU и памяти) в работе [127] используются методы машинного обучения. Однако практическая реализация этого подхода затруднена высокой трудоемкостью сбора данных: обучение и тестирование регрессионной модели требуют 18 различных входных параметров.

В работах [123; 128] предложены аналитические модели для оценки средней длительности миграции виртуальных машин и времени простоя сервиса. Данные модели учитывают объем оперативной памяти, скорость изменения страниц памяти и пропускную способность сети. Исследователи используют детерминированный подход, устанавливающий функциональные зависимости между двумя ключевыми параметрами, влияющими на время миграции: скоростью изменения страниц памяти и пропускной способностью сети. Однако расчет общей длительности миграции в этих работах выполняется на основе средних значений. Такой детерминированный подход не позволяет оценить вероятность достижения тех или иных значений времени миграции. Решение этой задачи требует знания закона распределения вероятностей, который дает наиболее полную вероятностную характеристику процесса.

Процесс миграции по своей природе — это случайный процесс, который представляется в виде последовательности операций, каждая из которых является случайной величиной.

Получить плотность распределения вероятности можно путем обработки результатов наблюдений и выдвижения статистических гипотез. При этом виды эмпирических распределений зависят от характера обрабатываемых приложений и разброс очень значителен, поэтому путь поиска ответа на вопрос какова вероятность длительности миграции классическими методами обработки статистики очень трудоемок, длителен и требует значительных вычислительных ресурсов.

Поэтому в данной работе предлагается другой подход к ответу на поставленный вопрос, так называемый аналитический подход, сущность которого состоит в поиске формулы плотности распределения, которая положена в основу метода расчета длительности миграции для различных её типов с учетом характера приложений.

3.5.5 Набор данных миграций виртуальных машин

Анализ длительности миграции выполнен на основе набора данных, представленного в работе Changyeon Jo [127]. Данный набор содержит более 40 000 записей миграций виртуальных машин, выполненных с использованием пя-

Таблица 8 — Распределение общего времени миграции

Тип миграции	**Мин.**	**1 кв.**	**Медиана**	**Среднее**	**3 кв.**	**Макс.**
Precopy	2.346	10.374	20.159	28.275	36.594	263.527
THR	2.336	9.835	19.491	27.537	34.409	179.746
DLTC	2.386	8.098	15.427	18.669	24.762	166.771
DTC	3.605	16.446	34.518	57.328	69.987	388.641
POST	2.327	6.395	10.163	11.985	16.069	78.643

ти различных алгоритмов динамической миграции: пост-копирования (POST), предварительного копирования (PRE), а также трех его модификаций — с дросселированием ЦП (THR), дельта-сжатием (DLTC) и сжатием данных (DTC). На каждую из пяти стратегий приходится примерно по 8000 записей. Распределение количества миграций по типам алгоритмов представлено на рис. 3.6.

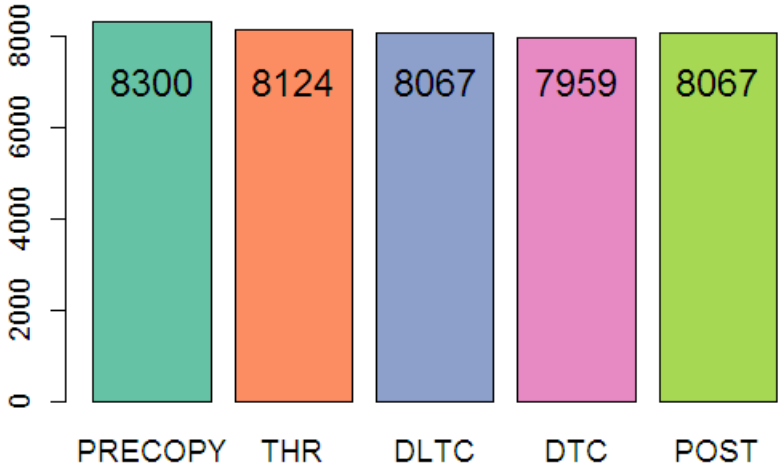

Рисунок 3.6 — Число миграций различных видов в датасете

Гистограммы распределений длительности миграций различных видов приведено в приложении А на рис. А.1. Анализ гистограмм показывает, что все распределения являются одномодальными, характеризуются положительной асимметрией и наличием длинного правого хвоста. Основные числовые характеристики распределений сведены в таблицу 8.

Миграция с пост-копированием (POST) демонстрирует наименьшее общее время миграции и минимальное время простоя среди всех рассматриваемых

алгоритмов. Однако, с точки зрения снижения производительности виртуальной машины, POST показывает наихудшие результаты: средняя деградация производительности составляет 30% [127]. Миграция с дельта-сжатием страниц памяти (DLTC) позволяет сократить общее время миграции по сравнению с предварительным копированием (PRE) и дросселированием ЦП (THR), но при этом создает дополнительную нагрузку на процессоры. Что касается алгоритма сжатия данных (DTC), то он демонстрирует худшие показатели по общему времени миграции.

Производительность алгоритмов живой миграции не только сильно различается между различными алгоритмами, но и сильно зависит от рабочих нагрузок, выполняемых внутри виртуальной машины.

В наборе использовалось 37 рабочих нагрузок, полученных из тестовых наборов и приложений, моделирующих реальные сценарии. Нагрузки включают следующие наборы тестов:

- SPECWeb — для имитации работы веб-сервера, обслуживающего банковские сервисы и услуги электронной коммерции;
- OLTPBench — приложение для обработки онлайн-транзакций (OLTP) в базах данных;
- Memcached — хранилище кэша в формате «ключ-значение» в оперативной памяти;
- DaCapo — набор Java-приложений;
- PARSEC — набор многопоточных нагрузок;
- Bzip2 — приложение с интенсивными вычислениями и операциями ввода-вывода;
- mplayer — имитация работы мультимедийного проигрывателя;
- synthetic — синтетические нагрузки;
- idle — отсутствие нагрузки.

Число миграций для различных типов нагрузок приведено на рис. 3.7.

Диаграммы размаха позволяют наглядно оценить различия в общем времени миграции для разных алгоритмов. Соответствующие диаграммы, представленные в Приложении А на рис. А.2, демонстрируют, как распределяется общее время миграции в зависимости от типа рабочей нагрузки. Анализ показывает, что некоторые приложения (например, idle, memcached, а в ряде случаев specweb и oltp) характеризуются незначительной случайной вариативностью

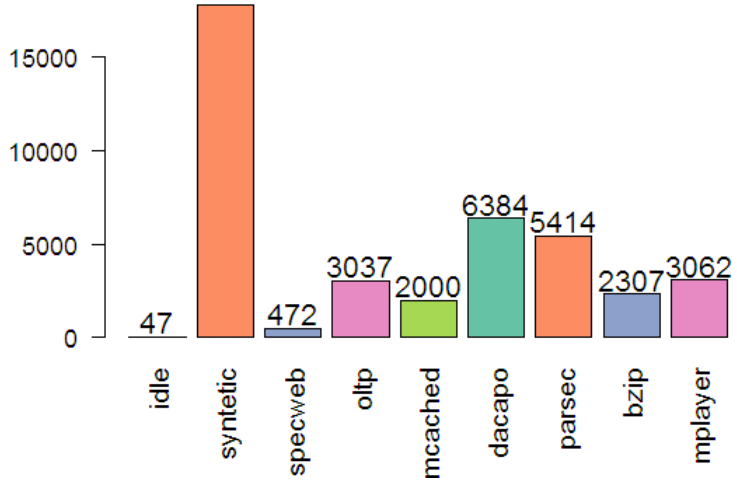

Рисунок 3.7 — Число миграций для различных типов нагрузок

времени миграции, в то время как для других нагрузок наблюдается высокая степень случайности.

Таким образом, вид приложения существенно влияет как на форму эмпирического распределения (гистограммы) времени миграции, так и на его числовые характеристики. Данное наблюдение обосновывает необходимость получения аналитического выражения для закона распределения длительности миграции, учитывающего специфику различных типов нагрузок.

3.5.6 Аппроксимация плотности вероятности рядами Грама-Шарлье и Лагерра

Функции распределения, близкие к нормальному, которые характеризуются одновершинностью и достаточно быстрым убыванием хвостов (ветвей) при возрастании аргумента, удобно аппроксимировать с помощью полиномов Чебышева—Эрмита и полиномов Лагерра [129].

Плотность вероятности может быть представлена в виде бесконечного ряда Эджворта, который строится на основе полиномов Чебышева—Эрмита. Первые четыре члена этого ряда образуют ряд Грама—Шарлье типа А [129;

130], который имеет следующий вид:

$$W_1(\xi) = \frac{1}{\sigma}\left[\Phi'(\frac{\xi - m}{\sigma}) - \frac{\hat{S}_k}{3!}\Phi^4(\frac{\xi - m}{\sigma}) + \frac{\hat{E}_k}{4!}\Phi^5(\frac{\xi - m}{\sigma})\right] \tag{3.10}$$

где $\Phi'(z) = \Phi'(\frac{\xi - m}{\sigma})$ — производная от функции нормального распределения центрированной и нормированной случайной величины;

$\Phi^4(z)$ и $\Phi^5(z)$ — производные четвертого и пятого порядков от функции Лапласа Φ(z);

$\hat{S}_k$—оценка ассиметрии;

$\hat{E}_x$—оценка эксцесса.

Таким образом, зная первый начальный момент, а также второй, третий и четвертый центральные моменты длительности миграции, можно получить приближенное выражение для ее закона распределения.

Анализ имеющихся данных показывает, что ряд Грама-Шарлье обеспечивает приемлемую аппроксимацию для выборок с небольшой асимметрией. Однако, как видно на рис. 3.8 и рис. 3.9, эмпирические функции распределения в данном случае характеризуются высоким коэффициентом асимметрии. Более точную аппроксимацию может дать ряд Лагерра, поскольку случайную величину длительности миграции можно представить как сумму независимых положительных случайных величин.

Ряд Лагерра имеет следующий вид:

$$W_1(\xi) = \sum_{n=0}^{\infty} c_n e^{-\xi}\xi^{\alpha} L_n^{(\alpha)}(\xi), \tag{3.11}$$

где $L_n^{(\alpha)}(\xi)$—обобщенный полином Лагерра

$$L_n^{(\alpha)}(z) = e^z \frac{z^{-\alpha}}{n!} \cdot \frac{d^n}{dz^n}(e^{-z}z^{n+\alpha}), \alpha > -1. \tag{3.12}$$

В связи с трудностью вычислений обычно ряд Лагерра применяют, если уже первый член дает достаточно хорошее приближение, который имеет следующий вид:

$$W_1(\xi) = \frac{1}{\beta\Gamma(\alpha + 1)}\left(\frac{\xi}{\beta}\right)^{\alpha} e^{(-\frac{\xi}{\beta})}, \tag{3.13}$$

где

$$\alpha = \frac{m^2}{\sigma^2} - 1, \tag{3.14}$$

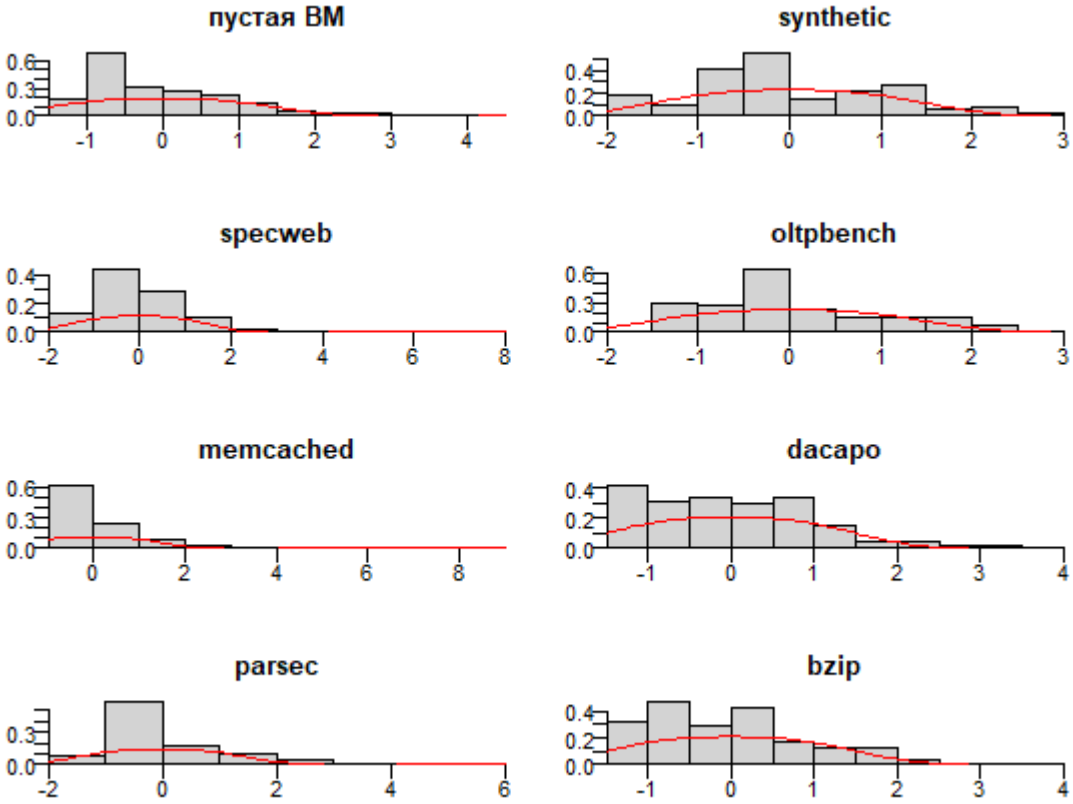

Рисунок 3.8 — Аппроксимация длительности миграции рядом Грама-Шарлье для миграции с дросселированием процессора

$$\beta = \frac{\sigma^2}{m}, \tag{3.15}$$

m—оценка математического ожидания и σ^2—дисперсия случайной величины.

На рис. 3.10, 3.11 приведена аппроксимация длительности миграции и простоя ВМ во время миграции рядами Лагерра.

Аналогичные графики для видов миграции с предварительным копированием, дельта-сжатием, сжатием данных и пост-копированием приведены в приложении А.

3.5.7 Алгоритм метода расчета длительности миграции виртуальных машин в облачных центрах обработки данных

Для построения ряда Грама-Шарлье необходимо выполнить ряд шагов [131]:

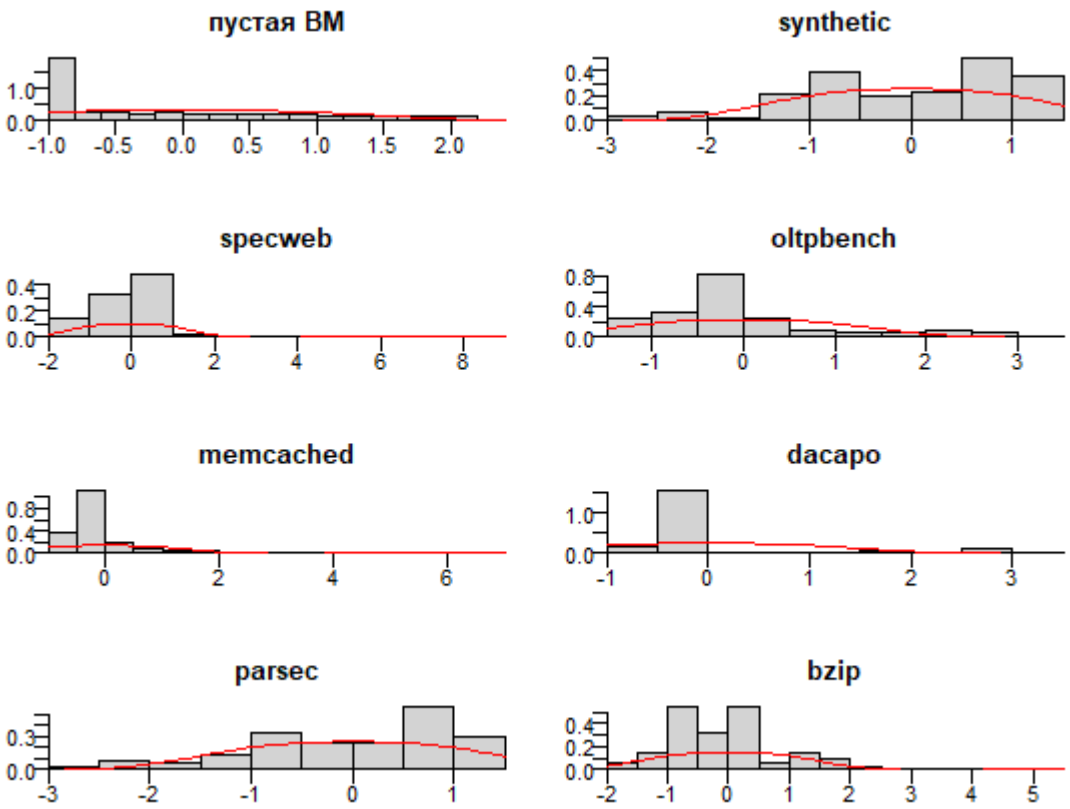

Рисунок 3.9 — Аппроксимация длительности простоя ВМ рядом Грама-Шарлье для миграции с дросселированием процессора

1. Найти оценку математического ожидания и дисперсии для случайной величины.
2. Нормировать случайную величину.
3. Рассчитать коэффициент ассиметрии и эксцесс.
4. Построить ряд по формуле 3.10. Производные от функции Лапласа затабулированы.

Алгоритм построения ряда Лагерра следующий:

1. Найти оценку математического ожидания и дисперсии для случайной величины.
2. Рассчитать α и β по формулам 3.14, 3.15.
3. Рассчитать коэффициент ассиметрии и эксцесс.
4. Построить ряд по формуле 3.13.

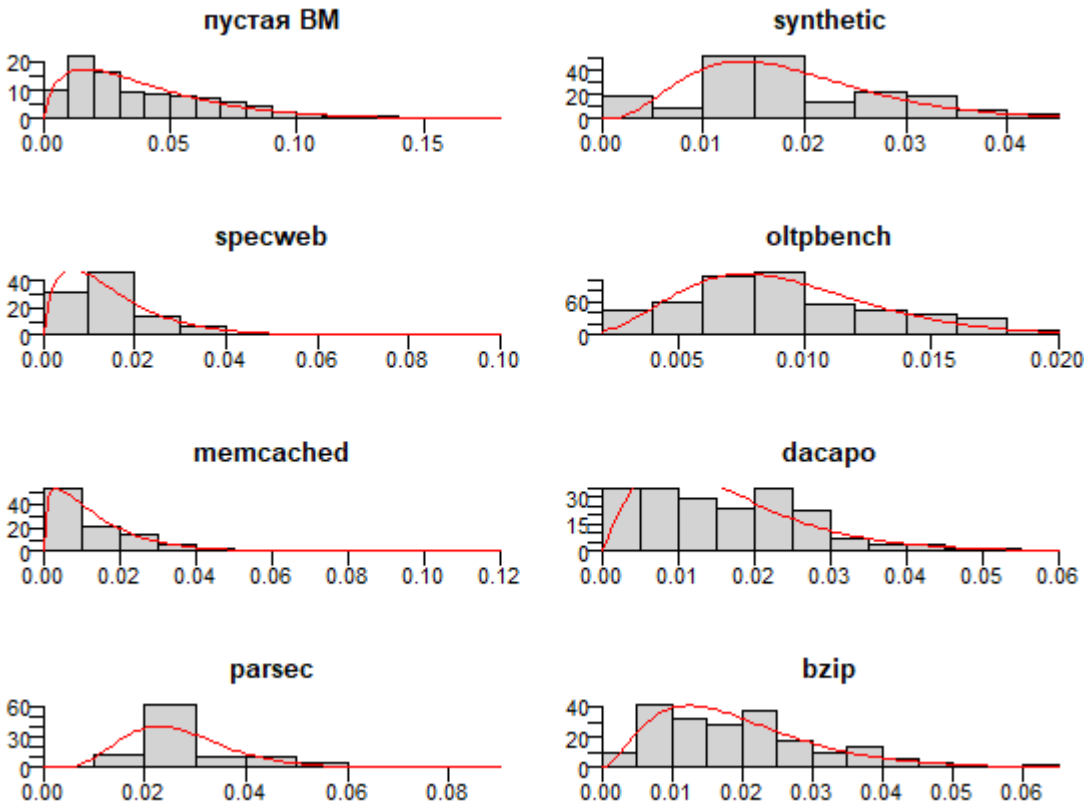

Рисунок 3.10 — Аппроксимация длительности миграции рядом Лагерра для миграции с дросселированием процессора

3.5.8 Сравнение предложенного метода с существующими работами

Отличия предлагаемого метода от существующих работ приведены в таблице 9 [132]. Основным отличием является то, что данный метод позволяет найти аналитическое выражение плотности вероятности длительности миграции и простоя ВМ. Благодаря этому можно определить квантиль по заданной вероятности или вероятность того, что длительность миграции не превысит заданную величину.

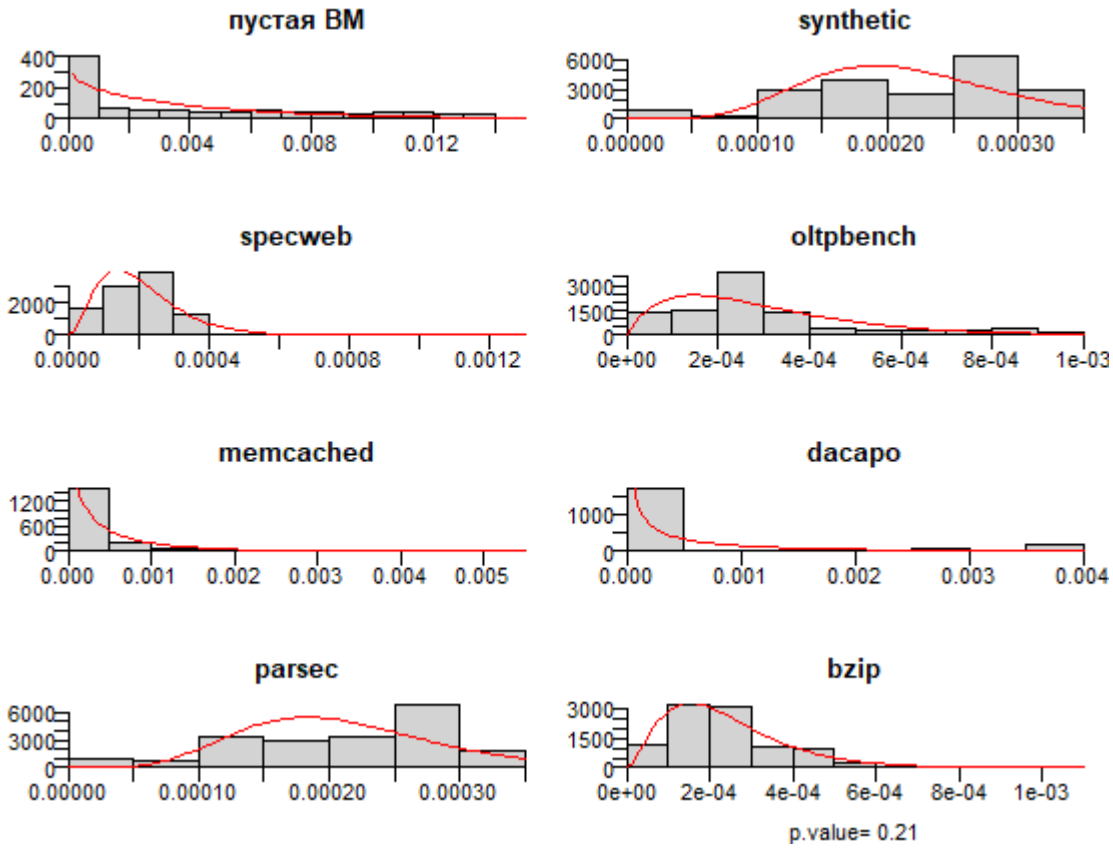

Рисунок 3.11 — Аппроксимация длительности простоя ВМ рядом Лагерра для миграции с дросселированием процессора

Таблица 9 — Сравнение предлагаемого подхода к определению длительности миграции с существующими работами

Характеристики	**Jo и др. [127]**	**Akoush и др. [128]**	**Алексанков [123]**	**Предлагаемый метод**
Подходит для всех типов миграций	+	-	-	+
Оценка длительности миграции с заданной вероятностью	-	-	-	+
Прогноз длительности миграции в зависимости от текущих параметров системы	+	+	+	-

3.6 Выводы

1. Использование метода группового учета аргументов позволяет повысить точность прогноза перегрузки сервера, что уменьшает число ненужных миграций и повышает устойчивость облачного ЦОД в целом.

2. Отношение средней длительности миграции к ширине скользящего окна может служить показателем устойчивости процесса живой миграции. Экспериментально установлено, что превышение значения 1/3 для данного коэффициента приводит к резкому росту числа обратных миграций и ухудшению качества прогноза, что позволяет рекомендовать это значение в качестве порогового при настройке системы мониторинга.
3. Исследовано поведение динамической миграции на основе набора данных динамической миграции виртуальной машины, который включает записи миграции с пятью различными алгоритмами динамической миграции: миграция после копирования (POST), миграция перед копированием (PRE) и ее модификации: дросселирование ЦП (THR), Дельта-сжатие (DLTC) и сжатие данных (DTC) с 9 типами рабочих нагрузок. Анализ полученных результатов позволяет сделать вывод, что вид приложения существенно влияет как на форму эмпирического распределения (гистограммы) продолжительности миграции, так и на ее характеристики, поэтому целесообразно получить аналитическое выражение для закона распределения суммарного времени миграции с учетом этих обстоятельств.
4. Предлагаемый аналитический подход основан на получении аналитического выражения распределения плотности вероятности суммарного времени миграции целесообразно включить в основу метода расчета ее характеристик с учетом приложений, вносящих наибольшую случайность в общее время миграции. К ним относятся такие приложения, как Dacapo, Bzip, mplayer, PARSEC и др. Для упрощения расчета вероятности общего времени миграции можно рекомендовать использование рядов Лагерра как дающих более надежные результаты по сравнению с рядами Грама-Шарлье.
5. Полученная плотность вероятности длительности миграции может использоваться для расчета коэффициента устойчивости, значение которого можно регулировать, изменяя размер скользящего окна.

Глава 4. Оптимизация размещения виртуальных машин по физическим серверам

4.1 Постановка задачи

4.1.1 Выбор критериев оптимизации

Выбор серверов для миграции на них виртуальных машин является ответственным этапом, поскольку неудачный выбор сервера может привести к новым нежелательным миграциям, так как сами миграции дополнительно нагружают систему и приводят к ухудшению производительности и простою виртуальных машин.

В большинстве работ, в которых рассматривается статическое размещение виртуальных машин, используются постановки задачи размещения виртуальных машин в форме задачи об упаковке в контейнеры или о рюкзаке [88; 89; 133—135]. Эти задачи относятся к классу NP-сложных задач. Поэтому на практике для выбора целевого сервера для размещения мигрирующих виртуальных машин широко используются жадные эвристические алгоритмы, такие как First Fit Decreasing (FFD), Best Fit Decreasing и их модификации [67; 136—138]. Однако, поскольку размещение виртуальных машин – это NP-сложная задача, эвристические алгоритмы не гарантируют получения оптимального и близкого к оптимальному решению.

В последнее время метаэвристические муравьиные алгоритмы (ACO) и генетические алгоритмы использовались для решения задачи размещения виртуальных машин [86; 135; 139—142]. Однако такие алгоритмы также не гарантируют получения оптимальных решений. Кроме этого, в некоторых работах [139; 140] учитывается только одномерный ресурс.

Время, необходимое для решения задачи оптимизации, является одним из основных факторов, влияющих на качество принятия решений в реальном времени. Один цикл работы контроллера длится несколько минут. В работе [79] для моделирования использовался 5-минутный цикл. В работе [143] использовался 2-минутный цикл для отслеживания загрузки процессора, а 6-минутный

цикл использовался для обнаружения низкой энергоэффективности. В работе [144] рассматривалась проблема выбора оптимального размера окна для обеспечения устойчивости процесса миграции. Он основан на оценке длительности миграции, приведенной в работах [145—147]. В течение этого времени контроллеры должны обнаружить проблему на серверах (перегрузка, недогрузка или перегрев), выбрать виртуальные машины для миграции и серверы для размещения виртуальных машин и перенести их. Задержки в принятии решений могут привести к значительным штрафам за нарушение соглашений SLA и дополнительным операционным расходам. Нерегулируемое увеличение задержек сделает невозможным внедрение инновационных высокопроизводительных услуг облачных центров обработки данных.

Некоторые подходы к динамической консолидации виртуальных машин [65] пытаются «упаковать» виртуальные машины в минимальное количество серверов, уменьшая при этом число миграций. Необходимость инициирования миграций обсуждалась в параграфе 3.4. Вопрос заключается в том, какие физические серверы для размещения виртуальных машин необходимо выбрать.

Задачи многокритериального размещения рассматривались в [86; 135; 138; 140—143; 148]. Для решения таких многокритериальных задач чаще всего используются методы формирования обобщенного критерия [79; 86; 135; 143].

Однако большинство вышеперечисленных работ не гарантируют точного решения и не полностью учитывают вопросы размерности задачи и времени решения. Поэтому желательно сформулировать задачу, которая позволила бы масштабировать ресурсы в широком диапазоне и дать возможность получить точное решение в реальном времени. Процесс размещения виртуальных машин на физических серверах описывается рядом показателей: эффективность использования ресурсов [86; 88; 135], энергопотребление [38; 65; 86; 88; 135—137; 149; 150], равномерное распределение температуры [88; 143], выполнение SLA-соглашений [86; 88], балансировка нагрузки [67], минимизация трафика [150; 151] и другие. Белоглазов и Буйя [79] предложили комбинированную метрику, отражающая как уровень нарушений SLA, так и потребление энергии, которая обсуждается в параграфе 3.4.1 и используется в данной работе в имитационном моделировании.

Поскольку предполагается, что ВМ уже выбраны для миграции, то такой критерий как энергопотребление для данной задачи не подходит.

Наиболее подходящими критериями для данной задачи являются критерий неэффективности использования ресурсов и уровень нарушений SLA-соглашений.

Пусть имеется N активных физических серверов и такое же количество виртуальных машин для размещения путем миграции. Данные серверы являются работающими и могут обслуживать другие виртуальные машины.

Для каждой виртуальной машины заданы производительность процессора VM_i^{CPU} и объем оперативной памяти VM_i^{RAM}, $i = 1, \ldots, N$. Каждый физический сервер имеет свои собственные характеристики: производительность процессора $PM_j^{CPU_0}$ и объем памяти $PM_j^{RAM_0}$. Предположим, что в рассматриваемый момент времени у каждого сервера задействована часть ресурсов другими виртуальными машинами, которую обозначим для процессора $PM_j^{CPU_1}$ и $PM_j^{RAM_1}$ для памяти. Предположим также, что оставшейся свободной части ресурсов сервера достаточно для размещения любой ВМ из N имеющихся.

Глобальный контроллер должен определить физические серверы, на которые будут перемещены виртуальные машины. Учтем цикличность процесса динамического размещения виртуальных машин, что дает возможность предположить: за один цикл работы контроллера на каждом отдельном сервере может быть размещена только одна виртуальная машина точно так же, как каждая виртуальная машина может быть размещена только на одном сервере. Постановку задачи можно проиллюстрировать в виде полного двудольного графа (рис. 4.1).

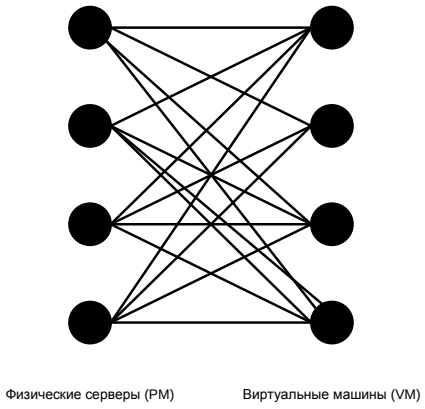

Рисунок 4.1 — Иллюстрация задачи размещения виртуальных машин по серверам в виде полного двудольного графа

Задача состоит в том, чтобы найти оптимальное размещение виртуальных машин с учетом минимального времени принятия решения для процесса размещения, которое не превышает рабочего цикла контроллера в 2-6 минут.

Предположим, что размещаемая на сервере виртуальная машина занимает всю выделенную ей память и процессорное время. Тогда обозначим u_{ij}^{CPU} загрузку процессора сервера j после размещения виртуальной машины i, а u_{ij}^{RAM} загрузку памяти сервера j после размещения виртуальной машины i.

$$u_{ij}^{CPU} = \frac{PM_j^{CPU_1} + VM_i^{CPU}}{PM_j^{CPU_0}} \tag{4.1}$$

$$u_{ij}^{RAM} = \frac{PM_j^{RAM_1} + VM_i^{RAM}}{PM_j^{RAM_0}} \tag{4.2}$$

Критерий неэффективности использования ресурсов отражает, насколько несбалансированно используются ресурсы на каждом сервере. Остаток ресурсов на каждом сервере должен быть сбалансирован по нескольким видам ресурсов. Например, неудачным размещением виртуальных машин является такое размещение, при котором вся память сервера использована, а процессор малозагружен.

Критерий неэффективности использования ресурсов для сервера можно сформулировать следующим образом:

$$f_{res}^{j}(u_{ij}^{CPU}, u_{ij}^{RAM}) = 1 - u_{ij}^{CPU} \cdot u_{ij}^{RAM} \tag{4.3}$$

Данный критерий отражает насколько полно загружены ресурсы серверов различных типов. Значения данного критерия лежат в диапазоне от 0 до 1. Чем ближе значение критерия к нулю, тем лучше загружены ресурсы сервера.

Еще одним рассматриваемым критерием является нарушение SLA-соглашений. Требование к качеству обслуживания для облачного сервиса задается как среднее время отклика, которое в основном зависит от загрузки процессора. При превышении некоторого порога загрузки наблюдается снижение производительности приложения и увеличение времени отклика. В работе [137] было исследовано влияние порога загрузки процессора на нарушения SLA. В большинстве работ используется порог загрузки в диапазоне 80-90% [67; 143].

В качестве критерия нарушения SLA-соглашений предлагается использовать следующую логистическую функцию:

$$f_{SLA}^{j}(u_{ij}^{CPU}) = \frac{1}{1 + e^{-(u_{ij}^{CPU} - 80)}} \tag{4.4}$$

Значения этой функции также лежат в диапазоне от 0 до 1, в точке порогового значения $u_{ij}^{CPU} = 80\%$ значение функции равно 0,5 и быстро увеличивается при превышении порогового значения. Данный критерий должен быть минимизирован.

4.1.2 Математическая постановка задачи размещения виртуальных машин

Введем переменные задачи x_{ij}, которые будут соответствовать назначению ВМ на сервер. В этом случае $x_{ij} = 1$, если i-ая ВМ назначается на выполнение j-му серверу, и $x_{ij} = 0$, если i-ая ВМ не назначается на выполнение j-му серверу.

Математическая постановка задачи размещения ВМ выглядит следующим образом [152]:

$$\sum_{i=1}^{N}\sum_{j=1}^{N} f_{res}^{j}(u_{ij}^{CPU},u_{ij}^{RAM}) \cdot x_{ij} \to \min_{x\in\Delta_\beta}, \tag{4.5}$$

$$\sum_{i=1}^{N}\sum_{j=1}^{N} f_{SLA}^{j}(u_{ij}^{CPU},u_{ij}^{RAM}) \cdot x_{ij} \to \min_{x\in\Delta_\beta}, \tag{4.6}$$

где множество допустимых альтернатив Δ_β формируется следующей системой ограничений:

$$\begin{cases} \sum_{j=1}^{N} x_{ij} = 1,\ \forall i \in \{1,2,\ldots,N\}, \\ \sum_{i=1}^{N} x_{ij} = 1,\ \forall j \in \{1,2,\ldots,N\}, \\ x_{ij} \in \{0,1\},\ \forall i,j \in \{1,2,\ldots,N\}. \end{cases} \tag{4.7}$$

Поставленная задача по структуре, необходимым условиям, характеру переменных эквивалентна известной основной задаче о назначениях.

Классическая постановка задачи о назначениях формулируется следующим образом. Существует конечное число видов работ (должностей), которые могут быть выполнены потенциальными кандидатами. Однако каждый кандидат может быть назначен только на одну работу, и каждая работа, в свою очередь, должна выполняться только одним кандидатом. Эффективность каждой работы, выполняемой любым из потенциальных кандидатов, известна.

Необходимо распределить всех кандидатов по должностям так, чтобы общая производительность всех работ была самой высокой. Целевой функцией в данной задаче является суммарная эффективность всех работ, а ограничениями – дополнительные условия для выполнения каждой работы только одним кандидатом и для участия каждого кандидата только в одной работе.

4.2 Методы решения

4.2.1 Многокритериальный метод

Поставленная в работе задача относится к классу многокритериальных, для которых разработано несколько подходов и методов решения. Наибольшее распространение получил метод линейной свертки. Он заключается в назначении тем или иным способом коэффициентов в линейной свертке исходных критериев и последующем нахождении ее экстремума на множестве допустимых альтернатив. Согласно этому методу, найденное таким способом решение считается «наилучшим» и является Парето-оптимальным [153]. Кроме этого, данный метод наиболее подходит для решения задачи с булевыми переменными. Таким образом, фактически осуществляется переход от многокритериальной задачи к однокритериальной.

$$F = CX = \alpha_1 \sum_{i=1}^{N} \sum_{j=1}^{N} f_{res}^{j}(u_{ij}^{CPU}, u_{ij}^{RAM}) \cdot x_{ij} + \alpha_2 \sum_{i=1}^{N} \sum_{j=1}^{N} f_{SLA}^{j}(u_{ij}^{CPU}, u_{ij}^{RAM}) \cdot x_{ij}, \tag{4.8}$$

где α_1, α_2–веса частных критериев; C–матрица затрат на размещение.

Венгерский метод

Специальный Венгерский метод был разработан Куном для решения задачи о назначении [154].

Оригинальность этого метода основана на следующем свойстве матрицы затрат. Если ко всем элементам c_{ij} некоторой i-й строки прибавить произвольное постоянное число u_i, а ко всем элементам j-го столбца – произвольное постоянное число v_j, то получится новая матрица затрат с элементами: $d_{ij} = c_{ij} + u_i + v_j$. В этом случае все элементы предыдущей матрицы будут равны $c_{ij} = d_{ij} - u_i - v_j$. Таким образом, идея нахождения оптимального решения задачи о назначениях венгерским методом заключается в переходе к эквивалентной задаче путем модификации целевой функции и системы ограничений: наименьшие элементы последовательно вычитаются из элементов каждой строки и каждого столбца исходной матрицы затрат. После чего анализируется получение допустимого решения. Если получено допустимое решение, которому соответствуют нули в модифицированной матрице затрат, то оно является оптимальным назначением. Если же решение недопустимое, то выполняется дальнейшая модификация матрицы затрат. Такой подход позволил получить решение за полиномиальное время: оригинальный алгоритм Куна имеет вычислительную сложность $O(n^4)$ [154] и усовершенствованный алгоритм, предложенный в [155] $O(n^3)$.

Преимущество метода заключается в том, что нет необходимости хранить в памяти матрицу, соответствующую системе ограничений, что значительно экономит память. Другие модификации венгерского алгоритма направлены на снижение асимптотической сложности, а также на учет специфики предметной области.

Сведение задачи о назначениях к закрытой транспортной задаче

Известно [155; 156], что задача о назначении может быть сведена к закрытой транспортной задаче путем замены ограничения на $x_{ij} \geqslant 0$.

В итоге постановка задачи будет выглядеть следующим образом:

$$\alpha_1 \cdot \sum_{i=1}^{N} \sum_{j=1}^{N} f_{res}^{j}(u_{ij}^{CPU}, u_{ij}^{RAM}) \cdot x_{ij} + \alpha_2 \cdot \sum_{i=1}^{N} \sum_{j=1}^{N} f_{SLA}^{j}(u_{ij}^{CPU}, u_{ij}^{RAM}) \cdot x_{ij} \to \min_{x \in \Delta_\beta}, \quad (4.9)$$

при ограничениях

$$\begin{cases} \sum_{j=1}^{N} x_{ij} = 1,\ \forall i \in \{1,2,\ldots,N\}, \\ \sum_{i=1}^{N} x_{ij} = 1,\ \forall i \in \{1,2,\ldots,N\}, \\ x_{ij} \geqslant 0,\ \forall i,j \in \{1,2,\ldots,N\}, \end{cases} \tag{4.10}$$

где α_1, α_2 – веса критериев; $\alpha_1 + \alpha_2 = 1$, $\alpha_1,\alpha_2 \geqslant 0$.

Это даёт возможность использовать для получения оптимального решения известное и доступное программное обеспечение. В настоящее время разработано большое количество методов решения транспортных задач: оригинальный симплекс-метод, разработанный Данцигом [157], метод потенциалов Канторовича [158], метод разрешающих слагаемых [159] и др.

4.2.2 Ограничения на ресурсы

Предположение о том, что у каждого сервера достаточно ресурсов для размещения любой виртуальной машины, не всегда применимо. В этом случае граф назначений является неполным двудольным графом (рис. 4.2).

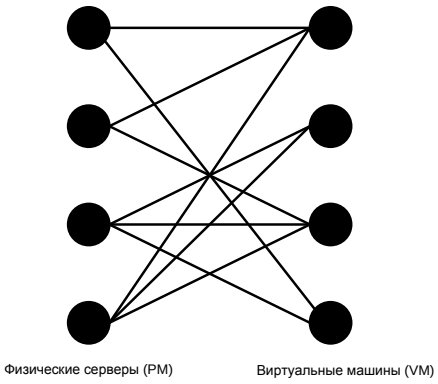

Рисунок 4.2 — Иллюстрация задачи размещения виртуальных машин по серверам в виде неполного двудольного графа

В этом случае элементу матрицы затрат c_{ij}, соответствующему VM_i и PM_j, должно быть присвоено неэффективное большое значение M, чтобы это назначение не было включено в решение задачи. Желательно выбрать число M равное N. Значение целевой функции при правильном размещении виртуальной машины всегда меньше N. Если значение целевой функции становится

больше N, это означает, что ресурсов хоста недостаточно для обслуживания виртуальных машин и необходимо включить дополнительные физические серверы. Более того, можно быстро определить, скольким виртуальным машинам не хватает места на физических серверах: для этого надо нацело разделить полученное значение целевой функции F на N. Вышесказанное можно проиллюстрировать в виде следующего алгоритма 4.

4.2.3 Несимметричная задача

Задача решается, если количество серверов равно количеству виртуальных машин. В этом случае она может быть решена венгерским методом или сведена к закрытой транспортной задаче. Однако, если количество серверов больше, чем количество виртуальных машин, то может быть использован метод сведения открытой транспортной проблемы к закрытой: введение фиктивных виртуальных машин $x_{m+1,j}$ $\forall j \in 1, \cdots, N$ и присвоение больших значений затрат $c_{m+1,j}$, соответствующих дополнительной фиктивной $(m+1)$ виртуальной машине.

Случай, когда число виртуальных машин для миграции превышает количество доступных физических серверов, встречается довольно редко. Аналогично, проблема может быть решена путем введения фиктивных узлов физических серверов. Оставшиеся виртуальные машины должны быть перемещены во время следующего цикла контроллера.

4.2.4 Вычислительные эксперименты

Описание экспериментов

Для оценки предложенного многокритериального подхода к размещению виртуальных машин были проведены три серии экспериментов. Первая серия экспериментов направлена на сравнение многокритериального алгоритма с эв-

Таблица 10 — Исходные данные для экспериментов

Серия экспериментов	**Размер ЦОД**
1. Эффективность размещения	50 ВМ, 50 серверов
2. Масштабируемость	50∼250 ВМ и серверов
3. Имитационное моделирование крупного ЦОД	150 ВМ, 100 серверов

ристическими алгоритмами задачи об упаковке в контейнеры, используемые на практике. Вторая серия была направлена на определение времени решения задачи в зависимости от размерности. Для этих двух экспериментов моделирование выполнялось с использованием языка R. Параметры ресурсов виртуальных машин были сгенерированы случайным образом. Производительность процессора виртуальной машины в ГГц равномерно распределяется по следующему набору значений 0.25, 0.5, 1, 1.5, 2, 2.5, 3, 4, аналогично память в ГБ случайно задавалась из того же набора значений. Количество доступных физических и виртуальных машин устанавливалось в качестве начальных значений для моделирования центров обработки данных различных размеров. Для каждой серии экспериментов генерируются наборы случайных входных данных. Каждый эксперимент проводился 20 раз. Результаты усреднялись.

В третьем наборе экспериментов оцениваются предлагаемые алгоритмы распределения ресурсов в крупномасштабной виртуализированной инфраструктуре центра обработки данных с использованием платформы имитационного моделирования CloudSim [160].

В таблице 10 приведены настройки параметров для двух наборов экспериментов. Для каждой серии экспериментов сгенерированы наборы случайных входных данных. Каждый эксперимент был проведен 20 раз. Полученные результаты усреднены.

Эффективность размещения

Первая серия экспериментов направлена на сравнение многокритериального алгоритма с эвристическими алгоритмами для задачи упаковки в контейнеры, используемыми на практике, такими как:

– алгоритм Best Fit Decreasing (BFD)—список серверов сортируется по убыванию в соответствии с производительностью процессора (bfd_cpu) или памяти (bfd_mem), далее каждая виртуальная машина закрепляется за сервером таким образом, чтобы остаток используемого ресурса (процессора или памяти) был минимальным.
– алгоритм First Fit Decreasing (FFD)—список серверов сортируется по убыванию в соответствии с производительностью процессора (ffd_cpu) или памяти (ffd_mem), далее каждая виртуальная машина закрепляется за первым походящим сервером в списке.

При решении задачи оптимизации методом свертки значения вектора весовых коэффициентов изменялись в диапазоне (0,1; 0,9) (0,9; 0,1) с шагом 0,2.

Также вычислялись минимальные и максимальные значения по каждому критерию, относительно которых вычислялись нормализованные значения критериев. Результаты по критериям нарушения SLA-соглашений и эффективности использования ресурсов для алгоритмов bfd_cpu, bfd_mem, ffd_cpu, ffd_mem и α_1, α_2 = 0,1; 0,3; 0,5; 0,7; 0,9 приведены на рис. 4.3 и рис. 4.4.

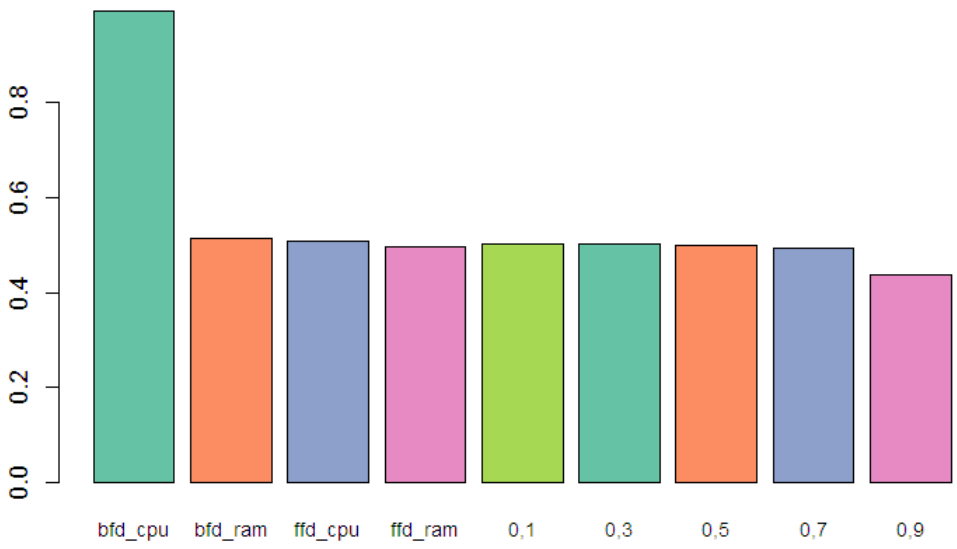

Рисунок 4.3 — Нормализованные значения критерия нарушения SLA-соглашений для различных алгоритмов

Как видно в соответствии с алгоритмом bfd_cpu виртуальные машины размещались плотнее на серверы и, как следствие, имеется самый высокий уровень нарушения SLA-соглашений. Также алгоритмы bfd_cpu, bfd_mem,

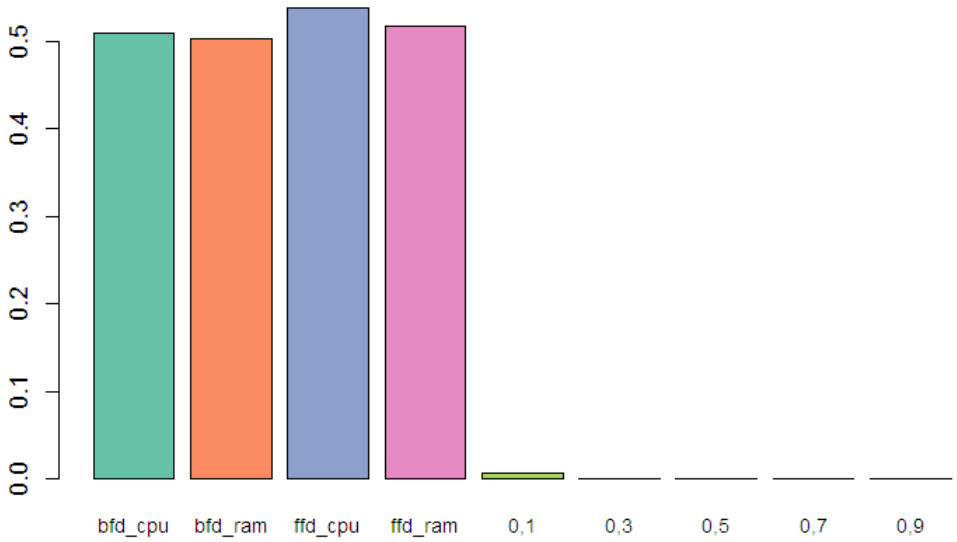

Рисунок 4.4 — Нормализованные значения критерия неэффективности использования ресурсов

ffd_cpu, ffd_mem, являясь однокритериальными, демонстрируют неэффективную загрузку нескольких видов ресурсов (рис. 4.4). В этом отношении свертка является предпочтительным методом, дающим эффективное решение по двум критериям.

Масштабируемость

Вторая серия экспериментов была направлена на определение скорости решения в зависимости от размерности задачи. Транспортная задача решалась симплекс-методом, реализованным в пакете lpSolve для языка R на компьютере Pentium(R) Dual-Core CPU E5700 3 ГГц 4 Гб ОЗУ. Время решения в зависимости от размерности задачи приведено в таблице 11. Зависимость времени вычисления задачи от размерности является квадратичной $O(n^2)$ с величиной достоверности аппроксимации $R^2 = 0{,}9916$ и показана на рис. 4.5. Для венгерского алгоритма вычислительная сложность составляет $O(n^3)$ при $R^2 = 0{,}9985$, но в целом венгерский алгоритм быстрее решает задачу и экономичнее использует память. Для задач большой размерности (500-1500 виртуальных машин)

Таблица 11 — Зависимость времени решения от размерности задачи

Размер ЦОД	**Время решения симплекс-методом (с.)**	**Время решения венгерским алгоритмом (с.)**
50 ВМ, 50 серверов	0,73	0,37
100 ВМ, 100 серверов	1,72	0,41
150 ВМ, 150 серверов	4,84	0,57
200 ВМ, 200 серверов	11,44	0,88
250 ВМ, 250 серверов	22,17	1,16
500 ВМ, 500 серверов	Не хватает памяти	3.46
1000 ВМ, серверов	Не хватает памяти	11,89
1500 ВМ, серверов	Не хватает памяти	33,87

симплекс метод, реализованный в пакете *lpSolve*, не дал результата, поскольку не хватило памяти для размещения матрицы системы ограничений.

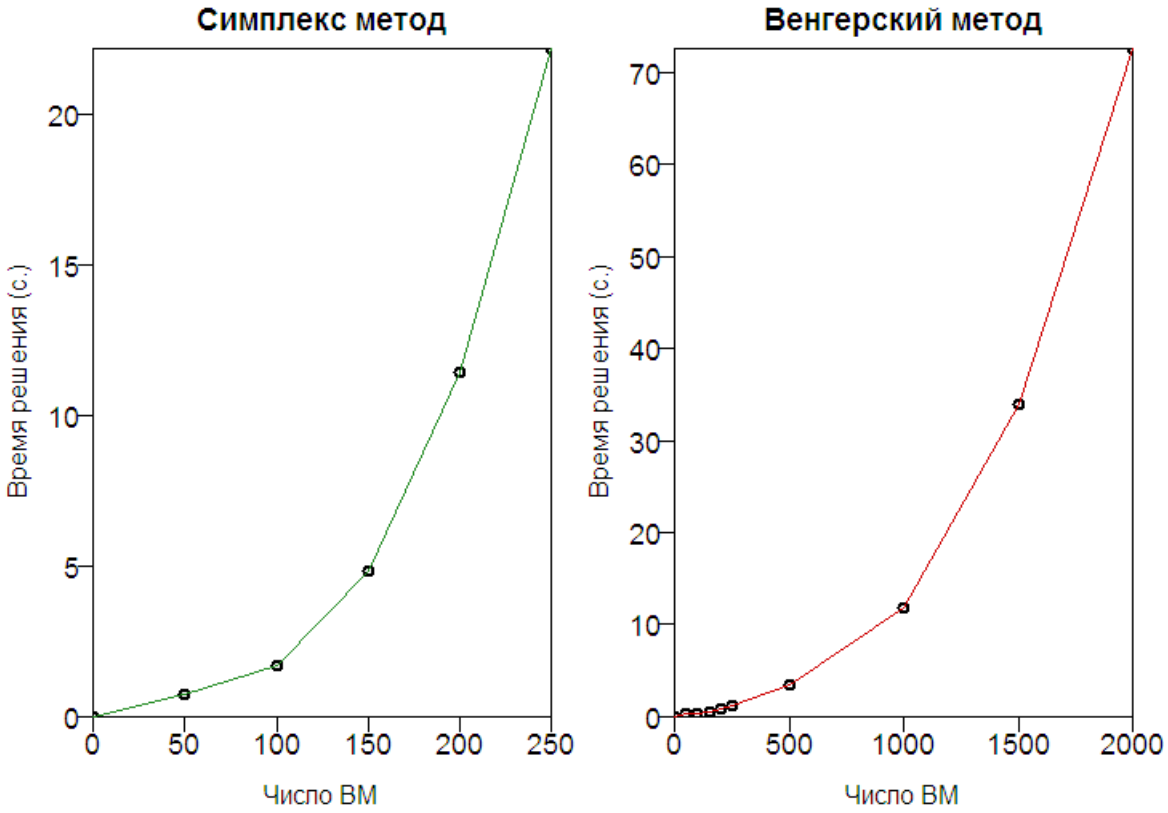

Рисунок 4.5 — Зависимость времени решения задачи от размерности

Имитационное моделирование крупного ЦОД

Предлагаемый метод размещения виртуальных машин встроен в модель энергоэффективного центра обработки данных на платформе CloudSim, разработанную в [79].

В начале моделирования каждому серверу случайным образом назначаются одна-две виртуальные машины. В процессе симуляции отслеживается загрузка серверов, и на первом этапе определяются перегруженные хосты. Если хост перегружен (что приводит к нарушению SLA), на нем выбирается одна виртуальная машина для миграции. Выбор падает на машину с минимальным объемом оперативной памяти, так как это обеспечивает наименьшее время миграции (MMT — Minimum Migration Time).

Для оценки эффективности алгоритмов размещения виртуальных машин в имитационных моделях не применялись методы прогнозирования перегрузки. Вместо этого использовался статический порог загрузки процессора, равный 0,8. При превышении данного порога хост считался перегруженным, и требовалось выполнить миграцию ВМ для его разгрузки. Для соблюдения преемственности в обозначениях, в частности с работой [79], принято кодовое обозначение данной имитационной модели — THR-MMT-0,8.

На втором этапе выполняется поиск неиспользуемых хостов. Если хост недогружен, его виртуальные машины переносятся на другие хосты, а сам хост переводится в спящий режим. Таким образом, алгоритм пытается объединить виртуальные машины на меньшем количестве физических серверов.

На этапе поиска подходящих хостов для миграции виртуальных машин был реализован предлагаемый алгоритм размещения ВМ. Для этого в классе PowerVMAllocationPolicyMigrationAbstract пакета org.cloudbus.cloudsim.examles.power.planetlab были изменены методы getNewVMPlacement и getNewVMPlacementFromUnderUtilizedHost.

Эффективность алгоритмов оценивалась по таким показателям, как общее энергопотребление центра обработки данных (E) и уровень нарушения соглашений об уровне обслуживания ($SLAV$).

В ходе экспериментов моделировалась работа ЦОД, состоящего из 100 хостов и 150 виртуальных машин. Имитационное моделирование выполнялось для всех 10 рабочих нагрузок, входящих в состав CloudSim и перечисленных

в таблице 14. В процессе экспериментов варьировались весовые коэффициенты критериев от 0,1 до 0,9 с шагом 0,1. Полученные результаты сравнивались с показателями алгоритма FFD.

Всего было проведено 100 экспериментов. Результаты одного из них (для первых 150 экземпляров рабочих нагрузок, датированных 03/06/2011) представлены в таблице 12. На рис. 4.6 показаны значения метрики ESV для данной модели. Остальные результаты вынесены в приложение Б.

Таблица 12 — Результаты моделирования для первых 150 экземпляров рабочих нагрузок 20110306

Алгоритм (α_1, $\alpha 2$)	**ESV** $\cdot 10^{-3}$	**E** (КВт/ч)	**SLAV** $\cdot 10^{-5}$	**SLATAH** (%)	**PDM** (%)	**Число миграций**
THR-MMT-0.8 (0.9; 0.1)	56,76	17,21	3,30	16,49	0,02	4708
THR-MMT-0.8 (0.8; 0.2)	70,20	17,54	4,00	20,01	0,02	4599
THR-MMT-0.8 (0.7; 0.3)	32,71	17,29	1,89	18,92	0,01	4559
THR-MMT-0.8 (0.6; 0.4)	79,31	18,41	4,31	14,36	0,03	4829
THR-MMT-0.8 (0.5; 0.5)	31,15	18,25	1,71	5,69	0,03	4484
THR-MMT-0.8 (0.4; 0.6)	26,75	18,93	1,41	4,71	0,03	4668
THR-MMT-0.8 (0.3 0.7)	34,42	19,08	1,80	4,51	0,04	4578
THR-MMT-0.8 (0.2; 0.8)	37,89	18,83	2,01	5,03	0,04	4692
THR-MMT-0.8 (0.1; 0.9)	25,04	19,1	1,31	4,37	0,03	4728
THR-MMT-0.8 FFD	137,21	27,72	4,95	4,95	0,1	4627

Результаты таблицы 25 и таблиц, приведенных в приложении Б показывают, что весовые коэффициенты частных критериев оптимизации в целом влияют на результат. Чем больше весовой коэффициент энергопотребления, тем меньше показатель *E*. Аналогично, чем больше значение весового коэффициента для нарушения SLA, тем меньше показатель *SLAV*. Тем не менее по комбинированному критерию *ESV* нельзя однозначно сказать какую комбинацию весовых коэффициентов лучше использовать.

Во всех экспериментах предлагаемый алгоритм размещения виртуальных машин превосходит алгоритм FFD как по частным критериям, так и по комплексному показателю *ESV* (таблица 13). На рисунке 4.7 приведены ящичковые диаграммы, отражающие соотношение метрики *ESV* для предлагаемого алгоритма и алгоритма FFD в имитационных моделях с различными типами нагрузки.

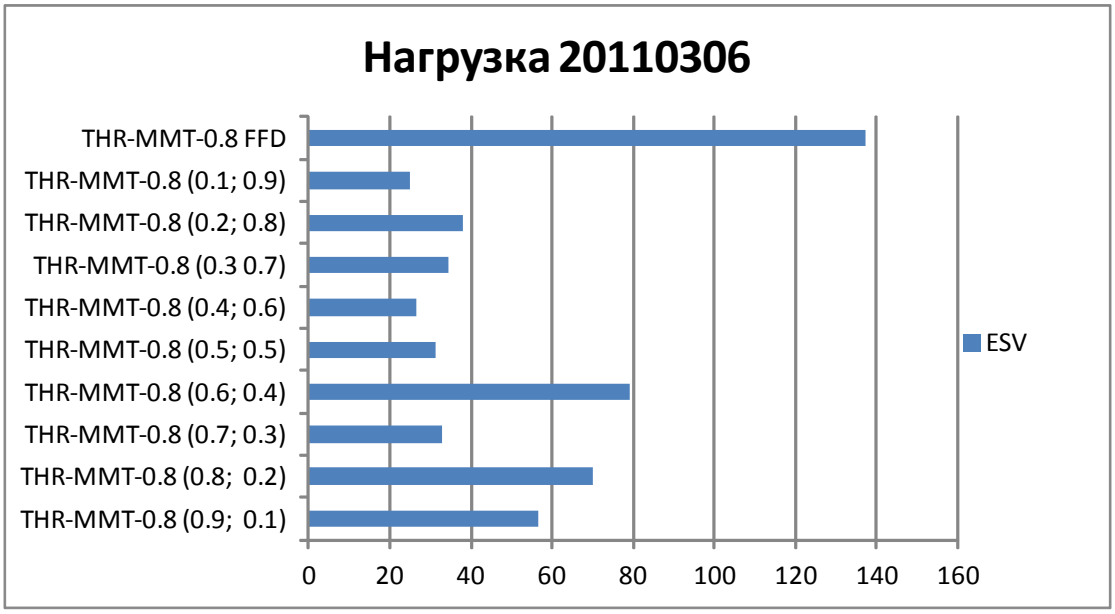

Рисунок 4.6 — Метрика ESV предлагаемого алгоритма с различными весами критериев и алгоритм FFD для имитационной модели рабочей нагрузки 06/03/2011

Таблица 13 — Оценки числовых характеристик распределения отношения метрики *ESV* для алгоритма FFD к предлагаемому алгоритму для различных нагрузок

Нагрузка	**Мин**	**Квартиль 1**	**Медиана**	**Среднее**	**Квартиль 3**	**Макс**
03/03/2011	1,800	3,522	3,952	4,383	5,131	8,011
06 03/2011	1.730	2,418	3,986	3,658	4,405	5,480
09/03/2011	1,723	1,951	2,611	2,447	2,717	3,283
22/03/2011	2,237	3,204	3,426	3,588	4,337	4,484
25/03/2011	2,158	2,740	3,448	3,214	3,746	4,032
03/04/2011	1,273	1,626	3,313	2,866	3,889	4,425
09/04/2011	1,845	2,614	4,179	3,697	4,726	5,162
11/04/2011	2,093	3,973	4,693	4,730	5,011	8,330
12/04/2011	1,426	2,352	3,655	3,589	4,677	6,003
20/04/2011	2,607	2,800	3,886	3,658	4,217	4,913

В целом, результаты моделирования показывают, что предложенный метод размещения виртуальных машин позволяет в среднем в 3,5 раза улучшить значение комбинированной метрики *ESV* (учитывающей энергопотребление и нарушения SLA), по сравнению с широко используемым на практике эвристическим алгоритмом FFD.

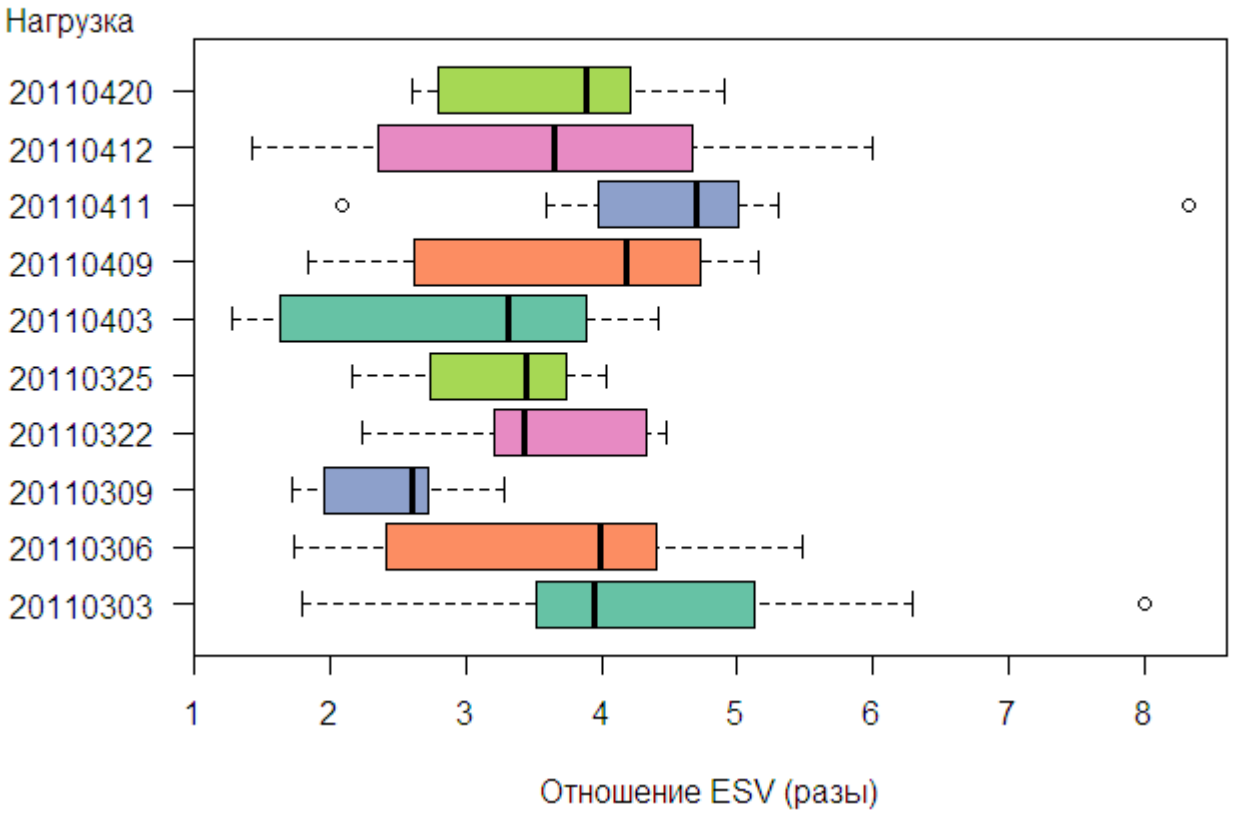

Рисунок 4.7 — Ящичковые диаграммы распределения отношения метрики ESV для алгоритма FFD к предлагаемому алгоритму для различных нагрузок

4.2.5 Другие области применения задачи размещения виртуальных машин

Рассмотренная в работе постановка задачи размещения виртуальных машин также может быть применена для:

- горизонтального масштабирования ресурсов в системах SaaS и PaaS при добавлении дополнительных виртуальных машин в кластеры;
- организации отказоустойчивых кластеров, в которых каждый экземпляр виртуальной машины должен располагаться на отдельном физическом сервере. В этом случае в настройках платформ виртуализации задается правило распределения, запрещающее размещение ВМ на одном хосте, и предложенная постановка задачи позволяет оптимизировать данное распределение.

Предложенный алгоритм размещения ВМ имеет некоторые ограничения. Во-первых, его применение становится неэффективным при сильной несбалансированности задачи, а именно, когда число хостов значительно превышает число виртуальных машин. В этом случае существенно возрастают вычислительные и временные затраты на решение задачи, рассматриваемой как симметричная. Во-вторых, в текущей постановке учитываются только два критерия оптимизации. Однако облачного провайдера могут интересовать и другие метрики, такие как пропускная способность каналов связи, сетевая задержка и др. Включение дополнительных критериев является перспективным направлением дальнейших исследований.

4.3 Выводы

1. В рамках четвертой главы формализована задача оптимизации размещения виртуальных машин, учитывающая два основных и зачастую конфликтующих критерия: соблюдение SLA-соглашений и эффективность использования вычислительных ресурсов инфраструктуры.
2. Для получения точного и вычислительно эффективного решения предложено использовать венгерский метод. Небольшая вычислительная сложность алгоритма венгерского метода обеспечивает возможность работы в режиме, близком к реальному времени, что является критическим требованием для современных облачных сред в условиях быстрого роста масштабов инфраструктуры.
3. Результаты серии вычислительных экспериментов и имитационного моделирования доказали высокую практическую эффективность предложенного метода. Установлено, что по комплексной метрике ESV, объединяющей показатели энергопотребления и нарушений SLA-соглашений, разработанный алгоритм превосходит широко применяемый на практике эвристический алгоритм FFD в среднем в 3,5 раза.
4. Разработанный алгоритм прогнозирования перегрузки серверов, математическая постановка задачи оптимизации размещения и подтвержденная результатами экспериментов программа могут быть непосредственно интегрированы в математическое и программное обес-

печение планировщика ресурсов современного ЦОД для повышения эффективности управления инфраструктурой.

Глава 5. Вопросы практической реализации моделей и методов распределения ресурсов инфокоммуникационной системы облачного ЦОД

5.1 Выбор метода экспериментального исследования

Согласно классификации, предложенной в [161], методы экспериментального исследования крупномасштабных и распределенных систем делятся на следующие категории:

Натурное моделирование предполагает выполнение реальной задачи на реальном оборудовании в реальных сетевых условиях. В качестве примеров платформ, поддерживающих такой подход, можно привести Grid'5000 [162], PlanetLab [163], EdgeNet [164] и др.

Эмуляция заключается в запуске реального приложения в искусственно созданной среде, моделирующей поведение целевой вычислительной инфраструктуры. Основная задача эмуляции — обеспечить контролируемые условия эксперимента при сохранении аутентичности исполняемого кода.

Эталонное тестирование представляет собой процедуру автоматизированной оценки производительности вычислительной системы с использованием синтетических нагрузок, генерируемых специальными модельными приложениями (например, HPL Linpack, NAS). Основная цель такого тестирования заключается не в получении содержательного результата вычислений, а в измерении ключевых параметров системы: вычислительной мощности, пропускной способности сети, скорости ввода-вывода. Принципиальное отличие модельного приложения от реального состоит в том, что ценность представляют не выходные данные, а показатели функционирования системы, зафиксированные в ходе выполнения теста.

Имитационное моделирование как метод исследования вычислительных систем применяется давно и весьма широко.

Имитационные модели воспроизводят работу системы на временном интервале, предоставляя количественные данные для анализа. Для изучения вычислительных систем с дискретными событиями в случайные моменты времени используется событийное статистическое моделирование.

Главное достоинство подхода — гибкость, обусловленная модельной природой как приложения, так и вычислительной среды, что упрощает изменение параметров эксперимента. Недостаток — значительная трудоемкость создания этих моделей. К инструментам такого класса относятся, например, GridSim, SimGrid и CloudSim [165—167].

Облачный ЦОД представляет собой крупномасштабную систему, включающую в себя сотни серверов и тысячи виртуальных машин, поэтому важно оценить эффективность предлагаемых алгоритмов в крупномасштабном ЦОД. Однако сложно обеспечить повторяемость экспериментов на реальной инфраструктуре, не говоря о том, что это очень дорого. Поэтому для проведения экспериментов было выбрано имитационное моделирование.

5.2 Платформа имитационного моделирования CloudSim

Для моделирования распределенных систем используются как универсальные (GPSS), так и специализированные платформы: GridSim, SimGrid, GangSim (для грид-систем) и CloudSim (для облачных) [166; 167]. CloudSim — наиболее распространенная система, разработанная в университете Мельбурна. Ее возможности включают моделирование ЦОД, ВМ с настраиваемыми политиками, сетевых топологий, взаимодействия приложений, различных типов облачных инфраструктур, а также полный инструментарий управления моделированием.

Для работы с CloudSim необходимо обеспечить: наличие Java Runtime Environment версии не ниже 8; использование интегрированной среды разработки (рекомендуются Eclipse или NetBeans); включение пакета CloudSim в проект в качестве внешней библиотеки.

На рис. 5.1 приведена схема многоуровневой архитектуры CloudSim, отражающая организацию взаимодействия между компонентами платформы.

Ядро CloudSim реализует базовые механизмы дискретно-событийного моделирования, в частности:

– управление очередями событий;
– создание и взаимодействие моделируемых сущностей (entities);
– диспетчеризацию и синхронизацию хода симуляции.

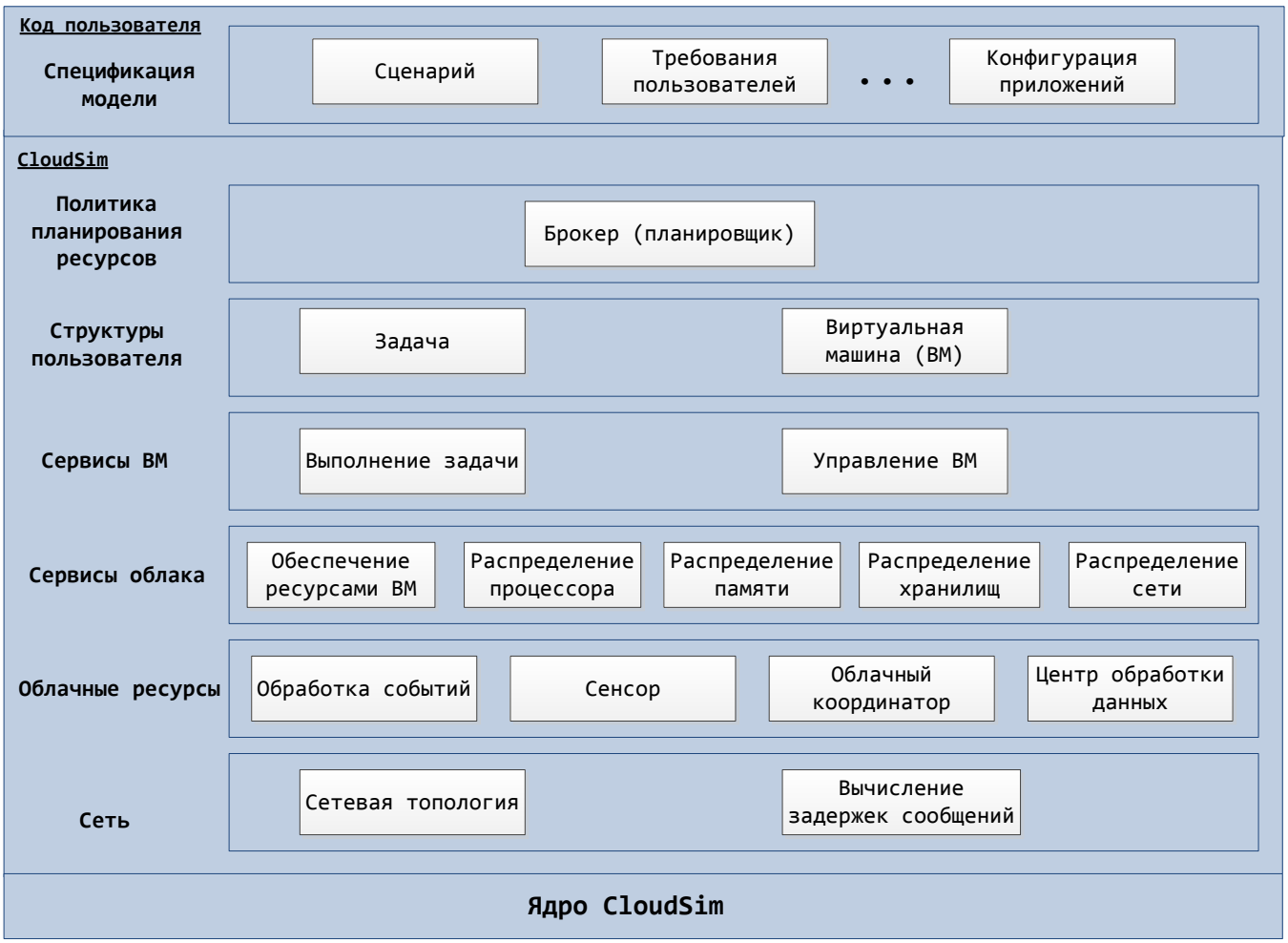

Рисунок 5.1 — Архитектура платформы CloudSim

Процесс разработки модели заключается в программной реализации описания системы и сценария эксперимента на языке Java. Разработчик может использовать как стандартные компоненты, поставляемые с платформой, так и собственные модифицированные классы. После запуска созданная модель передается ядру CloudSim, которое выполняет имитацию.

Процесс построения модели включает следующую последовательность шагов [168]:

1. Инициализация пакета CloudSim для размещения моделируемых сущностей (вызов метода CloudSim.init(...)).
2. Создание центра обработки данных — формирование списка физических хостов с заданными атрибутами.
3. Создание планировщика ресурсов (Broker) — агента, отвечающего за создание виртуальных машин с требуемыми характеристиками, их назначение на подходящие хосты и последующее удаление.

4. Создание виртуальных машин — описание их конфигурации в виде набора атрибутов.
5. Создание задач (cloudlet) — определение параметров вычислительных заданий и их привязка к брокеру.
6. Запуск моделирования.
7. Вывод результатов.

Библиотека CloudSim допускает интеграцию пользовательских алгоритмов управления, включая политики распределения ресурсов хоста между ВМ и эвристики размещения виртуальных машин. В работе [51] описано расширение CloudSim, позволяющее моделировать GPU-серверы.

Для имитационного моделирования моделей и методов, предложенных в данной работе, была использована библиотека CloudSim версии 3.0. Её функционал включает набор Java-классов для описания инфраструктуры облачного ЦОД и проведения экспериментов.

5.3 Основной цикл работ по управлению ресурсами облачных ЦОД

Разработанные в данной диссертационной работе модели и методы могут являться основной для цикла работ по управлению ресурсами облачного ЦОД, схематично представленного на рис. 5.2. На первом этапе пользователем задаются количество виртуальных машин и их размеры. На втором этапе выполняется оптимальное размещение ВМ на имеющихся серверах (как обычных, так и оснащенных GPU). Критериями оптимизации выступают энергопотребление, уровень нарушений SLA и равномерность загрузки ресурсов. Постановка и методы решения этой задачи подробно описаны в главе 2.

Графическое представление этапов статического размещения дано в приложении В (рис. В.1).

Непрерывный мониторинг ресурсов осуществляется локальными контроллерами системы управления. Для обеспечения стабильности функционирования применяется метод скользящего окна, оптимальный размер которого выбирается в зависимости от длительности миграции виртуальных машин (см. параграф 3.5). На начальном этапе, до накопления достаточной статистики,

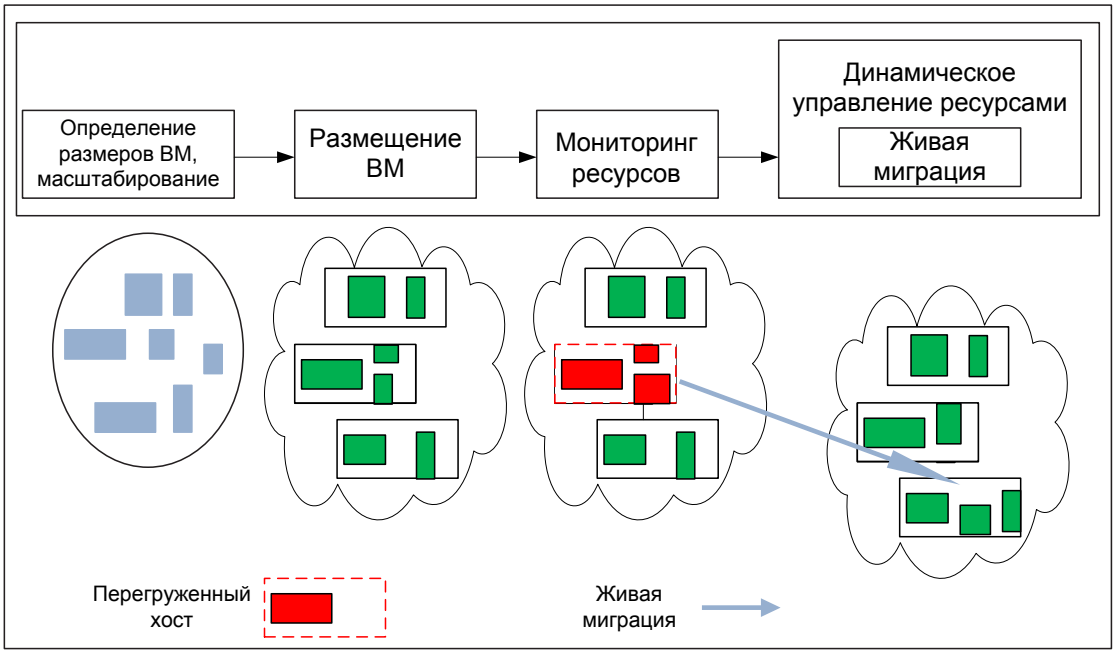

Рисунок 5.2 — Основной цикл работ по управлению ресурсами облачного ЦОД

размер окна для метрики загрузки процессора рекомендуется устанавливать в диапазоне от 3 до 5 минут.

Для оценки длительности миграции с требуемым уровнем вероятности применяется метод, предложенный в главе 3, использующий накопленные статистические данные о ранее выполненных миграциях.

Динамическое управление ресурсами осуществляется на основе данных системы мониторинга. Этапы динамического размещения приведены в приложении В на рис. В.2

Рассмотрим более подробно разработанные алгоритмы, включенные в имитационное моделирование, подтверждающие эффективность предложенного основного цикла работ по управлению ресурсами инфокоммуникационной системы ЦОД.

5.4 Имитационное моделирование динамического размещения

5.4.1 Обнаружение недогрузки хоста

Для определения того, что хост является недогруженным, предлагается простой эвристический алгоритм, приведенный в листинге 1. Алгоритм рассчитывает среднюю загрузку процессора за последние n измерений и сравнивает полученное значение с пороговым. Если средняя загрузка ниже порогового значения, то алгоритм обнаруживает, что хост недогружен. Алгоритм принимает три параметра: пороговое значение загрузки процессора, число последних замеров загрузки процессора для расчета среднего и массив измерений загрузки.

Алгоритм 1: Алгоритм определения недогруженного хоста

Вход: порог, n, загрузка[]
Выход: Является ли хост недогруженным

```
1 if загрузка не пусто then
2 |   загрузка ← последние n значений загрузки
3 |   средняяЗагрузка ← sum(загрузка) / len(загрузка)
4 |   return средняяЗагрузка ⩽ порог
5 else
6 |   return false
7 end
```

5.4.2 Обнаружение перегрузки хоста

Для каждого вычислительного узла собирается статистика его загрузки и температуры.

Для того, чтобы предотвратить возможную перегрузку и перегрев хоста необходимо периодически выполнять алгоритм обнаружения перегрузки и перегрева.

Предлагаемый алгоритм основан на прогнозе загрузки ЦП и температуры с использованием метода МГУА.

Псевдокод алгоритма представлен в листинге 2. На вход алгоритма подается загрузка ЦП за несколько окон наблюдений, далее вызывается метод МГУА, который возвращает оценки коэффициентов параметров модели порядка не выше заданного и рассчитывается прогноз загрузки на следующее окно наблюдения.

Если прогнозное значение превышает пороговое, то алгоритм возвращает логическое значение ИСТИНА, что означает что хост будет перегружен. Также в алгоритм введен параметр безопасности, позволяющий администратору системы масштабировать прогнозное значение для увеличения или уменьшения чувствительности алгоритма к потенциальным перегрузкам.

Алгоритм 2: Алгоритм обнаружения перегрузок сервера

Вход: порог, загрузка[], порядокМодели, параметрБезопасности, числоОкон

Выход: Является ли хост перегруженным

```
1  if len(загрузка) < числоОкон then
2  |  return false;
3  end
4  X ← 1:9;
5  оценка←MGUA(числоОкон, числоВходныхПеременных, X, загрузка[],
   |  числоТочекВОбучающейВыборке, числоТочекВПроверочнойВыборке,
   |  порядокМодели);
6  x←числоОкон;
7  switch порядокМодели do
8  |  case 1 do
9  |  |  прогноз ← оценка[0] + оценка[1]*x; break;
10 |  case 2 do
11 |  |  прогноз ← оценка[0] + оценка[1]*x + оценка[2]*x^2; break;
12 |  case 3 do
13 |  |  прогноз ← оценка[0] + оценка[1]*x + оценка[2]*x^2 +оценка[3]*x^3;
   |  |   break;
14 end
15 return параметрБезопасности * прогноз ⩾ порог
```

5.4.3 Алгоритм метода группового учета аргументов

Алгоритм комбинаторного метода группового учета аргументов (МГУА) имеет сложную структуру и состоит из нескольких последовательных блоков:

1. **Блок подготовки исходных данных**. На вход алгоритма подаются следующие параметры: число наблюдений (точек), число входных переменных, массив входных переменных, массив выходной переменной (загрузки), число точек в обучающей выборке, число точек в проверочной выборке, порядок модели (максимальная степень полинома).

2. **Блок формирования базиса**. В соответствии с порядком модели σ_{max} и числом входных переменных v рассчитывается число параметров полной модели (полинома Колмогорова–Габора) n по формуле числа сочетаний с повторениями:

$$n = \frac{(v + \sigma_{max})!}{\sigma_{max}! \cdot v!} = \prod_{j=1}^{v} \frac{\sigma_{max} + j}{j}. \tag{5.1}$$

Сам полный полином записывается в следующем общем виде:

$$y = \sum_{i=1}^{n} a_i \prod_{j=1}^{v} z_j^{\sigma_{ij}} = \sum_{i=1}^{n} a_i x_i, \tag{5.2}$$

где каждый обобщенный аргумент x_i представляет собой некоторую нелинейную функцию от исходных переменных z_j:

$$x_i = \prod_{j=1}^{v} z_j^{\sigma_{ij}}. \tag{5.3}$$

Здесь σ_{ij} — степень j-й переменной в i-м члене полинома.

В соответствии с описанной процедурой формируется матрица наблюдений обобщенных аргументов $\mathbf{X} \in \mathbb{R}^{N \times n}$, где N — общее количество выполненных измерений (наблюдений), а n — количество обобщенных аргументов.

Полученная совокупность данных разделяется на три непересекающиеся подвыборки:

– обучающая выборка A длины N_A;

– проверочная выборка B длины N_B;

– экзаменационная выборка C длины N_C.

При этом выполняется условие полноты:

$$N_A + N_B + N_C = N. \tag{5.4}$$

3. **Блок комбинаторного перебора и селекции моделей**. В данном блоке выполняются основные операции по генерации и отбору моделей.

Формализация структуры модели. Структура частной модели задается структурным вектором $\mathbf{d} = (d_1, d_2, \ldots, d_n)$, где $d_i \in \{0,1\}$. Значение $d_i = 1$ указывает на включение i-го обобщенного аргумента в модель, $d_i = 0$ — на его исключение. Для генерации всех возможных структурных векторов применяется схема, аналогичная работе двоичного счетчика.

Построение нормальных систем. Для повышения эффективности вычислений матрицы полной нормальной системы рассчитываются однократно:

$$\mathbf{X}^\mathsf{T}\mathbf{X} \in \mathbb{R}^{n\times n}, \quad \mathbf{X}^\mathsf{T}\mathbf{y} \in \mathbb{R}^n. \tag{5.5}$$

Для формирования любой частной нормальной системы, соответствующей вектору $\mathbf{d}$, достаточно извлечь из $\mathbf{X}^\mathsf{T}\mathbf{X}$ элементы, расположенные на пересечении строк и столбцов с индексами, где $d_i = 1$, а также соответствующие компоненты из вектора $\mathbf{X}^\mathsf{T}\mathbf{y}$.

Решение полученной системы линейных алгебраических уравнений

$$(\mathbf{X}_e^\mathsf{T}\mathbf{X}_e)\mathbf{a}_e = \mathbf{X}_e^\mathsf{T}\mathbf{y} \tag{5.6}$$

позволяет получить оценки коэффициентов $\mathbf{a}_e$ для текущей частной модели.

Вычисление критериев селекции. Для каждой частной модели могут быть вычислены следующие критерии:

– критерий регулярности (на проверочной выборке B);

– критерий несмещенности;

– критерий минимума смещения;

– комбинированные критерии.

Критерий несмещенности $n^2_{\text{см}}$ оценивает близость оценок модели, полученных на разных выборках:

$$n^2_{\text{см}} = (\hat{\mathbf{y}}_A - \hat{\mathbf{y}}_B)^\mathsf{T}(\hat{\mathbf{y}}_A - \hat{\mathbf{y}}_B), \tag{5.7}$$

где $\hat{\mathbf{y}}_A, \hat{\mathbf{y}}_B$ — оценка выходной величины, полученная на полных данных A и B по коэффициентам $\mathbf{a}_A$ и $\mathbf{a}_B$ соответственно.

Критерий регулярности $\Delta(B)$ оценивает точность модели на проверочной выборке:

$$\Delta(B) = (\mathbf{y}_B - \hat{\mathbf{y}}_B^A)^\mathsf{T}(\mathbf{y}_B - \hat{\mathbf{y}}_B^A) = \mathbf{y}_B^\mathsf{T}\mathbf{y}_B - 2\mathbf{a}_A^\mathsf{T}\mathbf{X}_B^\mathsf{T}\mathbf{y}_B + \mathbf{a}_A^\mathsf{T}\mathbf{X}_B^\mathsf{T}\mathbf{X}_B\mathbf{a}_A, \qquad (5.8)$$

где $\mathbf{y}_B$ — вектор измерений выходной величины на выборке B, а $\hat{\mathbf{y}}_B^A$ — оценка выхода на выборке B по модели, оцененной на выборке A (с коэффициентами $\mathbf{a}_A$).

Для финальной оценки отобранных моделей может использоваться независимая экзаменационная выборка C:

$$\Delta(C) = (\mathbf{y}_C - \hat{\mathbf{y}}_C^W)^\mathsf{T}(\mathbf{y}_C - \hat{\mathbf{y}}_C^W) = \mathbf{y}_C^\mathsf{T}\mathbf{y}_C - 2\mathbf{a}_W^\mathsf{T}\mathbf{X}_C^\mathsf{T}\mathbf{y}_C + \mathbf{a}_W^\mathsf{T}\mathbf{X}_C^\mathsf{T}\mathbf{X}_C\mathbf{a}_W, \qquad (5.9)$$

где $\mathbf{a}_W$ — оценки коэффициентов, пересчитанные на объединенной выборке $W = A \cup B$.

Последовательный отбор по критериям. Для повышения селективности применяется многоэтапная селекция:

1. Первичный отбор выполняется по критерию минимума смещения $n^2_{\text{см}}$. Из всего множества моделей-претендентов отбирается F_1 наименее смещенных моделей.
2. Вторичный отбор осуществляется среди F_1 отобранных моделей по критерию регулярности $\Delta(B)$. Для дальнейшего рассмотрения оставляется F моделей, имеющих наименьшую ошибку на данной выборке:

$$\Delta(B) = \frac{1}{N_B}\sum_{t\in B}(y_t - \hat{y}_t(\mathbf{a}_A))^2 \to \min_{F_1} \Rightarrow F. \qquad (5.10)$$

Селекция продолжается до тех пор, пока не будет достигнут минимум выбранного критерия (например, критерия смещения), после чего отбирается наиболее регулярная модель.

4. **Блок оценки качества лучших моделей**. На этом этапе выполняется углубленный анализ качества отобранных моделей-претендентов для окончательного выбора оптимальной. Он может включать:

– вычисление дополнительных критериев (например, финальной ошибки на выборке C);
– проверку устойчивости модели к вариациям исходных данных;
– анализ статистической значимости коэффициентов.

Ошибка МНК после пересчета коэффициентов на объединенной выборке W может быть выражена через нормальные матрицы:

$$E^2(W) = (\hat{\mathbf{y}}_W - \mathbf{y}_W)^\mathsf{T}(\hat{\mathbf{y}}_W - \mathbf{y}_W) = \mathbf{y}_W^\mathsf{T}\mathbf{y}_W - \mathbf{a}_W^\mathsf{T}\mathbf{X}_W^\mathsf{T}\mathbf{y}_W, \qquad (5.11)$$

так как $\mathbf{a}_W^\top \mathbf{X}_W^\top \mathbf{X}_W \mathbf{a}_W = \mathbf{a}_W^\top \mathbf{X}_W^\top \mathbf{y}_W$.

5. **Блок вывода результатов**. В программе предусмотрен вывод параметров нескольких лучших моделей, а также расчет прогнозного значения по одной выбранной лучшей модели.

Разработанный программный модуль представляет собой реализацию комбинаторного метода группового учета аргументов на объектно-ориентированном языке Java. В приложении приведена копия свидетельства о регистрации программы для ЭВМ [116].

Модуль предназначен для прогнозирования загрузки серверов в системе имитационного моделирования облачных центров обработки данных CloudSim, а также может быть встроен в монитор виртуальных машин (гипервизор).

5.4.4 Выбор виртуальных машин

Следующим этапом процесса динамического управления ресурсами является определение множества виртуальных машин, подлежащих переносу на другие физические серверы. Наиболее широкое распространение на практике получил алгоритм выбора ВМ с минимальным временем миграции (ММТ — Minimum Migration Time).

В соответствии с данным алгоритмом приоритет отдается виртуальной машине, характеризующейся наименьшим объемом оперативной памяти. Данный выбор основан на предположении, что ВМ с меньшим объемом RAM потребует минимального времени для выполнения миграции [79]. В случае, если несколько ВМ имеют одинаковый (минимальный) объем памяти, дальнейший выбор осуществляется по критерию максимальной загрузки процессора среди этих кандидатов.

В листинге 3 приведен псевдокод алгоритма выбора ВМ с минимальным временем на миграцию и максимальной загрузкой процессора.

Алгоритм 3: Алгоритм выбора ВМ с минимальным временем на миграцию и максимальной загрузкой процессора

Вход: n, загрузкаЦП[], объёмОП[]
Выход: выбраннаяВМ

```
1  минОП ← min(объёмОП)
2  максЦП ← 0
3  выбраннаяВМ ← None
4  for ВМ, ЦП в загрузкаЦП do
5      if объёмПамятиВМ[ВМ] > минОП then
6          continue
7      end
8      значенияЦП ← последние n значений ЦП
9      среднее ← sum(значенияЦП) / len(значенияЦП)
10     if максЦП < среднее then
11         максЦП ← среднее
12         выбраннаяВМ ← ВМ
13     end
14 end
15 return выбраннаяВМ
```

5.4.5 Размещение ВМ на физических серверах

Размещение виртуальной машины производится с использованием метода, разработанного в параграфе 4.2. Имеется список ВМ для миграции (VMList) и список хостов, на которые можно переместить виртуальные машины (hostList). Если их число не равно, то задачу о назначениях нужно привести к симметричному виду, путем ввода фиктивных узлов.

Далее необходимо рассчитать коэффициенты целевой функции. Для этого необходимо знать последние значения загрузки процессора и памяти. В соответствии с заданными весовыми коэффициентами рассчитывается значение целевой функции. Если доступных ресурсов для размещения ВМ на сервере не хватает, то соответствующему коэффициенту целевой функции ставится заведомо неэффективное значение, равное размерности задачи N. После этого вызывается венгерских метод, который возвращает оптимальное назначение

ВМ по физическим серверам. Псевдокод данного алгоритма приведен в листинге 4.

5.4.6 Рабочая нагрузка

Для проведения имитационного моделирования важно использовать нагрузку реальной системы. В библиотеке CloudSim имеется нагрузка, полученная в рамках проекта инфраструктуры мониторинга CoMon в глобальной исследовательской сети PlanetLab [163; 169].

В качестве исходных данных для моделирования использован набор данных, содержащий показатели загрузки процессора, собранные с более чем 1000 виртуальных машин, размещенных на серверах, распределенных по всему миру. Измерения проводились в течение 10 случайным образом выбранных дней с интервалом дискретизации 5 минут.

Во время моделирования каждой ВМ случайным образом присваивается одна из нагрузок ВМ отдельного дня. Ограничения на объем памяти при моделировании не учитывались.

5.4.7 Параметры серверов и виртуальных машин

Число хостов и ВМ варьировалось в зависимости от эксперимента. Моделировалось 2 типа физических серверов в равных долях: HP ProLiant ML110 G4 и HP ProLiant ML110 G5. Тактовая частота процессоров измерялась в миллионах инструкций в секунду (MIPS): 1860 MIPS каждое ядро для сервера HP ProLiant ML110 G4 и 2660 MIPS для сервера HP ProLiant ML110 G5. Каждый сервер имеет пропускную способность 1 Гбит/с.

Энергопотребление серверов изменяется в зависимости от загрузки. Вместо аналитической модели энергопотребления сервером в работе используются реальные данные об энергопотреблении, представленные результатами опубликованных тестов SPECpower для данных типов серверов [170]. Энергопотребление выбранных серверов в зависимости от загрузки приведено в таблице 15. Для

Алгоритм 4: Многокритериальный алгоритм размещения виртуальной машины

Вход: N, VM_i^{CPU}, VM_i^{RAM}, $PM_j^{CPU_0}$, $PM_j^{RAM_0}$, $PM_j^{RAM_1}$, $PM_j^{CPU_1}$, α_1,α_2

Выход: Назначения y

for *i in 1:N* **do**
  $u_{CPU}[i,] = (VM\$CPU[i] + PM\$CPU_1)/PM\$CPU_0$
  $u_{RAM}[i,] = (VM\$RAM[i] + PM\$RAM_1)/PM\$RAM_0$
end
for *i in 1:N* **do**
  for *j in 1:N* **do**
    if $(u_{CPU}[i,j] > 1)$ **then**
      $u_{CPU}[i,j] = NaN$
    end
    if $(u_{RAM}[i,j] > 1)$ **then**
      $u_{RAM}[i,j] = NaN$
    end
  end
end
$f_{res} \leftarrow 1 - u_{CPU} * u_{RAM}$
$f_{sla} \leftarrow 1/(1 + exp(-(u_{CPU} \cdot 100 - 80)))$
$C \leftarrow \alpha_1 \cdot f_{res} + alpha_2 \cdot f_{sla}$
for *i in* $1 : N$ **do**
  for *j in* $1 : N$ **do**
    if *C[i,j] is not a number* **then**
      $C[i,j] \leftarrow N$
    end
  end
end
y $\leftarrow$ HungarianMethod(C)
if $F > N$ **then**
  print ("Не хватает ресурсов для floor(F/N), "виртуальных машин")
end

Таблица 14 — Параметры нагрузки (загрузка процессора)

Дата	Число ВМ	Среднее	Стандартное отклонение	Первая квартиль	Медиана	Третья квартиль
03/03/2011	1052	12.31%	17.09%	2%	6%	15%
06/03/2011	898	11.44%	16.83%	2%	5%	13%
09/03/2011	1061	10.70%	15.57%	2%	4%	13%
22/03/2011	1516	9.26%	12.78%	2%	5%	12%
25/03/2011	1078	10.56%	14.14%	2%	6%	14%
03/04/2011	1463	12.39%	16.55%	2%	6%	17%
09/04/2011	1358	11.12%	15.09%	2%	6%	15%
11/04/2011	1233	11.56%	15.07%	2%	6%	16%
12/04/2011	1054	11.54%	15.15%	2%	6%	16%
20/04/2011	1033	10.43%	15.21%	2%	4%	12%

упрощения модели для серверов взяты двухядерные процессоры, что является достаточным, чтобы показать, как многоядерные процессоры обрабатывают предлагаемые алгоритмы.

Таблица 15 — Энергопотребление выбранного сервера при различных уровнях загрузки в Вт.

Сервер	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
HP ProLiant G4	86	89.4	92.6	96	99.5	102	106	108	112	114	117
HP ProLiant G5	93.7	97	101	105	110	116	121	125	129	133	135

Характеристики ВМ соответствуют экземплярам ВМ Amazon EC2 с тем исключением, что все ВМ являются одноядерными, что объясняется тем фактом, что нагрузка, используемая в имитационном моделировании, получена с одноядерных ВМ.

При моделировании использовались ВМ следующих типов:

– высокоскоростной средний экземпляр (2500 MIPS, 0.85 ГБ ОЗУ);

– большой экземпляр (2000 MIPS, 3.75 ГбБ ОЗУ);

– малый экземпляр (1000 MIPS, 1.7 ГБ ОЗУ);

– микро экземпляр (500 MIPS, 0,613 ГБ ОЗУ).

Первоначально все ВМ были размещены в соответствии со своей потребностью в ресурсах, определенные типом ВМ. Однако с течением времени ВМ использует меньше ресурсов в соответствии с нагрузкой, что дает возможность их динамического размещения на меньшем числе серверов.

5.5 Предложения по реализации разработанных методов распределения

5.5.1 Архитектура системы

Система управления ресурсами облачного ЦОД имеет двухуровневую архитектуру, состоящую из глобального и локальных контроллеров (рис. 3.1). Двухуровневая система управления ресурсами предпочтительнее централизованного подхода, при котором все функции управления реализуются централизованно. Поскольку виртуальные машины независимы друг от друга и могут реализовываться на различных платформах, то возможны различные реализации локальных контроллеров. Все внутренние сложности функций управления в виртуальных машинах и контейнерах переводятся локальными контроллерами в простые запросы на объем необходимых ресурсов.

Локальные контроллеры анализируют состояние физических серверов, на которых они расположены и определяют возможные состояния недогрузки, перегрузки и перегрева на основании прогноза для следующего окна наблюдения. В случае выявления одного из перечисленных состояний, локальный контроллер сообщает об этом глобальному контроллеру, который выбирает сервер назначения, на который будет производиться миграция ВМ.

В данной системе локальный контроллер отвечает на следующий вопрос:

– Когда нужно производить перемещение ВМ?

Тогда как глобальный

– Какие ВМ выбрать для миграции?

– Куда перемещать ВМ?

Для этого локальные контроллеры постоянно анализирует данные системы мониторинга, и в случае выхода параметров рабочей модели за допустимый уровень сообщают об этом глобальному контроллеру, который активизирует процесс миграции виртуальных машин.

Проверка показателей системы осуществляется последовательно, в соответствии с важностью критериев (перегрев, перегрузка, недогрузка хоста), при этом процесс наблюдения осуществляется непрерывно, в том числе в случае выполнения действий по перемещению ВМ. Приоритет критериев может быть

изменен администратором системы, однако в силу их противоречивости этап проверки должен оставаться последовательным.

В случае возникновения условия для миграции ВМ, глобальный контроллер определяет ВМ на проблемных хостах и запускает процесс поиска оптимального хоста назначения. Выбор хоста назначения осуществляется в соответствии с алгоритмом выбора хоста назначения, рассмотренным в параграфе 3.5. После определения хоста назначения и запуска процесса перемещения, данный физический сервер временно изымается из списка доступных узлов до окончания миграции на него ВМ и получения статистической информации о стабильности системы.

Обобщенный алгоритм оптимизации размещения ВМ, приведен на рис. 3.2. Вначале просматривается список хостов и определяется список перегруженных серверов. В случае обнаружения перегруженного хоста запускается процедура выбора ВМ для миграции с этого хоста. Выбранные ВМ заносятся в список, после чего запускается алгоритм выбора хоста назначения для перемещения на него ВМ. Второй ветвью алгоритма является определение незагруженных хостов и миграция ВМ с этих хостов. Алгоритм возвращает карту миграции, которая содержит информацию о новом размещении выбранных ВМ как в случае перегруженных, так и недогруженных хостов.

5.5.2 Глобальный контроллер

Глобальный контроллер развернут на главном хосте и отвечает за принятие решений о размещении виртуальных машин и инициирование миграции виртуальных машин. Он принимает запросы от локальных контроллеров трёх типов:

1. недогрузка хоста;
2. перегрузка хоста.
3. перегрев хоста.

Обработка запроса первого типа состоит в следующем. Во-первых, локальный менеджер определяет недогрузку узла с использованием алгоритма обнаружения недогрузки сервера. Затем, он отправляет запрос о недогрузке к глобальному контроллеру, указав в нем имя недогруженного хоста. Глобаль-

ный контроллер запрашивает в центральной БД список ВМ, размещенных на данном хосте, а также данные об использовании ресурсов и состоянии физических хостов. Затем все эти сведения передаются в блок прогнозирования, в котором осуществляется прогноз состояния каждого сервера после перемещения на него виртуальной машины, после чего список хостов передается в блок принятия решения, реализующий алгоритм выбора сервера для размещения виртуальных машин.

Затем, в соответствии с алгоритмом размещения ВМ, глобальный контроллер обращается к гипервизору с запросом на живую миграцию соответствующих ВМ и контролирует процесс миграции виртуальных машин для определения времени завершения миграции. После завершения миграции виртуальной машины глобальный контроллер переключает разгруженный сервер в спящий режим.

Обработка запроса второго и третьего типов похожа на запросы первого типа. Разница состоит в том, что вместо отправки только имени хоста, локальный контроллер также отправляет список идентификаторов виртуальных машин, выбранных с использованием соответствующего алгоритма. Как только список ВМ получен, глобальный контроллер передает список в качестве параметра в алгоритм размещения виртуальных машин. Если некоторые виртуальные машины, размещенные на узле, находятся в спящем режиме, глобальный контроллер активирует их с использованием технологии Wake-on-LAN. Затем, глобальный контроллер отправляет гипервизору запрос на миграцию ВМ.

5.5.3 Локальный контроллер

Локальный контроллер развернут на каждом вычислительном узле как служба ОС в фоновом режиме. Служба периодически выполняет функцию, которая определяет, нужно ли перераспределять виртуальные машины с хоста. В начале каждой итерации контроллер читает из локального хранилища статистические данные о средней загрузке и температуре процессора за время окна, которые записываются сборщиком данных. На основе считанных данных последовательно проверяются три условия: перегрев, перегрузка и недогрузка хоста,

при этом делается прогноз состояния сервера на ближайший период наблюдения с использованием МГУА. Если, например, сервер недогружен, локальный контроллер отправляет запрос недогрузки к глобальному контроллеру для того, чтобы перенести все ВМ с этого хоста, а сам хост перевести в спящий режим. Общий алгоритм работы контролеров приведен на рис. 3.2.

5.5.4 Блок сбора данных

Блок сбора данных развернут на всех вычислительных узлах в качестве службы хостовой операционной системы (или гипервизора), работающей в фоновом режиме. Служба периодически собирает данные об использовании ЦП для каждой виртуальной машины, запущенной на хосте, а также данные об использовании процессора гипервизором. Собранные данные хранятся в локальном файловом хранилище, а также передаются в центральную базу данных. Данные хранятся как среднее число МГц, потребляемой виртуальной машиной за последний интервал измерения длины τ. В частности, использование ЦП $C_i^v(t_0,t_1)$ виртуальной машиной i, которое является функцией от границ интервала измерения $[t_0, t_1]$, рассчитывается как

$$C_i^v(t_0,t_1) = (n_i^v F(\tau_i^v(t_1) - \tau_i^v(t_0))/(t_1 - t_0), \tag{5.12}$$

где n_i^v—число виртуальных ядер процессора, выделенных виртуальной машине i;

F—частота одного процессорного ядра в МГц;

τ_i^v—процессорное время, выделенное ВМ i до момента времени t.

Использование процессора гипервизором рассчитывается как разница между общим потреблением процессора и потреблением процессора виртуальными машинами, размещенных на данном хосте.

$$C_j^h(t_0,t_1) = \frac{n_j^h F(\tau_j^h(t_1) - \tau_j^h(t_0))}{(t_1 - t_0)} - \sum_{i \in V_j} C_i^v(t_0,t_1), \tag{5.13}$$

где n_j^h–число физических ядер хоста j;

$\tau_j^h(t)$–процессорное время, потребленное всем хостом до момента времени t;

v_j–набор виртуальных машин, размещенных на хосте j.

Данные об использовании процессора сохраняются как целые числа. Сохраненные значения могут быть приблизительно преобразованы в загрузку процессора для любого типа хоста или виртуальной машины.

В начале каждой итерации сборщик данных получает список виртуальных машин, работающих в данный момент на хосте с помощью API гипервизора и сравнивает их со списком виртуальных машин, работающих на предыдущем шаге времени. Если были обнаружены новые виртуальные машины, то сборщик данных выбирает статистические данные о них из центральной базы данных и сохраняет данные в локальном файловом хранилище. Если некоторые ВМ были удалены, сборщик данных удаляет сведения об этих виртуальных машинах из локального хранилища данных.

В данной работе для исключения ошибочного определения перегрузки сервера предлагается двухэтапная процедура проверки условий. На первом этапе применяется метод скользящего окна: данные, собранные за временное окно, усредняются и сравниваются с предыдущими значениями. На втором этапе осуществляется прогнозирование перегрузки и температурных показателей серверов с использованием метода МГУА. Существенное отклонение прогнозируемых значений от установленного порога свидетельствует о необходимости миграции виртуальной машины.

5.5.5 Хранилища данных

Система управления ресурсами содержит два типа хранилищ данных:

- Центральная база данных – сервер базы данных, который может быть развернут либо на сервере глобального контроллера, или на одном или нескольких выделенных узлах. При большом количестве хостов (>1000) для хранения метрик рекомендуется использовать распределённую базу данных (например, Cassandra).
- Локальное файловое хранилище данных – хранилище данных, расположенное на каждом вычислительном узле, и используется для временного кэширования данных об использовании ресурсов для использования локальными контроллерами.

Центральная база данных используется для хранения статистических данных об использовании ресурсов виртуальных машин и гипервизоров, а также аппаратных характеристик хостов. База данных заполняется блоками сбора данных, развернутых на вычислительных узлах. Есть два основных случаях, когда данные извлекаются из базы данных, вместо локальной памяти вычислительных узлов.

Во-первых, база данных используется локальными контроллерами, чтобы получить данные об использовании ресурсов после миграции виртуальных машин. После завершения миграции виртуальной машины сборщик данных, развернутый на узле назначения извлекает необходимые статистические данные из базы данных и сохраняет их локально, чтобы в дальнейшем их использовать.

Второй вариант использования центральной базы данных, когда глобальный контроллер вычисляет новое размещение виртуальных машин на хостах. Алгоритмам размещения ВМ необходима информация о потреблении ресурсов всеми узлами, чтобы принимать глобальные решения о распределении.

Поэтому, каждый раз, когда есть необходимость разместить виртуальные машины на хостах, глобальный контроллер запрашивает базу данных, чтобы получить актуальную информацию по использованию ресурсов гипервизорами и виртуальными машинами.

Схема БД приведена в приложении В на рис. В.3.

Для поддержки расширенного функционала управления ресурсами облачного ЦОД, включая учет GPU и отслеживание истории размещений, была разработана схема базы данных, включающая восемь взаимосвязанных таблиц.

Основные таблицы хранения статических данных:

Хосты (Hosts) – содержит информацию о физических серверах: имена узлов, частоту процессора, количество ядер CPU, объём оперативной памяти, характеристики GPU (количество, тип, общий объём памяти), а также архитектурные особенности.

Устройства GPU (GPU_Devices) – детализированная информация об отдельных GPU устройствах: уникальные идентификаторы, модели, производители, объём памяти, версии драйверов и текущий статус доступности.

Виртуальные машины (VM) – содержит данные о виртуальных машинах: идентификаторы, имена, владельцев, выделенные ресурсы (ядра CPU, RAM, GPU) и текущее состояние.

Таблицы мониторинга и динамических данных:

Использование ресурсов хоста (Hosts_Usage) – хранит исторические данные о потреблении ресурсов физическими серверами: загрузку CPU, использование RAM и GPU-памяти, потребление энергии, сетевую активность.

Использование ресурсов ВМ (VM_Usage) – фиксирует метрики использования ресурсов виртуальными машинами: загрузку CPU, использование RAM и GPU, активность ввода-вывода.

Таблицы управления размещением и миграциями:

Размещение ВМ (VM_Placement) – отслеживает историю размещения виртуальных машин на физических хостах с указанием периодов размещения, причин и приоритетов.

Выделение GPU (VM_GPU_Allocation) – управляет привязкой конкретных GPU устройств к виртуальным машинам с фиксацией периодов использования и типов выделения.

Журнал миграций (Migration_Log) – регистрирует все операции миграции ВМ: исходные и целевые хосты, время, длительность, статус выполнения и объём перенесённых данных.

Данная расширенная схема обеспечивает полный цикл управления ресурсами ЦОД: от мониторинга текущего состояния до анализа исторических данных для оптимизации размещения, поддержки многокритериальных алгоритмов распределения и обеспечения учёта SLA. Интеграция с компонентами управления (глобальный и локальные контроллеры) осуществляется через централизованный доступ к этим таблицам.

Локальное файловое хранилище. Локальный диспетчер на каждой итерации требует данных об использовании ресурсов виртуальными машинами и гипервизором соответствующего узла для того, чтобы определить недогрузку, перегрузку или перегрев хоста и передать соответствующий запрос глобальному контроллеру. Чтобы уменьшить объем передаваемых данных мониторинга состояния хоста по сети, данные сначала сохраняются локально в виде текстового файла и хранятся там некоторое время. Таким образом, локальный контроллер может читать данные из локального хранилища файла без обращения к центральной БД. Для каждой виртуальной машины создается свой файл с именем пользователя виртуальной машины. Данные об использовании ресурсов гипервизором хранятся в отдельном файле. Интервал съёма данных с датчиков составляет 3-7 секунд, после чего данные усредняются по числу точек, входящих в окно наблюдения. В центральную БД передаются усредненные данные.

5.5.6 Интеграция с OpenStack

Разработанные методы предлагается использовать в составе подсистемы распределения ресурсов в облачных вычислительных средах, например OpenStack.

В контексте решения задачи оптимизации размещения виртуальных машин наибольший интерес представляют компоненты Nova и Ceilometer. Nova отвечает за планирование и запуск ВМ, что непосредственно связано с задачей выбора хоста для размещения. Ceilometer обеспечивает сбор данных о потреблении ресурсов (загрузка CPU, памяти, сети), которые необходимы для оценки текущего состояния хостов и принятия решений о миграции. Остальные компоненты (Glance, Cinder, Neutron и др.) создают инфраструктурную основу, но не участвуют напрямую в алгоритмах размещения.

Основные компоненты платформы OpenStack, которые могут использоваться как совместно, так и независимо друг от друга, включают:

- **Nova** — вычислительный компонент (планировщик), отвечающий за запуск экземпляров виртуальных машин (instance) в облачной среде и управление ими. Поддерживает гипервизоры KVM, Xen, VMware, Microsoft Hyper-V и другие.
- **Swift** — объектное хранилище (Object Storage). Доступ к файлам осуществляется по протоколу HTTP.
- **Glance** — хранилище образов виртуальных машин. Может работать поверх обычной файловой системы или использовать Swift в качестве бэкенда.
- **Cinder** — компонент для работы с блочными устройствами, предоставляющий виртуальным машинам неразмеченные диски по протоколу iSCSI.
- **Neutron** — компонент для управления сетями (SDN), обеспечивающий коммутацию и маршрутизацию пакетов.
- **Ceilometer** — компонент для учета потребленных ресурсов (биллинг и мониторинг).
- **Horizon** — веб-интерфейс для пользователей и администраторов облачной среды.

– **Keystone** — компонент аутентификации, авторизации и хранения конфигурации. Интегрируется с LDAP и Microsoft Active Directory.
– **Heat** — компонент оркестрации, предназначенный для автоматизированного развертывания приложений по сценарию на нескольких виртуальных машинах.

Разработанные методы могут быть интегрированы в облачную платформу OpenStack через следующие механизмы:

– **Сбор данных**: предлагается использовать Ceilometer, который уже собирает метрики использования ресурсов. Разработанные алгоритмы могут подписываться на события Ceilometer через его API.
– **Управление ВМ**: глобальный контроллер взаимодействует с Nova Scheduler через REST API для инициирования живых миграций. Пример запроса на Python для миграции ВМ:

```
import novaclient.v2.client as nvclient

nova = nvclient.Client(...)
server = nova.servers.find(name="vm-01")
nova.servers.live_migrate(server, host="compute-node-02")
```

– **Аутентификация**: Keystone используется для авторизации всех запросов к OpenStack API.
– **Мониторинг**: Horizon может быть расширен для визуализации состояния системы управления ресурсами.

Использование стандартных компонентов OpenStack уменьшает сложность развертывания и повышает совместимость с существующей инфраструктурой.

Обобщенная схема алгоритма работы платформы OpenStack с разработанными модулями приведена на рис. B.4.

Интеграция с OpenStack реализуется через расширение модуля Nova Scheduler путем добавления собственных фильтров (CustomGpuLoadFilter, CustomEnergyFilter) и взвешивателей (CustomMultiCriteriaWeigher). Для сбора метрик используется Ceilometer, который передает данные о загрузке CPU/GPU в центральную базу данных. Placement API расширяется для поддержки новых типов ресурсов (например, VGPU), что позволяет учитывать гетерогенность оборудования при размещении ВМ.

Для интеграции алгоритмов в OpenStack необходимо внести следующие изменения в файл `/etc/nova/nova.conf`:

```
[filter_scheduler]
enabled_filters = AvailabilityZoneFilter, ComputeFilter,
CustomGpuLoadFilter, CustomEnergyFilter
weighers = CustomMultiCriteriaWeigher, RamWeigher
```

В модуле `nova/scheduler/filters/` реализуются классы CustomGpuLoadFilter и CustomEnergyFilter, которые проверяют загрузку GPU и энергопотребление хоста соответственно.

Примеры API-запросов для взаимодействия с OpenStack

Для инициации миграции виртуальной машины используется запрос к Nova API:

```
nova live-migration <vm_id> <target_host>
```

Для регистрации нового типа ресурса (например, VGPU) в Placement API отправляются следующие запросы:

```
openstack resource class create VGPU
openstack resource provider inventory set <host_uuid> VGPU total=8
```

Процесс развертывания подсистемы

Развертывание подсистемы включает следующие шаги:

1. Установка дополнительных пакетов на узлы OpenStack:

   ```
   pip install custom-scheduler-plugin
   ```

2. Настройка базы данных для хранения метрик:

```
mysql -u root -p -e "CREATE DATABASE resource_monitor;"
```

3. Развертывание локальных контроллеров на каждом вычислительном узле:

```
systemctl enable local-controller
systemctl start local-controller
```

4. Настройка глобального контроллера на отдельном сервере с минимальными требованиями: 8 ядер CPU, 16 ГБ ОЗУ.

При интеграции с OpenStack может возникнуть проблема совместимости версий: алгоритмы тестировались на OpenStack Stein и Train. Для более ранних версий может потребоваться адаптация API.

5.6 Выводы

1. Разработана архитектура распределенного планировщика ресурсов для облачного центра обработки данных (ЦОД), включающая:
   - двухуровневую структуру с глобальным контроллером и локальными контроллерами на каждом физическом узле;
   - механизм децентрализованного принятия решений, сочетающий локальный мониторинг и глобальную оптимизацию размещения виртуальных машин;
   - поддержку динамического перераспределения ресурсов на основе прогнозных моделей загрузки.
2. Реализована платформа имитационного моделирования на базе CloudSim 3.0 для валидации предложенной архитектуры и алгоритмов.
3. Спроектирована расширенная схема базы данных распределенной системы управления.
4. Разработаны и верифицированы алгоритмы распределенного планирования, включающие:
   - алгоритмы обнаружения недогрузки и перегрузки хостов на основе скользящего окна и прогноза МГУА;

- алгоритм выбора виртуальных машин для миграции с учетом времени миграции и загрузки CPU (MMT-модификация);
- многокритериальный алгоритм размещения виртуальных машин с использованием венгерского метода и взвешенных критериев (ресурсная эффективность, SLA);
- реализацию МГУА на Java для интеллектуального прогнозирования нагрузки серверов.

5. Разработаны рекомендации по интеграции распределенного планировщика в промышленную облачную платформу OpenStack.
6. Разработанные модели, алгоритмы и программные компоненты готовы к включению в системы управления ресурсами облачных ЦОД и могут быть адаптированы для использования в гибридных и распределенных инфокоммуникационных системах.

Заключение

Основные результаты работы заключаются в следующем.

1. Проанализированы проблемы повышения эффективности функционирования гетерогенных центров обработки, среди которых выделены недостаточная энергоэффективность инфокоммуникационной системы гетерогенного ЦОД и стабильность облачных сервисов.
2. Разработаны критерии и метод многокритериального первоначального размещения разнородных виртуальных машин на физических серверах с поддержкой GPU-вычислений. В основе данного метода лежит многокритериальная задача оптимизации с критериями энергопотребления, использования ресурсов, нарушения SLA-соглашения. Показано, что решение задачи муравьиным алгоритмом с использованием обобщённого критерия позволяет достичь баланса между данными противоречивыми критериями по сравнению с эвристическими алгоритмами размещения.
3. Исследовано применение метода группового учета аргументов для прогнозирования загрузки физических серверов на реальных данных, дающего более точный прогноз по сравнению с известными классическими статическими методами, что подтверждено имитационным моделированием.
4. Разработан критерий определения рационального размера окна наблюдения за серверами для чего исследованы характеристики процесса миграции виртуальных машин и предложен метод расчета вероятности длительности миграции. Предложенный подход позволяет дать оценку вероятности миграции заданной длительности при определении размера скользящего окна.
5. Разработан метод многокритериального динамического размещения виртуальных машин, который позволяет найти лучшее размещение по комбинированному критерию энергопотребления и нарушения SLA-соглашений за приемлемое время.
6. Разработаны имитационные модели распределения ресурсов облачного ЦОД, которые позволяют проверить эффективность разработанных методов и алгоритмов динамического распределения ресурсов.

7. Разработан программный модуль для прогнозирования загрузки серверов в системе имитационного моделирования облачных центров обработки данных CloudSim, который может быть встроен в монитор виртуальных машин (гипервизор). Также программный модуль может использоваться как реализация комбинаторного метода группового учета аргументов на языке Java.
8. Разработаны алгоритмы и рекомендации по практической реализации моделей и методов программно-аппаратного комплекса распределенного планировщика ресурсов в инфокоммуникационной системе облачного ЦОД, которые могут быть включены в отечественные облачные платформы.

Список сокращений и условных обозначений

ВМ	виртуальная машина
ИКТ	информационные коммуникационные технологии
МВМ	монитор виртуальных машин, гипервизор
МГУА	метод группового учёта аргументов
ОС	операционная система
ЦОД	центр обработки данных
ACO	Ant Colony Optimization, муравьиный алгоритм
BFD	Best Fit Decreasing, алгоритм «наилучший подходящий, отсортированный по убыванию»
CPU, ЦП	Central Processing Unit, центральный процессор
FFD	First Fit Decreasing, алгоритм «первый подходящий, отсортированный по убыванию»
GPU	Graphical Processing Unit, графический процессор
HTTP	HyperText Transfer Protocol, протокол передачи гипертекста
IaaS	Infrastructure as a Service, инфраструктура как услуга
MIG	Multi-Instance GPU
MMT	Minimum Migration Time, алгоритм выбора ВМ с минимальным размером ОП
OTF	Overload Time Fraction, доля времени, в течение которого активные хосты испытывают загрузку 100 %
PaaS	Platform as a Service, платформа как услуга
PDM	Performance Degradation due to Migrations, ухудшение производительности ВМ из-за миграций
PUE	Power Usage Effectiveness, коэффициент энергоэффективности
RAM, ОП	Random Access Memory, оперативная память
SaaS	Software as a Service, программное обеспечение как услуга
SLA	Service Level Agreement, соглашение об уровне сервиса
SLAV	SLA Violation, показатель уровеня нарушений SLA в системе
TCO	Total Cost of Ownership, совокупная стоимость владения

Список литературы и источников информации

1. *Toutov, A. V.* Analysis of Data Center Development Problems in the Era of Digital Transformation / A. V. Toutov, N. V. Toutova, A. S. Vorozhtsov // International Workshop on Model-Driven Organizational and Business Agility. — Springer. 2022. — С. 65—72.
2. Программа "Цифровая экономика Российской Федерации". — 2017. — URL: http://static.government.ru/media/files/9gFM4FHj4PsB79I5v7yLVuPgu4bvR7M0.pdf.
3. Национальный проект «Экономика данных и цифровая трансформация государства». — 2025. — URL: http://government.ru/rugovclassifier/923/about/ ; Дата обращения: 22.08.2025.
4. *Mell, P.* The NIST Definition of Cloud Computing : tech. rep. / P. Mell, T. Grance.
5. GPU as a Service Market Size to Cross USD 90 Bn by 2032. — 2023. — URL: https://www.gminsights.com/pressrelease/gpu-as-a-service-market ; Дата обращения: 22.08.2025.
6. Российский рынок облачных инфраструктурных сервисов 2024. — 2024. — URL: https://survey.iksconsulting.ru/page59801703.html ; Дата обращения: 22.08.2025.
7. Облачные сервисы (рынок России). — 2025. — URL: https://www.tadviser.ru/index.php/%D0%A1%D1%82%D0%B0%D1%82%D1%8C%D1%8F:%D0%9E%D0%B1%D0%BB%D0%B0%D1%87%D0%BD%D1%8B%D0%B5_%D1%81%D0%B5%D1%80%D0%B2%D0%B8%D1%81%D1%8B_(%D1%80%D1%8B%D0%BD%D0%BE%D0%BA_%D0%A0%D0%BE%D1%81%D1%81%D0%B8%D0%B8) ; Дата обращения: 22.08.2025.
8. A manifesto for future generation cloud computing: Research directions for the next decade / R. Buyya [et al.] // ACM computing surveys (CSUR). — 2018. — Vol. 51, no. 5. — P. 1—38.
9. Digital 2025 Global Overview report // We Are Social and Hootsuite. — 2025. — URL: https://wearesocial.com/uk/blog/2025/02/digital-2025-the-essential-guide-to-the-global-state-of-digital/ ; Дата обращения: 22.08.2025.

10. Итоги года для российской телеком-отрасли. — 2025. — URL: https://digital.gov.ru/news/itogi-goda-dlya-rossijskoj-telekom-otrasli.

11. Сколько камер видеонаблюдения на московских улицах? — 2020. — URL: https://zen.yandex.ru/media/id/5f6500c1d70a1e7c25216f13/skolko-kamer-videonabliudeniia-na-moskovskih-ulicah-5fbbda736ea65c24b39b25d3 ; Дата обращения: 22.08.2025.

12. Российский рынок облачных услуг (IaaS/PaaS) в 2024 году вырос на 31%. — 2023. — URL: https://www.gminsights.com/pressrelease/gpu-as-a-service-market ; Дата обращения: 22.08.2025.

13. *Cisco Systems, Inc.* Cisco Annual Internet Report / Cisco Systems, Inc. — 2025. — Дата обращения: 23.08.2025. https://www.cisco.com/c/en/us/solutions/executive-perspectives/annual-internet-report/index.html.

14. Electricity 2025. — 2025. — URL: https://www.iea.org/reports/electricity-2025 ; Дата обращения: 22.08.2025.

15. Recalibrating global data center energy-use estimates / E. Masanet [et al.] // Science. — 2020. — Vol. 367, no. 6481. — P. 984—986. — URL: https://www.science.org/doi/10.1126/science.aba3758.

16. *International Energy Agency.* Electricity 2024 / International Energy Agency. — 2024. — URL: https://www.iea.org/reports/electricity-2024 ; Дата обращения: 22.08.2025.

17. *Flautner, K.* Automatic Performance Setting for Dynamic Voltage Scaling / K. Flautner, S. Reinhardt, T. Mudge // Proceedings of the 7th Annual International Conference on Mobile Computing and Networking. — 2001. — P. 260—271. — (MobiCom '01).

18. *Uptime Institute.* Global Data Center Survey 2023 / Uptime Institute. — 2023. — URL: https://uptimeinstitute.com/resources/asset/2023-data-center-industry-survey ; Дата обращения: 23.08.2024. Uptime Institute, LLC.

19. *The Green Grid.* PUE: A Comprehensive Examination of the Metric / The Green Grid. — 2012. — URL: https://www.thegreengrid.org/en/resources/library-and-tools/20-PUE%7B%5C%%7D3A-A-.

20. Chaos Engineering: A Multi-Vocal Literature Review / J. Owotogbe [et al.] // arXiv preprint arXiv:2412.01416. — 2024.

21. *Microsoft.* Microsoft Digital Defense Report 2023 / Microsoft ; Microsoft Corporation. — 2023. — URL: https://www.microsoft.com/en-us/security/security-insider/threat-landscape/microsoft-digital-defense-report-2023 ; Дата обращения: 23.08.2025.

22. *Cisco.* Cisco Data Center Infrastructure 2.5 Design Guide : tech. rep. / Cisco. — 2007. — URL: http://www.cisco.com.

23. Design Strategies and Performance Enhancement Techniques for Spine-Leaf Architecture: A Review / M. Gupta [et al.] // NEU Journal for Artificial Intelligence and Internet of Things. — 2023. — Vol. 2, no. 4.

24. *Bhardwaj, A.* Virtualization in cloud computing: Moving from hypervisor to containerization—a survey / A. Bhardwaj, C. R. Krishna // Arabian Journal for Science and Engineering. — 2021. — Vol. 46, no. 9. — P. 8585—8601.

25. *IBM.* Containers vs. VMs: What's the difference? / IBM. — 2023. — Дата обращения: 23.08.2025. https://www.ibm.com/topics/containers-vs-vms.

26. *Hong, C.-H.* GPU virtualization and scheduling methods: A comprehensive survey / C.-H. Hong, I. Spence, D. S. Nikolopoulos // ACM Computing Surveys (CSUR). — 2017. — Vol. 50, no. 3. — P. 1—37.

27. vSphere Virtual Machine Administration : Cold Migration. — VMware, Inc., 2023. — URL: https://techdocs.broadcom.com/us/en/vmware-cis/vsphere/vsphere/8-0/vsphere-virtual-machine-administration-guide-8-0.html ; Дата обращения: 23.08.2025.

28. Thermal-aware virtual machine placement based on multi-objective optimization / B. Liu [et al.] // The Journal of Supercomputing. — 2023. — Vol. 79, no. 11. — P. 12563—12590.

29. Energy-aware virtual machine placement based on a holistic thermal model for cloud data centers / J. Lin [et al.] // Future Generation Computer Systems. — 2024. — Vol. 161. — P. 302—314.

30. Live migration of virtual machines / C. Clark [et al.] // Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2. — 2005. — P. 273—286.

31. *Dias, A. H.* A systematic literature review on virtual machine consolidation / A. H. Dias, L. H. Correia, N. Malheiros // ACM Computing Surveys (CSUR). — 2021. — Vol. 54, no. 8. — P. 1—38.

32. *Saidi, K.* Task scheduling and VM placement to resource allocation in Cloud computing: challenges and opportunities / K. Saidi, D. Bardou // Cluster Computing. — 2023. — Vol. 26, no. 5. — P. 3069—3087.

33. *Тутов, А. В.* Метод и алгоритм статического размещения виртуальных машин для повышения эффективности функционирования инфокоммуникационной системы центров обработки данных / А. В. Тутов, М. П. Фархадов // Нейрокомпьютеры: разработка, применение. — 2024. — Т. 26, № 5. — С. 107—119.

34. Многокритериальная оптимизация размещения виртуальных машин по физическим серверам в облачных центрах обработки данных / А. В. Тутов [и др.] // T-Comm-Телекоммуникации и Транспорт. — 2021. — Т. 15, № 1. — С. 28—34.

35. *Baydoun, A. M.* Network-, Cost-, and Renewable-Aware Ant Colony Optimization for Energy-Efficient Virtual Machine Placement in Cloud Datacenters / A. M. Baydoun, A. S. Zekri // Future Internet. — 2025. — Vol. 17, no. 6. — P. 261. — URL: https://www.mdpi.com/1999-5903/17/6/261.

36. PACO-VMP: parallel ant colony optimization for virtual machine placement / J. Peake [et al.] // Future Generation Computer Systems. — 2022. — Vol. 129. — P. 174—186.

37. *Sunil, S.* Energy-efficient virtual machine placement algorithm based on power usage / S. Sunil, S. Patel // Computing. — 2023. — Vol. 105, no. 7. — P. 1597—1621.

38. An ant colony system for energy-efficient dynamic virtual machine placement in data centers / F. Alharbi [et al.] // Expert Systems with Applications. — 2019. — Vol. 120. — P. 228—238.

39. *Zeng, T.* An energy-efficient virtual machine placement algorithm based on improved grouping ga / T. Zeng, X. Wu // 2024 IEEE 7th International Conference on Automation, Electronics and Electrical Engineering (AUTEEE). — IEEE. 2024. — P. 737—740.

40. An energy-optimized embedded load balancing using DVFS computing in cloud data centers / A. Javadpour [et al.] // Computer Communications. — 2023. — Vol. 197. — P. 255—266.

41. *Zhang, Y.-W.* DVFS-based energy-aware scheduling of imprecise mixed-criticality real-time tasks / Y.-W. Zhang // Journal of Systems Architecture. — 2023. — Vol. 137. — P. 102849.

42. *Saxena, D.* OP-MLB: An online VM prediction-based multi-objective load balancing framework for resource management at cloud data center / D. Saxena, A. K. Singh, R. Buyya // IEEE Transactions on Cloud Computing. — 2021. — Vol. 10, no. 4. — P. 2804—2816.

43. *Nikzad, B.* Sla-aware and energy-efficient virtual machine placement and consolidation in heterogeneous DVFS enabled cloud datacenter / B. Nikzad, B. Barzegar, H. Motameni // IEEE Access. — 2022. — Vol. 10. — P. 81787—81804.

44. Hlem-vmp: An effective virtual machine placement algorithm for minimizing sla violations in cloud data centers / Y. Jing [et al.] // IEEE Systems Journal. — 2024.

45. *Karmakar, K.* An ACO-based multi-objective optimization for cooperating VM placement in cloud data center / K. Karmakar, R. K. Das, S. Khatua // The Journal of Supercomputing. — 2022. — Vol. 78, no. 3. — P. 3093—3121.

46. Multi-objective optimization for VM placement in homogeneous and heterogeneous cloud service provider data centers / R. Regaieg [et al.] // Computing. — 2021. — Vol. 103, no. 6. — P. 1255—1279.

47. *Durairaj, S.* MOM-VMP: multi-objective mayfly optimization algorithm for VM placement supported by principal component analysis (PCA) in cloud data center / S. Durairaj, R. Sridhar // Cluster Computing. — 2024. — Vol. 27, no. 2. — P. 1733—1751.

48. Towards virtual machine scheduling research based on multi-decision AHP method in the cloud computing platform / H. Gu [et al.] // PeerJ Computer Science. — 2023. — Vol. 9. — e1675.

49. *Karimunnisa, S.* An AHP based task scheduling and optimal resource allocation in cloud computing / S. Karimunnisa, Y. Pachipala // International Journal of Advanced Computer Science and Applications. — 2023. — Vol. 14, no. 3.

50. *Attaoui, W.* Multi-criteria virtual machine placement in cloud computing environments: a literature review / W. Attaoui, E. Sabir // 2024 International Conference on Ubiquitous Networking (UNet). Vol. 10. — IEEE. 2024. — P. 1—11.

51. *Siavashi, A.* GPUCloudSim: an extension of CloudSim for modeling and simulation of GPUs in cloud data centers / A. Siavashi, M. Momtazpour // The Journal of Supercomputing. — 2019. — Vol. 75, no. 5. — P. 2535—2561.

52. NVIDIA Virtual GPU (vGPU) Software. — 2025. — Accessed: 2025-07-20. https://docs.nvidia.com/vgpu.

53. *Hong, C. H.* GPU virtualization and scheduling methods: A comprehensive survey / C. H. Hong, I. Spence, D. S. Nikolopoulos // ACM Computing Surveys (CSUR). — 2017. — Vol. 50, no. 3. — P. 1—37.

54. NVIDIA Multi-Instance GPU User Guide. — 2025. — Accessed: 2025-07-20. https://docs.nvidia.com/datacenter/tesla/mig-user-guide.

55. *Siavashi, A.* A Multi-Objective Framework for Optimizing GPU-Enabled VM Placement in Cloud Data Centers with Multi-Instance GPU Technology / A. Siavashi, M. Momtazpour // arXiv preprint. — 2025. — arXiv: 2502.01909. — URL: https://arxiv.org/abs/2502.01909.

56. *Weng, Q.* Beware of fragmentation: Scheduling GPU-Sharing workloads with fragmentation gradient descent / Q. Weng [et al.] // 2023 USENIX Annual Technical Conference (USENIX ATC 23). — 2023. — P. 995—1008. — URL: https://www.usenix.org/conference/atc23/presentation/weng.

57. *Kulkarni, A. K.* GPU-aware resource management in heterogeneous cloud data centers / A. K. Kulkarni, B. Annappa // The Journal of Supercomputing. — 2021. — Vol. 77, no. 11. — P. 12458—12485.

58. *Sivaraman, H.* TECN: Task Selection and Placement in GPU Enabled Clouds Using Neural Networks / H. Sivaraman, U. Kurkure, L. Vu // 2019 International Conference on High Performance Computing & Simulation (HPCS). — IEEE, 2019. — P. 890—896.

59. Virtual Machine Placement Solution for VGPU Enabled Clouds / A. Garg [et al.] // 2019 International Conference on High Performance Computing & Simulation (HPCS). — IEEE, 2019. — P. 897—903.

60. Serving DNN Models with Multi-Instance GPUs: A Case of the Reconfigurable Machine Scheduling Problem / C. Tan [et al.] // arXiv preprint. — 2021. — arXiv: 2109.11067.

61. Miso: Exploiting Multi-Instance GPU Capability on Multi-Tenant GPU Clusters / B. Li [et al.] // Proceedings of the 13th Symposium on Cloud Computing. — 2022. — P. 173—189.

62. ParvaGPU: Efficient Spatial GPU Sharing for Large-Scale DNN Inference in Cloud Environments / M. Lee [et al.] // SC24: International Conference for High Performance Computing, Networking, Storage and Analysis. — IEEE, 2024. — P. 1—14.

63. Optimizing Hardware Resource Partitioning and Job Allocations on Modern GPUs Under Power Caps / E. Arima [et al.] // Workshop Proceedings of the 51st International Conference on Parallel Processing. — 2022. — P. 1—10.

64. *Belgacem, A.* Dynamic resource allocation in cloud computing: analysis and taxonomies / A. Belgacem // Computing. — 2022. — Vol. 104, no. 3. — P. 681—710.

65. *Murtazaev, A.* Sercon: Server consolidation algorithm using live migration of virtual machines for green computing / A. Murtazaev, S. Oh // IETE Technical Review. — 2011. — Vol. 28, no. 3. — P. 212—231.

66. *Feller, E.* A case for fully decentralized dynamic VM consolidation in clouds / E. Feller, C. Morin, A. Esnault // Cloud Computing Technology and Science (CloudCom), 2012 IEEE 4th International Conference on. — 2012. — P. 26—33.

67. Vmware distributed resource management: Design, implementation, and lessons learned / A. Gulati [et al.] // VMware Technical Journal. — 2012. — Vol. 1, no. 1. — P. 45—64.

68. *Khallouli, W.* Cluster resource scheduling in cloud computing: literature review and research challenges / W. Khallouli, J. Huang // The Journal of supercomputing. — 2022. — Vol. 78, no. 5. — P. 6898—6943.

69. Multi-factor nature inspired SLA-aware energy efficient resource management for cloud environments / S. Bashir [et al.] // Cluster Computing. — 2023. — Vol. 26, no. 2. — P. 1643—1658.

70. Runtime management of service level agreements through proactive resource provisioning for a cloud environment / S. Nadeem [et al.] // Electronics. — 2023. — Vol. 12, no. 2. — P. 296.

71. Resource allocation with service affinity in large-scale cloud environments / Z. Chen [et al.] // 2024 IEEE 40th International Conference on Data Engineering (ICDE). — IEEE. 2024. — P. 5280—5293.

72. Energy-efficient and communication-aware resource allocation in container-based cloud with group genetic algorithm / Z. Fang [et al.] // International Conference on Service-Oriented Computing. — Springer. 2023. — P. 212—226.

73. A survey on virtual machine migration: Challenges, techniques, and open issues / F. Zhang [et al.] // IEEE Communications Surveys & Tutorials. — 2018. — Vol. 20, no. 2. — P. 1206—1243.

74. Live virtual machine migration: A survey, research challenges, and future directions / M. Imran [et al.] // Computers and Electrical Engineering. — 2022. — Vol. 103. — P. 108297.

75. *Le, T.* A survey of live virtual machine migration techniques / T. Le // Computer Science Review. — 2020. — Vol. 38. — P. 100304.

76. *Masdari, M.* Efficient VM migrations using forecasting techniques in cloud computing: a comprehensive review / M. Masdari, H. Khezri // Cluster Computing. — 2020. — Vol. 23, no. 4. — P. 2629—2658.

77. *Zolfaghari, R.* Virtual machine consolidation in cloud computing systems: Challenges and future trends / R. Zolfaghari, A. M. Rahmani // Wireless Personal Communications. — 2020. — Vol. 115, no. 3. — P. 2289—2326.

78. *Imai, S.* Maximum sustainable throughput prediction for data stream processing over public clouds / S. Imai, S. Patterson, C. A. Varela // 2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID). — IEEE. 2017. — P. 504—513.

79. *Beloglazov, A.* Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers / A. Beloglazov, R. Buyya // Concurrency and Computation: Practice and Experience. — 2012. — Vol. 24, no. 13. — P. 1397—1420.

80. *Chen, J.* A proactive resource allocation method based on adaptive prediction of resource requests in cloud computing / J. Chen, Y. Wang, T. Liu // EURASIP Journal on Wireless Communications and Networking. — 2021. — Vol. 2021, no. 1. — P. 24.

81. Predictive Resource Allocation Strategies for Cloud Computing Environments Using Machine Learning. / T. Kamble [et al.] // Journal of Electrical Systems. — 2023. — Vol. 19, no. 2.

82. A resource utilization prediction model for cloud data centers using evolutionary algorithms and machine learning techniques / S. Malik [et al.] // Applied Sciences. — 2022. — Vol. 12, no. 4. — P. 2160.

83. *Devarajan, M. V.* An improved BP neural network algorithm for forecasting workload in intelligent cloud computing / M. V. Devarajan // Journal of Current Science. — 2022. — Vol. 10, no. 3. — P. 45—60.

84. Distributed hybrid cpu and gpu training for graph neural networks on billion-scale heterogeneous graphs / D. Zheng [et al.] // Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. — 2022. — P. 4582—4591.

85. Characterization and prediction of deep learning workloads in large-scale gpu datacenters / Q. Hu [et al.] // Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. — 2021. — P. 1—15.

86. *Xu, J.* Multi-objective virtual machine placement in virtualized data center environments / J. Xu, J. Fortes // Green Computing and Communications (GreenCom), 2010 IEEE/ACM Int'l Conference on & Int'l Conference on Cyber, Physical and Social Computing (CPSCom). — 2010. — P. 179—188.

87. Standard Performance Evaluation Corporation. — URL: http://www.spec.org/power_ssj2008/results/power_ssj2008.html.

88. *Ворожцов, А. С.* Динамическое распределение вычислительных ресурсов центров обработки данных / А. С. Ворожцов, Н. В. Тутова, А. В. Тутов // T-Comm–Телекоммуникации и транспорт. — 2016. — Т. 10, № 7. — С. 47—51.

89. *Тутов, А. В.* Метод и алгоритм статического размещения виртуальных машин для повышения эффективности функционирования инфокоммуникационной системы центров обработки данных / А. В. Тутов, М. П. Фархадов // Нейрокомпьютеры: разработка, применение. — 2024. — Т. 18, № 5. — С. 107—119.

90. Heuristics for vector bin packing / R. Panigrahy [et al.] // research. microsoft. com. — 2011.

91. *Verma, A.* pMapper: power and migration cost aware application placement in virtualized systems / A. Verma, P. Ahuja, A. Neogi // Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware. — 2008. — P. 243—264.

92. G. Jung [et al.] // 2008 International Conference on Autonomic Computing. — 2008. — P. 23—32.

93. A two stage heuristic algorithm for solving the server consolidation problem with item-item and bin-item incompatibility constraints / R. Gupta [et al.] // 2008 IEEE International Conference on Services Computing. — 2008. — Vol. 2. — P. 39—46.

94. A systematic survey on energy-efficient techniques in sustainable cloud computing / S. Bharany [et al.] // Sustainability. — 2022. — Vol. 14, no. 10. — P. 6256.

95. *Wu, X.* A ga-based energy aware virtual machine placement algorithm for cloud data centers / X. Wu // 2021 12th International Symposium on Parallel Architectures, Algorithms and Programming (PAAP). — IEEE. 2021. — P. 42—46.

96. Energy-efficient virtual machine placement in data centers by ant colony optimization algorithm (ACO) / S. I. Suliman [et al.] // 2024 IEEE 6th Symposium on Computers & Informatics (ISCI). — IEEE. 2024. — P. 146—151.

97. *Dorigo, M.* Ant colony optimization / M. Dorigo, M. Birattari, T. Stutzle // IEEE computational intelligence magazine. — 2007. — Vol. 1, no. 4. — P. 28—39.

98. *Huebscher, M. C.* A survey of autonomic computing—degrees, models, and applications / M. C. Huebscher, J. A. McCann // ACM Computing Surveys (CSUR). — 2008. — Vol. 40, no. 3. — P. 1—28.

99. Программно-аппаратный комплекс распределенного планировщика ресурсов инфокоммуникационной системы облачного центра обработки данных / А. В. Тутов [и др.] // Управление большими системами. — 2024. — Т. 109. — С. 268—292.

100. *Singh, J.* A comprehensive review of cloud computing virtual machine consolidation / J. Singh, N. K. Walia // IEEE Access. — 2023. — Vol. 11. — P. 106190—106209.

101. Dynamic power management for nonstationary service requests / E.-Y. Chung [et al.] // Computers, IEEE Transactions on. — 2002. — Dec. — Vol. 51. — P. 1345—1361.

102. *Шелухин, О.* Сетевые аномалии. Обнаружение, локализация, прогнозирование / О. Шелухин. — М.: Научно-техническое издательство «Горячая линия – Телеком», 2019. — С. 448.

103. *Luiz, S. O.* Multisize sliding window in workload estimation for dynamic power management / S. O. Luiz, A. Perkusich, A. M. Lima Sr // IEEE Transactions on Computers. — 2010. — Vol. 59, no. 12. — P. 1625—1639.

104. An adaptive heuristic for managing energy consumption and overloaded hosts in a cloud data center / R. Yadav [et al.] // Wireless Networks. — 2020. — Vol. 26, no. 3. — P. 1905—1919.

105. *S. Cleveland, W.* Robust Locally Weighted Regression and Smoothing Scatterplots / W. S. Cleveland // Journal of the American Statistical Association. — 1979. — Dec. — Vol. 74. — P. 829—836.

106. *Светуньков, И. С.* Методы социально-экономического прогнозирования : учебник и практикум для вузов / И. С. Светуньков, С. Г. Светуньков. — М.:Издательство Юрайт, 2025. — С. 651. — URL: https://urait.ru/bcode/556612.

107. *Chehelgerdi-Samani, M.* PCVM. ARIMA: predictive consolidation of virtual machines applying ARIMA method / M. Chehelgerdi-Samani, F. Safi-Esfahani // The Journal of Supercomputing. — 2021. — Vol. 77, no. 3. — P. 2172—2206.

108. *Yadav, M. P.* Workload prediction over cloud server using time series data / M. P. Yadav, N. Pal, D. K. Yadav // 2021 11th international conference on cloud computing, data science & engineering (confluence). — IEEE. 2021. — P. 267—272.

109. *Awad, M.* Utilization prediction-based VM consolidation approach / M. Awad, N. Kara, A. Leivadeas // Journal of Parallel and Distributed Computing. — 2022. — Vol. 170. — P. 24—38.

110. *Naveen Kumar, M.* Efficient Kalman filter based deep learning approaches for workload prediction in cloud and edge environments / M. Naveen Kumar, B. Annappa, V. Yadav // Computing. — 2025. — Vol. 107, no. 1. — P. 10.

111. Developing machine learning and deep learning models for host overload detection in cloud data center / M. S. Ricardo [et al.] // 2021 IEEE 12th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON). — IEEE. 2021. — P. 0619—0626.

112. *Ивахненко, А.* Помехоустойчивость моделирования / А. Ивахненко, В. Степашко. — Киев : Наук. Думка, 1985. — 216 с.

113. *Степашко, В. С.* Обобщенный итерационный алгоритм метода группового учета аргументов / В. С. Степашко, А. С. Булгакова // Управляющие системы и машины. — 2013.

114. *Ивахненко, А.* Моделирование сложных систем по экспериментальным данным / А. Ивахненко, Ю. Юрачковский. — М.: Радио и связь, 1987. — С. 120.

115. Alibaba cluster trace program. — 2023. — URL: https://github.com/alibaba/clusterdata/tree/%20master/cluster-trace-gpu-v2023 ; Дата обращения: 09.09.2025.

116. *Тутов, А.* ПРОГРАММА ДЛЯ ПРОГНОЗИРОВАНИЯ ПЕРЕГРУЗКИ СЕРВЕРОВ С ИСПОЛЬЗОВАНИЕМ КОМБИНАТОРНОГО МЕТОДА ГРУППОВОГО УЧЕТА АРГУМЕНТОВ НА ЯЗЫКЕ ПРОГРАММИРОВАНИЯ JAVA : свидетельство о регистрации программы для ЭВМ RUS 2018666780 07.12.2018 / А. Тутов, Н. Тутова, А. Ворожцов. — 2018.

117. Optimizing the migration of virtual computers / C. P. Sapuntzakis [et al.] // ACM SIGOPS Operating Systems Review. — 2002. — Vol. 36, SI. — P. 377—390.

118. Xen live migration with slowdown scheduling algorithm / Z. Liu [et al.] // 2010 International Conference on Parallel and Distributed Computing, Applications and Technologies. — 2010. — P. 215—221.

119. Evaluation of delta compression techniques for efficient live migration of large virtual machines / P. Svärd [et al.] // ACM Sigplan Notices. — 2011. — Vol. 46, no. 7. — P. 111—120.

120. Live virtual machine migration with adaptive, memory compression / H. Jin [et al.] // 2009 IEEE International Conference on Cluster Computing and Workshops. — 2009. — P. 1—10.

121. *Sahni, S.* A hybrid approach to live migration of virtual machines / S. Sahni, V. Varma // 2012 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM). — 2012. — P. 1—5.

122. A novel hybrid-copy algorithm for live migration of virtual machine / Z. Lei [et al.] // Future Internet. — 2017. — Vol. 9, no. 3. — P. 37.

123. *Алексанков, С. М.* Модель динамической миграции виртуальных машин с гибридным подходом / С. М. Алексанков // Научно-технический вестник информационных технологий, механики и оптики. — 2017. — Т. 17, № 4. — С. 725—732.

124. *Vorozhtsov, A. S.* Resource control system stability of mobile data centers / A. S. Vorozhtsov, N. V. Toutova, A. V. Toutov // 2018 Systems of Signals Generating and Processing in the Field of on Board Communications. — 2018. — С. 1—4.

125. Минимизация времени простоя процессов при их миграции в облачном хостинге / П. О. Тихомиров [и др.] // Вестник НГУ. Серия: Информационные технологии. — 2014. — Т. 12, № 4. — С. 112—120.

126. *Aldhalaan, A.* Analytic performance modeling and optimization of live VM migration / A. Aldhalaan, D. A. Menascé // European Workshop on Performance Engineering. — 2013. — P. 28—42.

127. *Jo, C.* A Machine Learning Approach to Live Migration Modeling / C. Jo, Y. Cho, B. Egger // Proceedings of the 2017 Symposium on Cloud Computing. — 2017. — P. 351—364.

128. Predicting the performance of virtual machine migration / S. Akoush [et al.] // 2010 IEEE international symposium on modeling, analysis and simulation of computer and telecommunication systems. — 2010. — P. 37—46.

129. *Тихонов, В. И.* Статистическая радиотехника / В. И. Тихонов. — Советское радио, 1966. — 678 с.

130. *Пугачев, В. С.* Теория вероятностей и математическая статистика : учебник. Т. 2 / В. С. Пугачев. — Москва : Транспортная компания, 2017. — 496 с.

131. *Тутов, А. В.* Метод оценки характеристик процесса миграции виртуальных машин с учетом типа миграций и приложений / А. В. Тутов, А. В. Таратухин, С. С. Керимов // Динамика сложных систем - XXI век. — 2024. — Т. 18, № 3. — С. 48—59.

132. *Toutov, A.* Analytical Approach to Estimating Total Migration Time of Virtual Machines With Various Applications / A. Toutov, A. Vorozhtsov, N. Toutova // International Journal of Embedded and Real-Time Communication Systems (IJERTCS). — 2020. — Т. 11, № 2. — С. 58—75.

133. *Тутов, А. В.* Метод первоначального размещения виртуальных машин с графическим процессором в гетерогенных центрах обработки данных / А. В. Тутов // Нейрокомпьютеры: разработка, применение. — 2025. — № 5. — С. 3—23.

134. *Camati, R. S.* Solving the virtual machine placement problem as a multiple multidimensional knapsack problem / R. S. Camati, A. Calsavara, L. Lima Jr // ICN 2014. — 2014. — P. 264.

135. Virtual machine consolidation in cloud data centers using ACO metaheuristic / M. H. Ferdaus [et al.] // European conference on parallel processing. — Springer. 2014. — P. 306—317.

136. *Moges, F. F.* Energy-aware VM placement algorithms for the OpenStack Neat consolidation framework / F. F. Moges, S. L. Abebe // Journal of Cloud Computing. — 2019. — Vol. 8, no. 1. — P. 2.

137. *Beloglazov, A.* Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing / A. Beloglazov, J. Abawajy, R. Buyya // Future generation computer systems. — 2012. — Vol. 28, no. 5. — P. 755—768.

138. *Alhammadi, A. S. A.* Multi-Objective Algorithms for Virtual Machine Selection and Placement in Cloud Data Center / A. S. A. Alhammadi, V. Vasanthi // 2021 International Congress of Advanced Technology and Engineering (ICOTEN). — IEEE. 2021. — P. 1—7.

139. *Feller, E.* Energy-aware ant colony based workload placement in clouds / E. Feller, L. Rilling, C. Morin // 2011 IEEE/ACM 12th International Conference on Grid Computing. — IEEE. 2011. — P. 26—33.

140. A multi-objective ant colony system algorithm for virtual machine placement in cloud computing / Y. Gao [et al.] // Journal of computer and system sciences. — 2013. — Vol. 79, no. 8. — P. 1230—1242.

141. A Secure and Multiobjective Virtual Machine Placement Framework for Cloud Data Center / D. Saxena [et al.] // IEEE Systems Journal. — 2021.

142. Optimal machine placement based on improved genetic algorithm in cloud computing / J. Lu [et al.] // The Journal of Supercomputing. — 2021. — P. 1—29.

143. *Xu, J.* A multi-objective approach to virtual machine management in datacenters / J. Xu, J. Fortes // Proceedings of the 8th ACM international conference on Autonomic computing. — 2011. — P. 225—234.

144. *Vorozhtsov, A.* Resource control system stability of mobile data centers / A. Vorozhtsov, N. Toutova, A. Toutov // 2018 Systems of Signals Generating and Processing in the Field of on Board Communications. — IEEE. 2018. — P. 1—4.

145. *Toutov, A. V.* Estimation of Total Migration Time of Virtual Machines in Cloud Data Centers / A. V. Toutov, A. S. Vorozhtsov, N. V. Toutova // 2018 IEEE International Conference" Quality Management, Transport and Information Security, Information Technologies"(IT&QM&IS). — IEEE. 2018. — P. 389—393.

146. *Tutov, A. V.* The Method of Calculation the Total Migration Time of Virtual Machines in Cloud Data Centers / A. V. Tutov, A. S. Vorozhtsov, N. V. Tutova // Proceedings of the 24th Conference of Open Innovations Association FRUCT. — FRUCT Oy. 2019. — P. 110.

147. *Toutov, A.* Analytical Approach to Estimating Total Migration Time of Virtual Machines With Various Applications / A. Toutov, A. Vorozhtsov, N. Toutova // International Journal of Embedded and Real-Time Communication Systems (IJERTCS). — 2020. — Vol. 11, no. 2. — P. 58—75.

148. *Ворожцов, А. С.* Оптимизация размещения облачных серверов в центрах обработки данных / А. С. Ворожцов, Н. В. Тутова, А. В. Тутов // T-Comm: Телекоммуникации и транспорт. — 2015. — Т. 9, № 6. — С. 4—8.

149. *Shaw, R.* An energy efficient anti-correlated virtual machine placement algorithm using resource usage predictions / R. Shaw, E. Howley, E. Barrett // Simulation Modelling Practice and Theory. — 2019. — Vol. 93. — P. 322—342.

150. *Farzai, S.* Multi-objective communication-aware optimization for virtual machine placement in cloud datacenters / S. Farzai, M. H. Shirvani, M. Rabbani // Sustainable Computing: Informatics and Systems. — 2020. — P. 100374.

151. An algorithm for network and data-aware placement of multi-tier applications in cloud data centers / M. H. Ferdaus [et al.] // Journal of Network and Computer Applications. — 2017. — Vol. 98. — P. 65—83.

152. Multicriteria Optimization of Virtual Machine Placement in Cloud Data Centers / A. Toutov [et al.] // 2021 28th Conference of Open Innovations Association (FRUCT). — IEEE. 2021. — P. 482—487.

153. *Ногин, В. Д.* Линейная свертка критериев в многокритериальной оптимизации / В. Д. Ногин // Искусственный интеллект и принятие решений. — 2014. — № 4. — С. 73—82.

154. *Kuhn, H. W.* The Hungarian method for the assignment problem / H. W. Kuhn // Naval research logistics quarterly. — 1955. — Vol. 2, no. 1/2. — P. 83—97.

155. *Munkres, J.* Algorithms for the assignment and transportation problems / J. Munkres // Journal of the society for industrial and applied mathematics. — 1957. — Vol. 5, no. 1. — P. 32—38.

156. *Кротов, В. Ф.* Основы теории оптимального управления: учебное пособие / В. Ф. Кротов. — Высшая школа, 1990.

157. *Данциг, Д.* Линейное программирование, его обобщение и приложение / Д. Данциг // Москва: Прогресс. — 1966. — Т. 9.

158. *Канторович, Л.* Применение математических методов в вопросах анализа грузопотоков / Л. Канторович, М. Гавурин. — 1949.

159. *Александров, А.* Применение электроники вычислительных машин в оперативном планировании / А. Александров, А. Лурье, Ю. Олейник // Автомобильный транспорт. — 1959. — № 6. — Э1.

160. CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms / R. N. Calheiros [et al.] // Software Practice and Experience. — 2011. — Jan. — Vol. 41. — P. 23—50.

161. *Gustedt, J.* Experimental Methodologies for Large-Scale Systems: a Survey / J. Gustedt, E. Jeannot, M. Quinson // Parallel Processing Letters. — 2009. — Sept. — Vol. 19.

162. *Delamare, S.* Kwollect: Metrics collection for experiments at scale / S. Delamare, L. Nussbaum // IEEE INFOCOM 2021-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS). — IEEE. 2021. — P. 1—6.

163. PlanetLab: An Overlay Testbed for Broad-coverage Services / B. Chun [et al.] // SIGCOMM Comput. Commun. Rev. — New York, NY, USA, 2003. — July. — Vol. 33, no. 3. — P. 3—12. — URL: http://doi.acm.org/10.1145/956993.956995.

164. Edgenet: the global kubernetes cluster testbed / B. C. Şenel [et al.] // IEEE INFOCOM 2021-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS). — IEEE. 2021. — P. 1—2.

165. *Soares, J. A.* Evaluating SimGrid and CloudSim Plus for Hybrid Cloud Scientific Workflows / J. A. Soares, R. B. Amaral, G. Cavalheiro // Escola Regional de Redes de Computadores (ERRC). — SBC. 2024. — P. 25—29.

166. CloudSim: A Novel Framework for Modeling and Simulation of Cloud Computing Infrastructures and Services / R. N. Calheiros [et al.]. — 2009. — Apr.

167. *Sukhoroslov, O.* Fast and Flexible Framework for Simulation of Distributed Systems / O. Sukhoroslov, A. Makogon // Russian Supercomputing Days. — Springer, 2024. — P. 90—106.

168. *Ворожцов, А. С.* Методика использование пакета программ CloudSim в учебном процессе / А. С. Ворожцов, Н. В. Тутова // Методические вопросы преподавания инфокоммуникаций в высшей школе. — 2018. — Т. 6, № 1. — С. 13—15.

169. *PlanetLab.* PlanetLab: открытая платформа для разработки, развертывания и доступа к услугам планетарного масштаба / PlanetLab. — URL: https://planet-lab.org/.

170. SPECpower benchmark. —. — URL: http://www.spec.org/power_ssj2008/.

Список рисунков

Список таблиц

Приложение А

Приложение к 3-ей главе

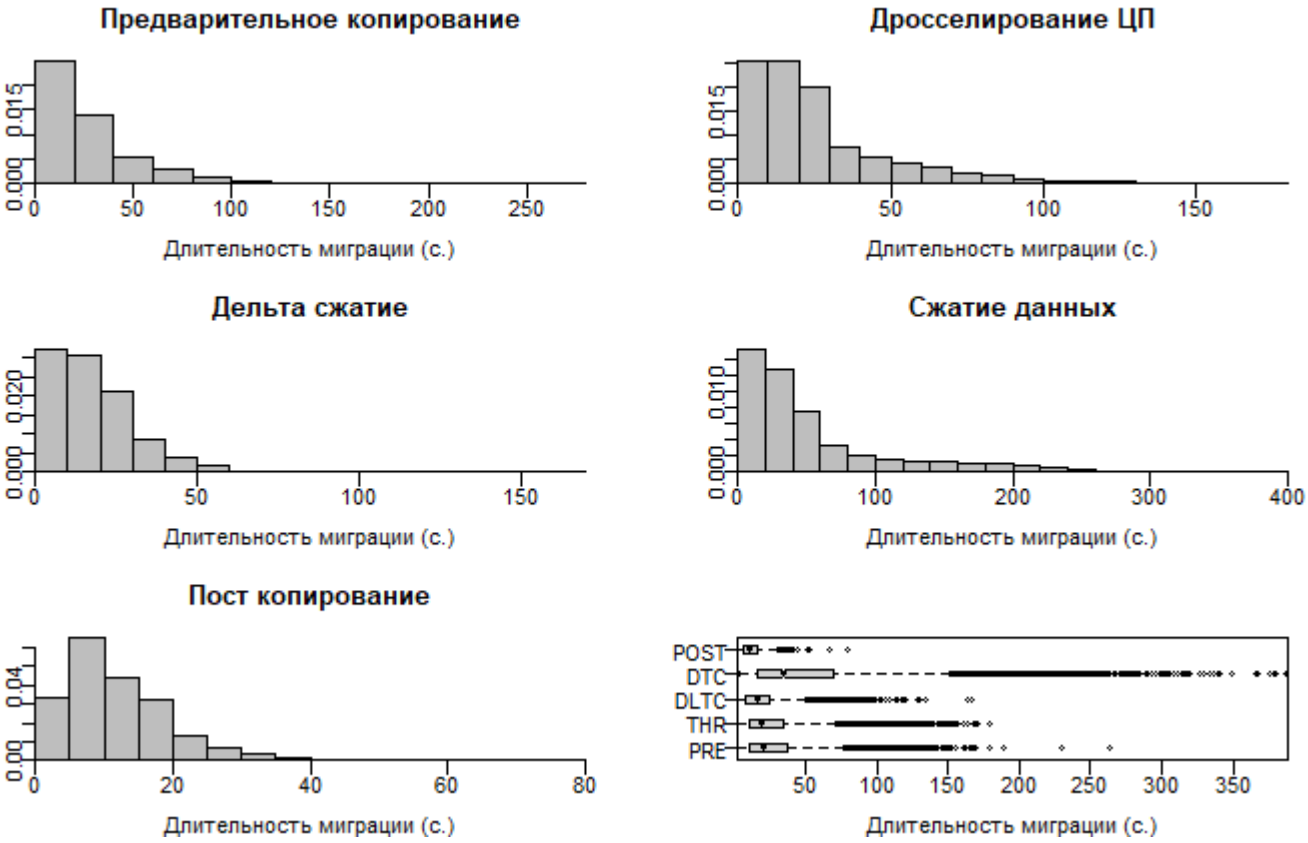

Рисунок А.1 — Распределение общего времени миграции по видам миграции

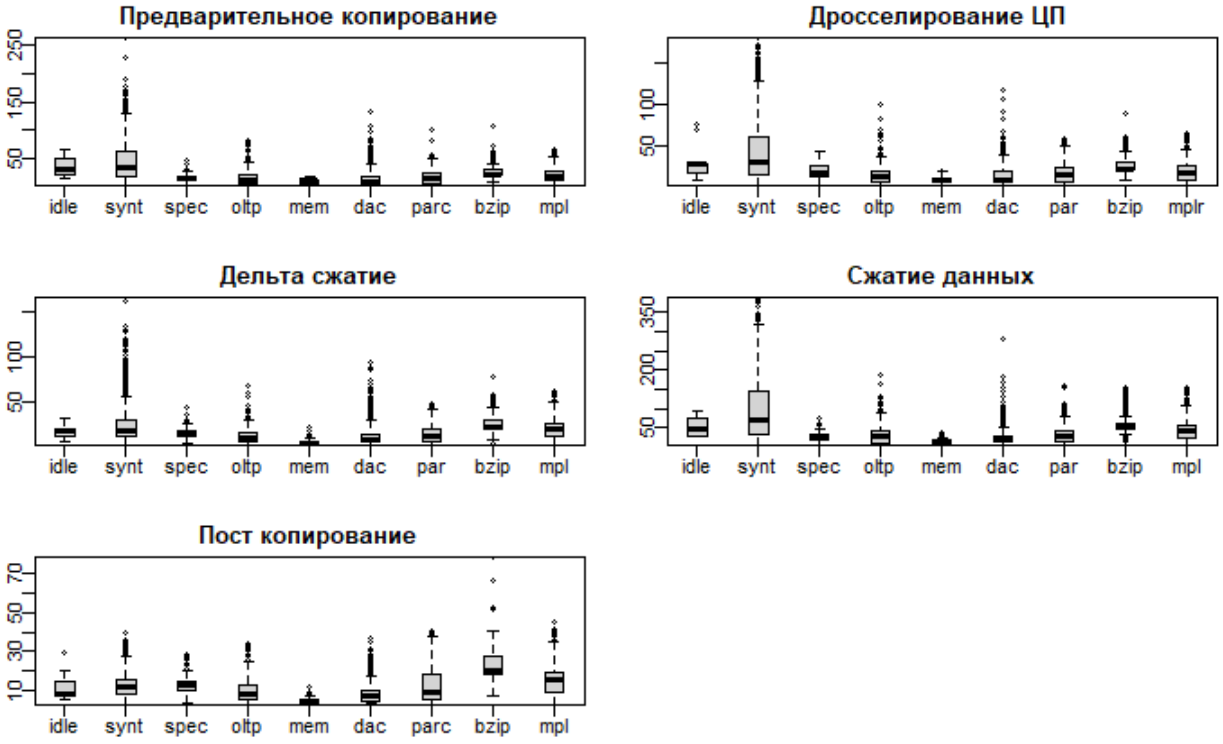

Рисунок А.2 — Распределение длительности миграции различных видов с учетом приложений

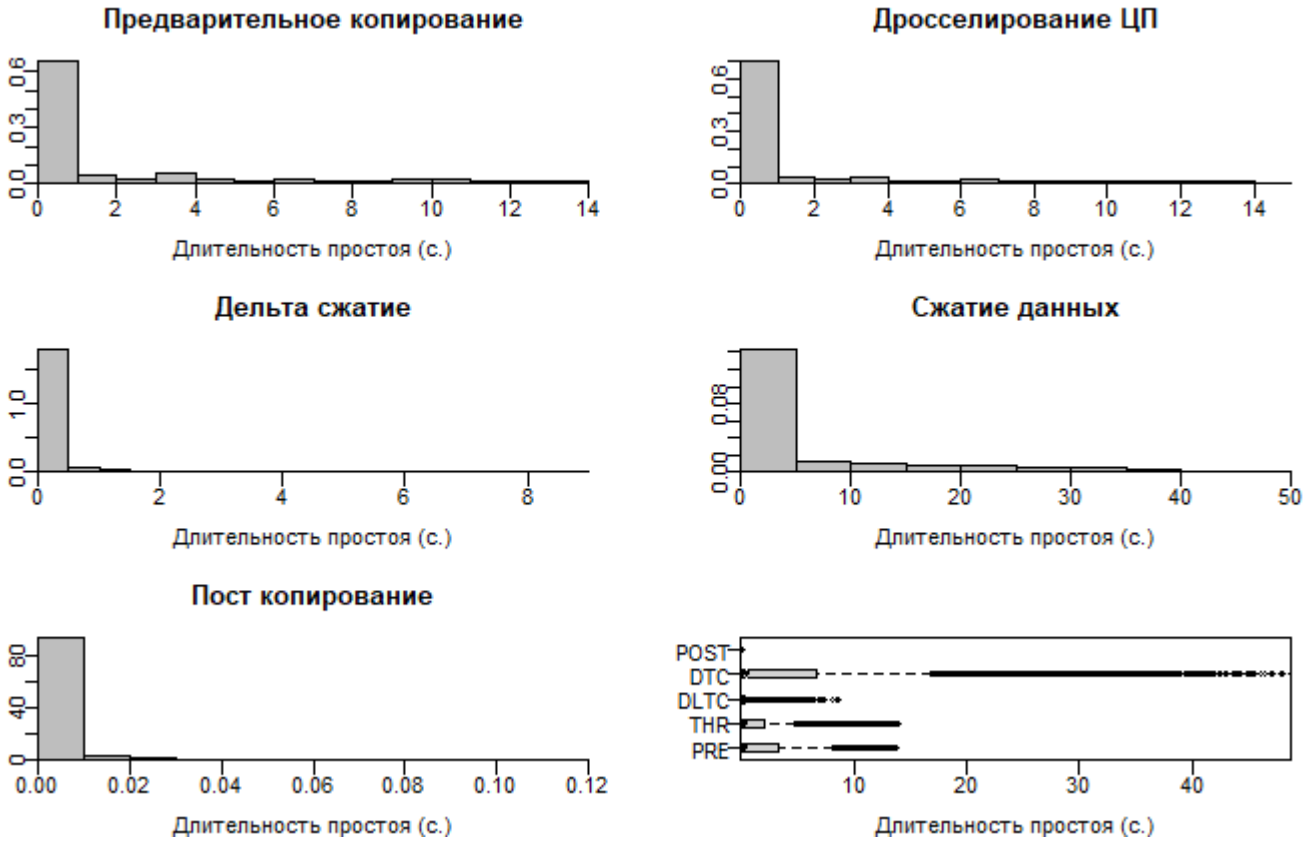

Рисунок А.3 — Распределение длительности простоя по видам миграции

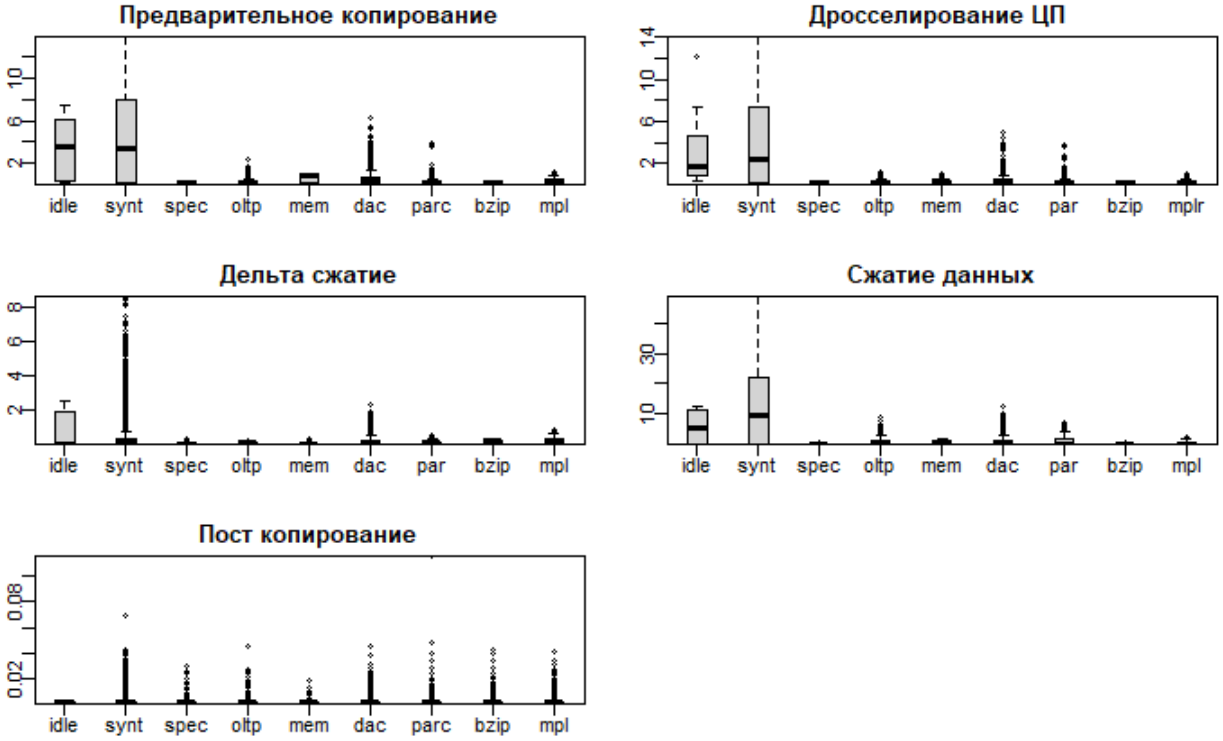

Рисунок А.4 — Распределение длительности простоя различных видов с учетом приложений

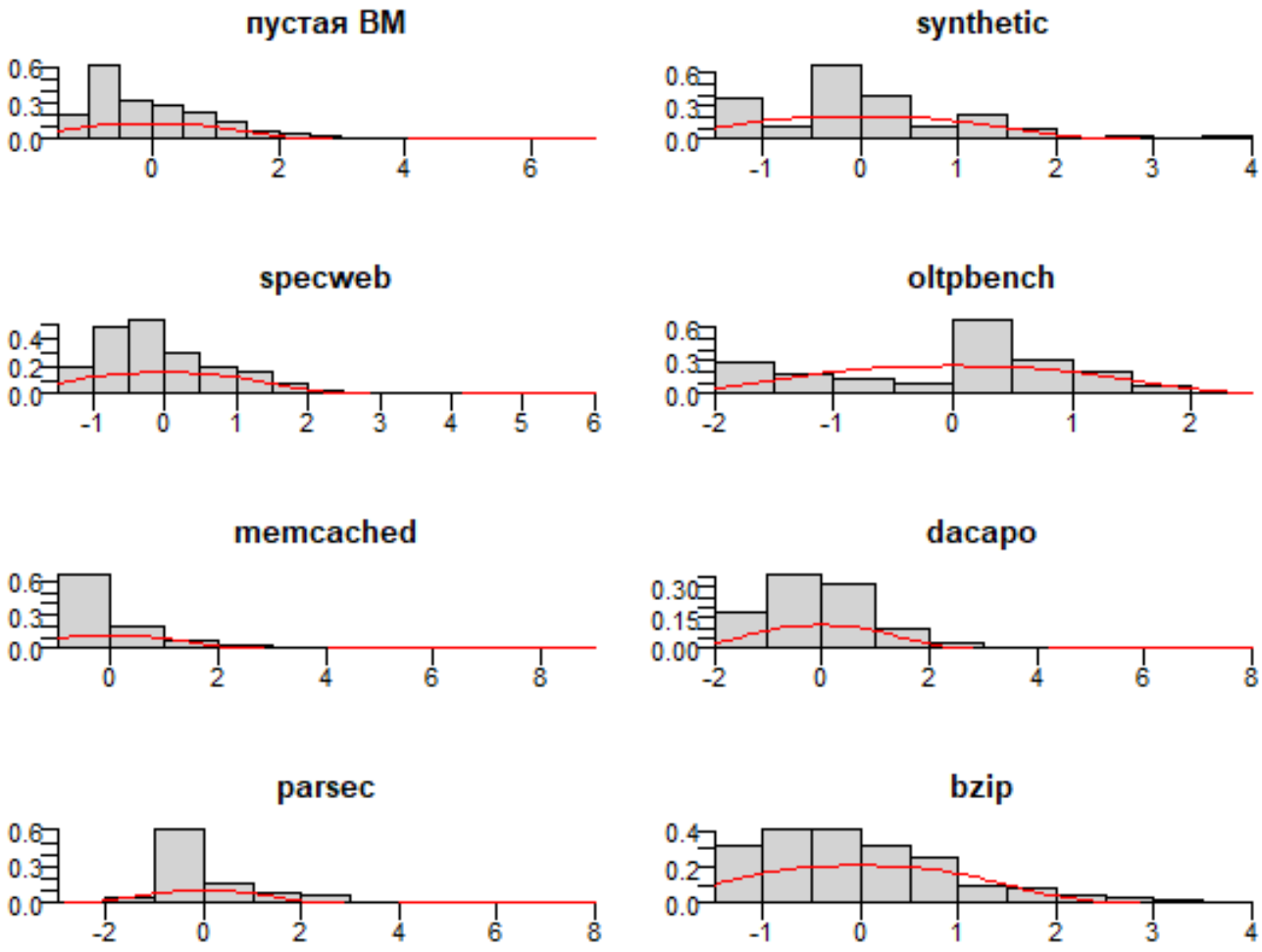

Рисунок А.5 — Аппроксимация длительности миграции ВМ рядом Грама-Шарлье для миграции с предварительным копированием

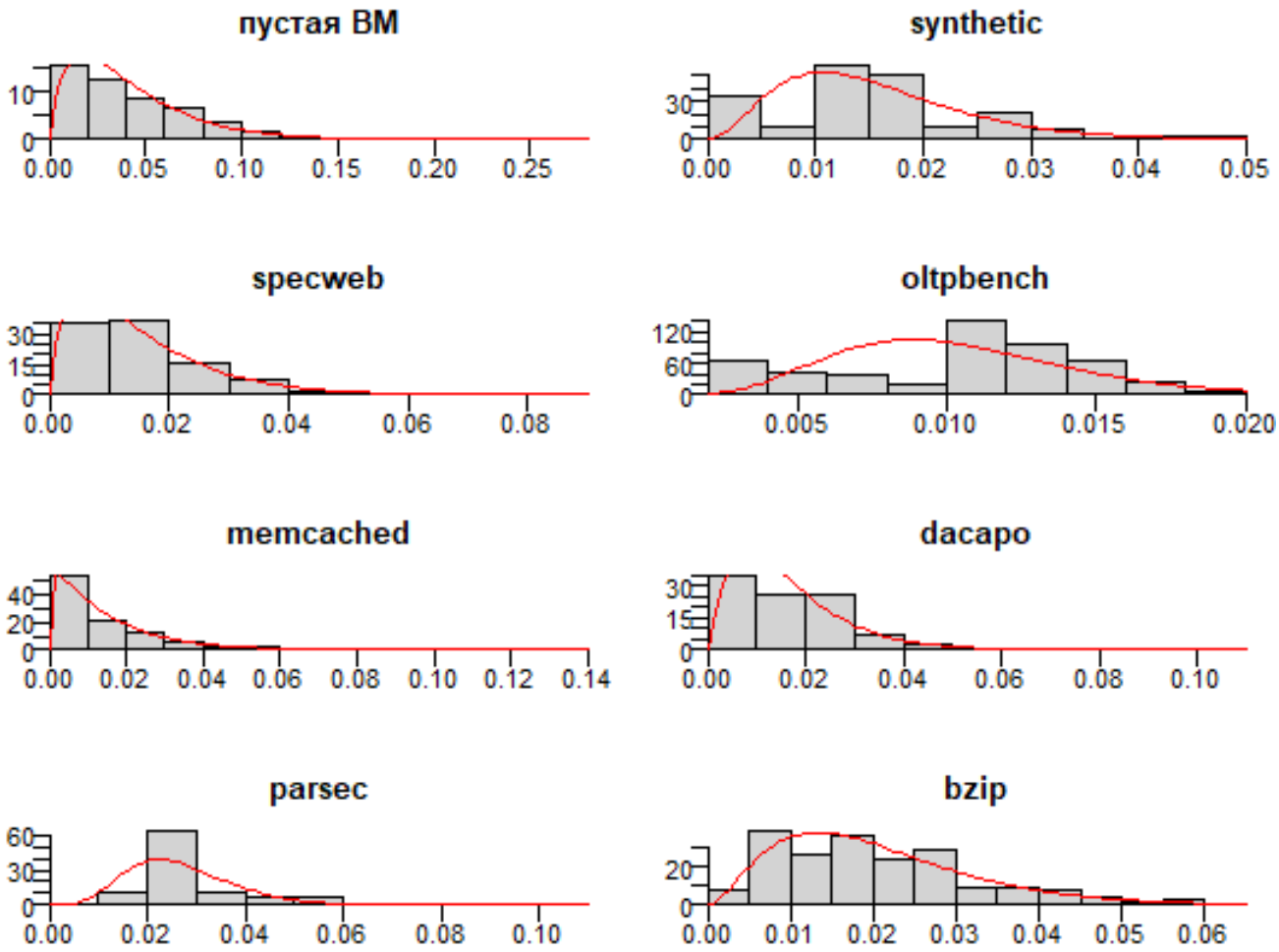

Рисунок А.6 — Аппроксимация длительности миграции ВМ рядом Лагерра для миграции с предварительным копированием

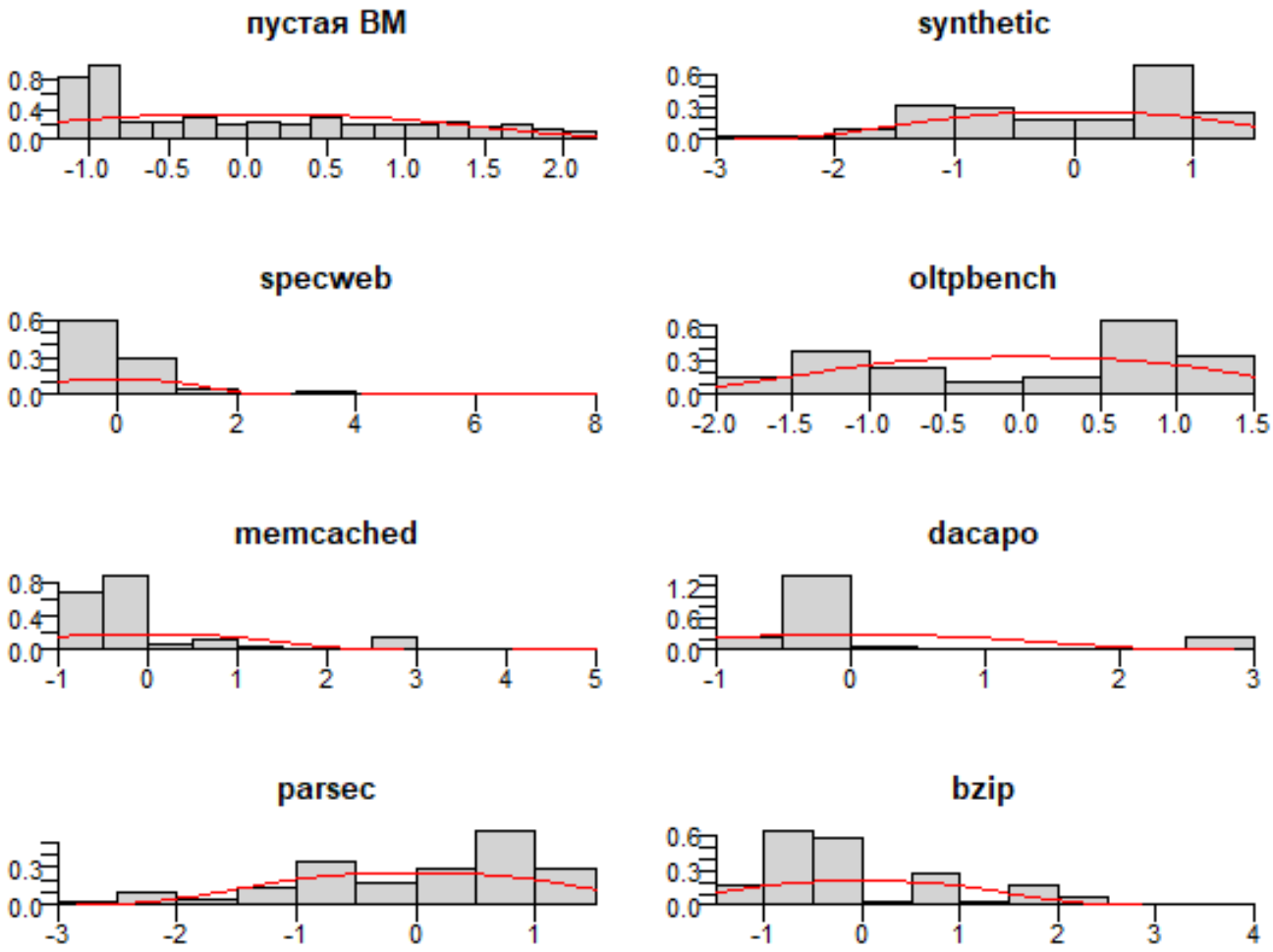

Рисунок А.7 — Аппроксимация длительности простоя ВМ рядом Грама-Шарлье для миграции с предварительным копированием

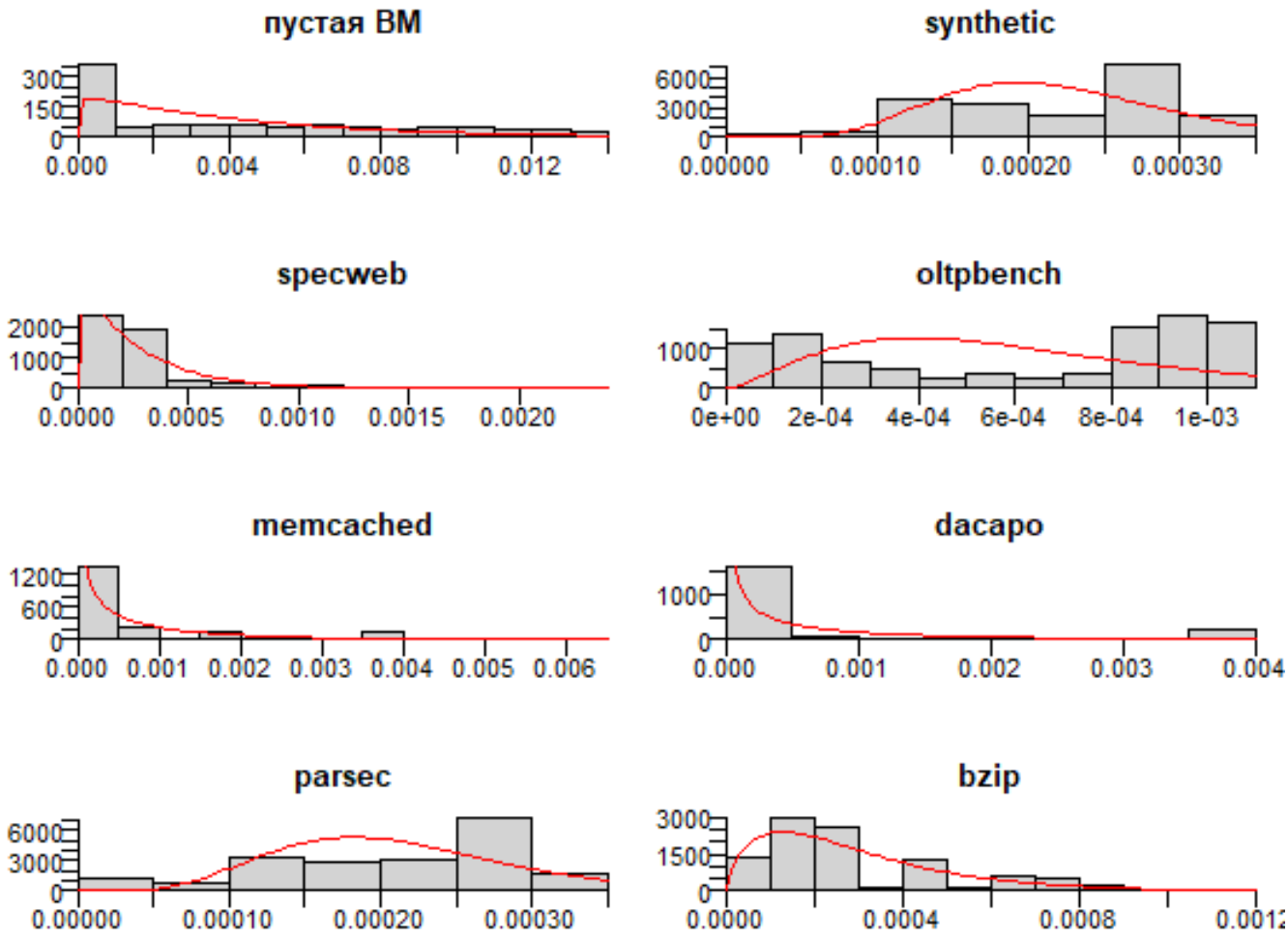

Рисунок А.8 — Аппроксимация длительности простоя ВМ рядом Лагерра для миграции с предварительным копированием

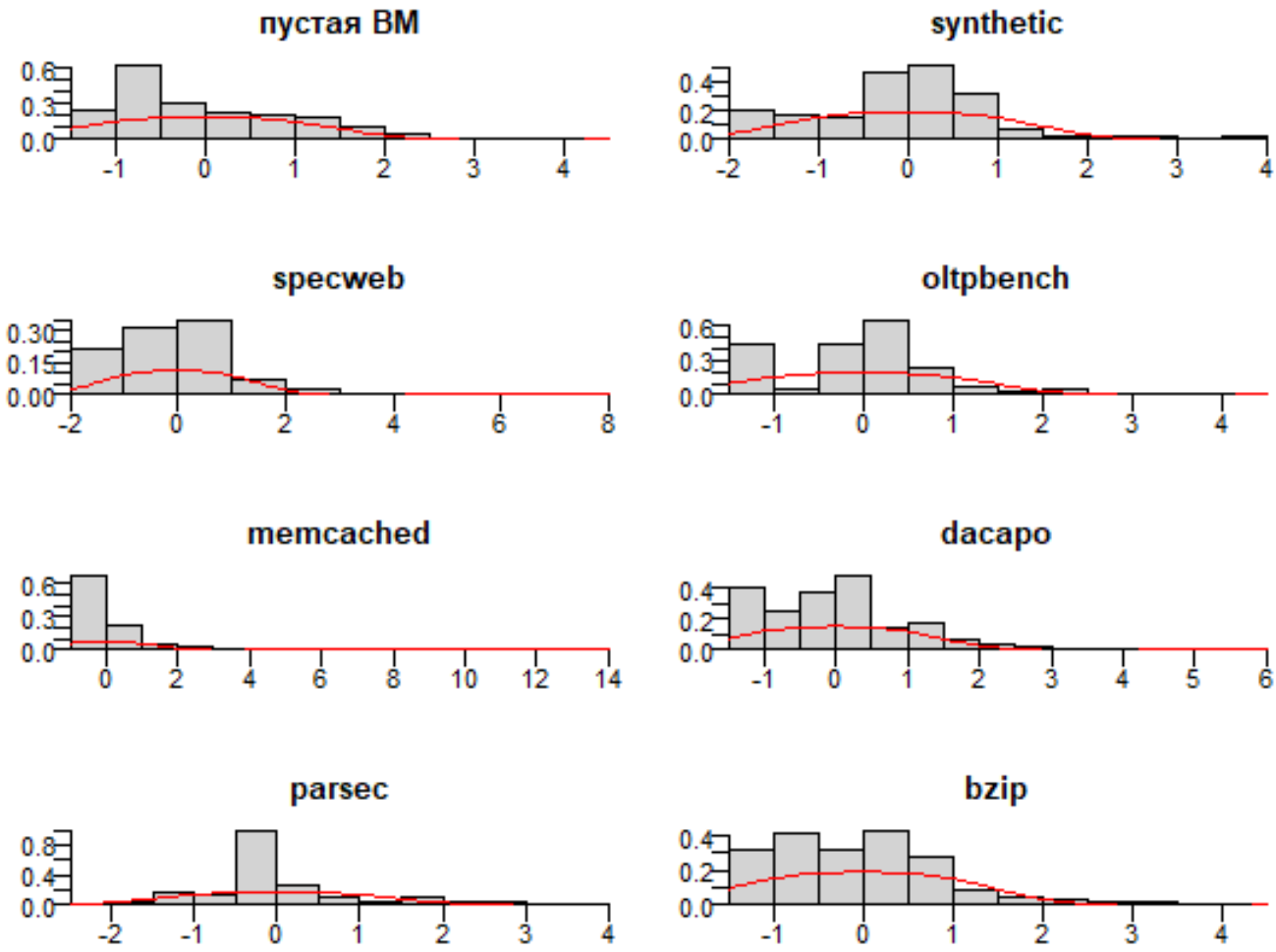

Рисунок А.9 — Аппроксимация длительности миграции ВМ рядом Грама-Шарлье для миграции с сжатием данных

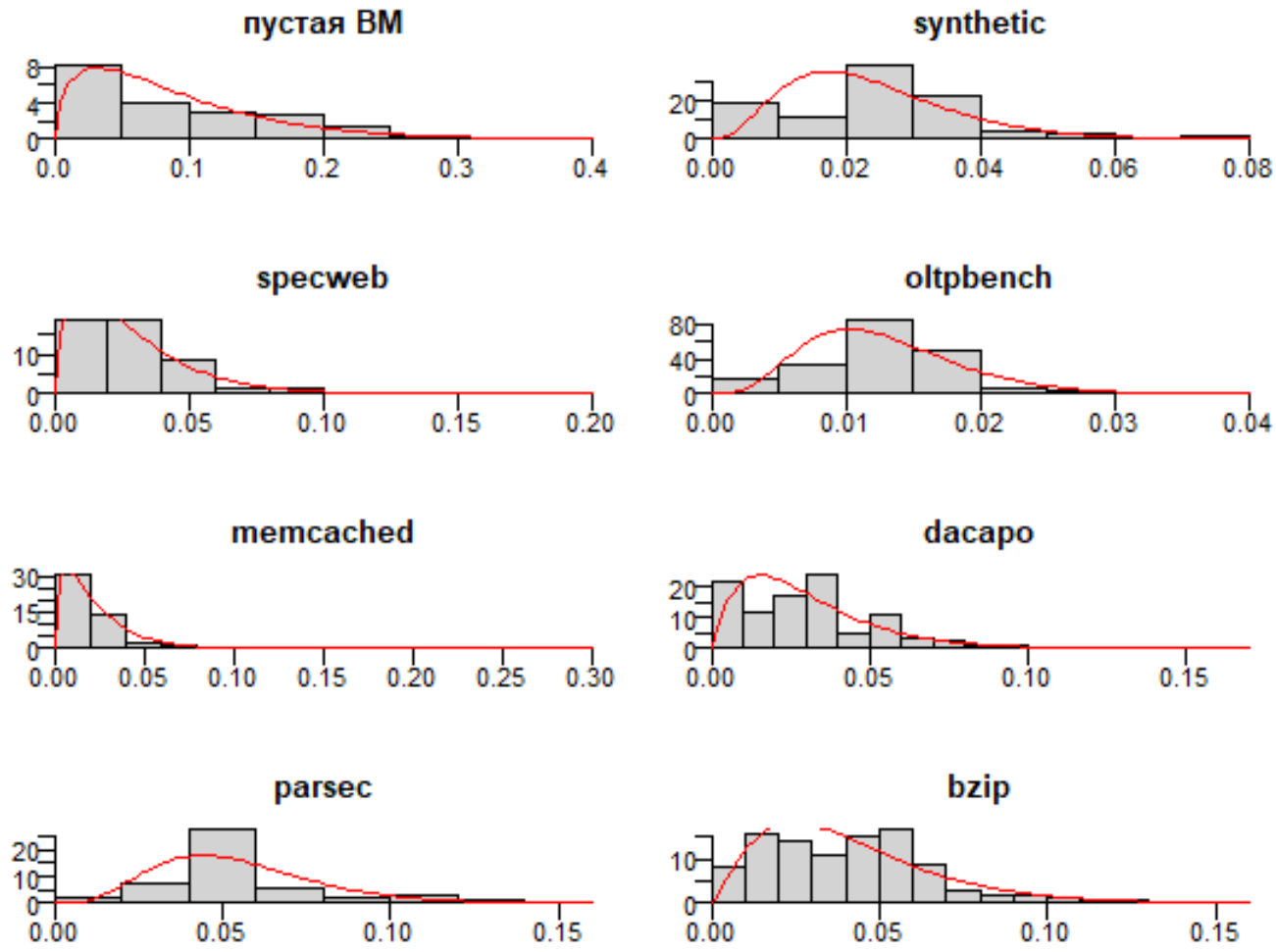

Рисунок А.10 — Аппроксимация длительности миграции ВМ рядом Лагерра для миграции с сжатием данных

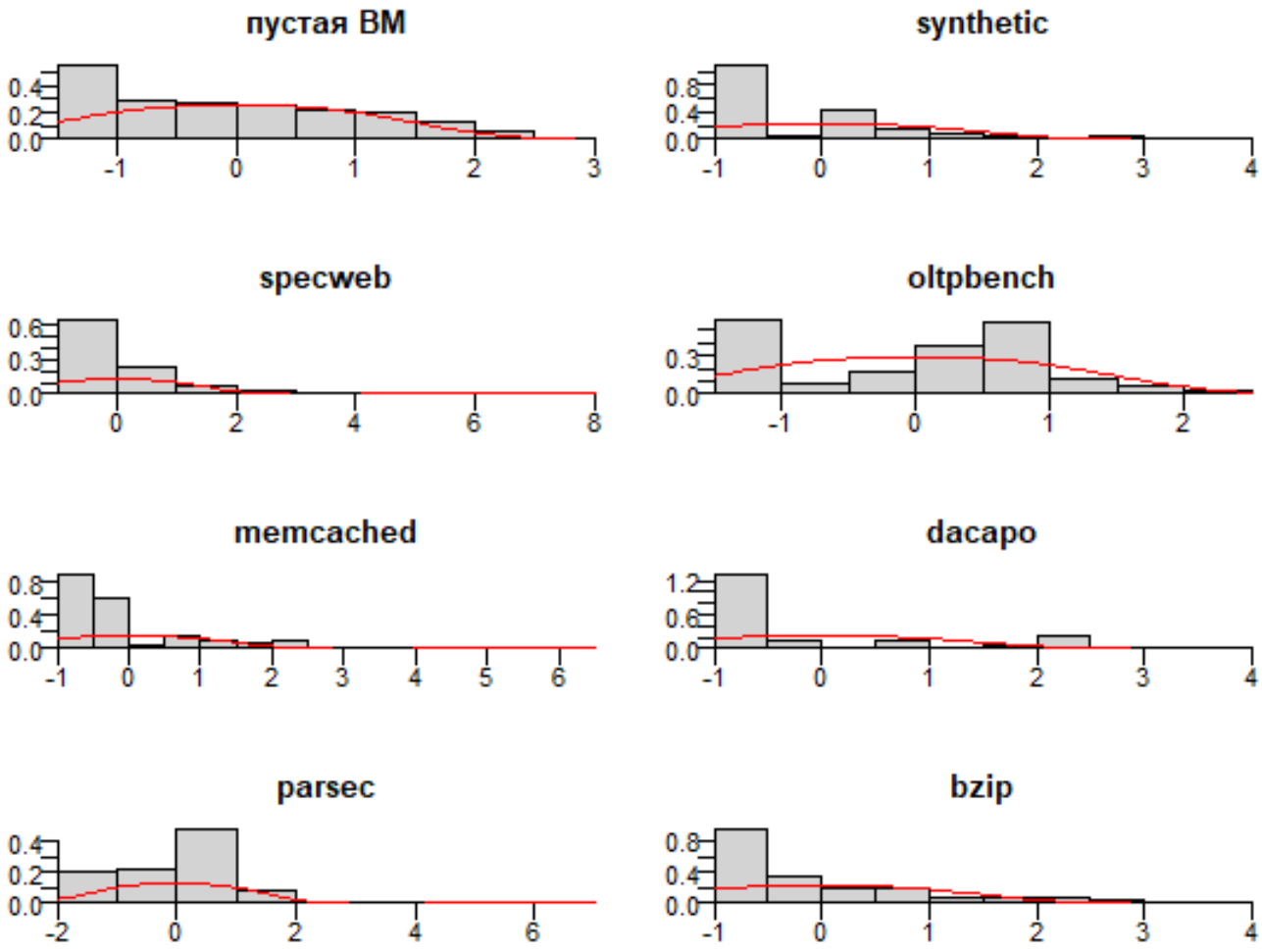

Рисунок А.11 — Аппроксимация длительности простоя ВМ рядом Грама-Шарлье для миграции с сжатием данных

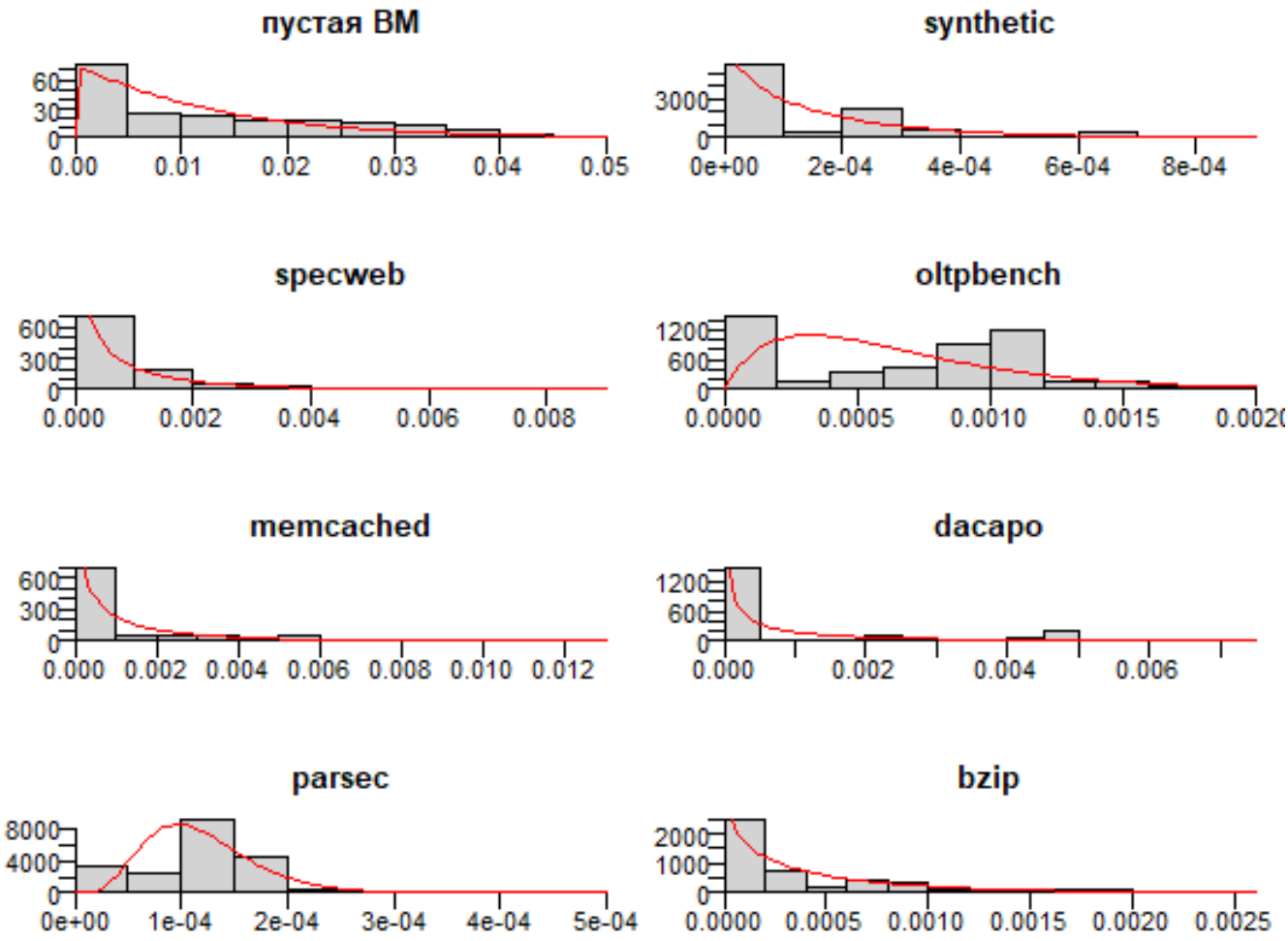

Рисунок А.12 — Аппроксимация длительности миграции ВМ рядом Лагерра для миграции с сжатием данных

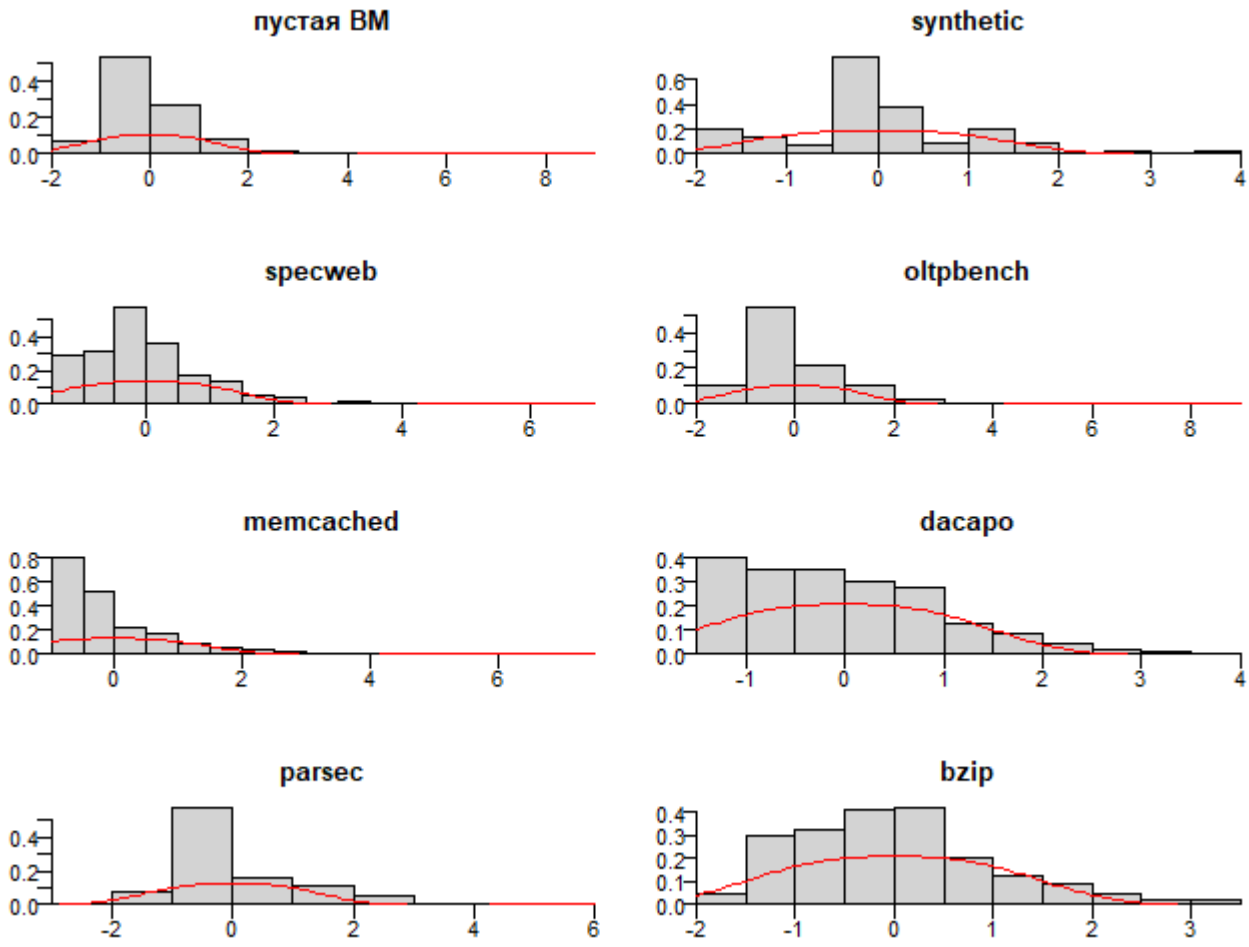

Рисунок А.13 — Аппроксимация длительности простоя ВМ рядом Грама-Шарлье для миграции с дельта-сжатием

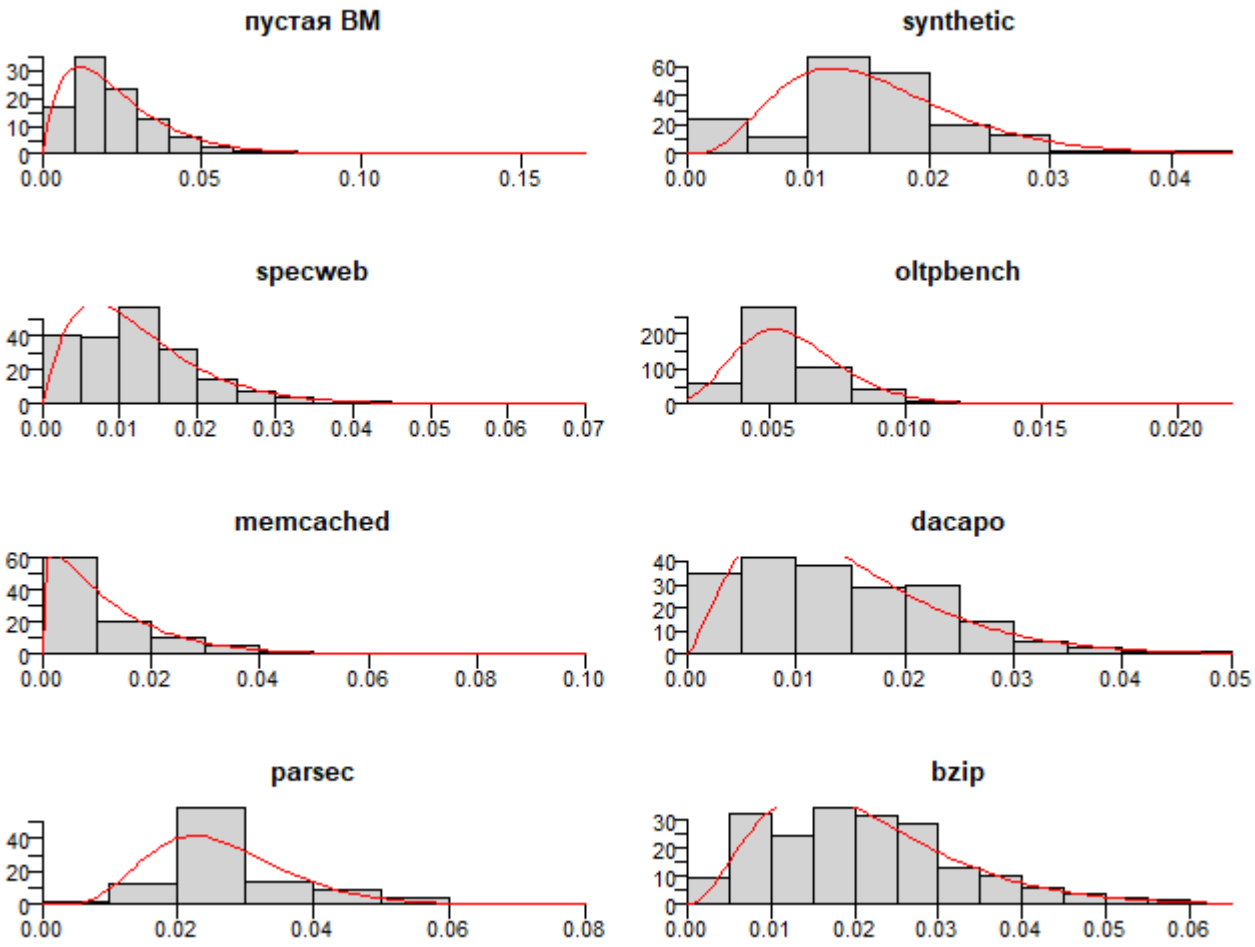

Рисунок А.14 — Аппроксимация длительности простоя ВМ рядом Лагерра для миграции с дельта-сжатием

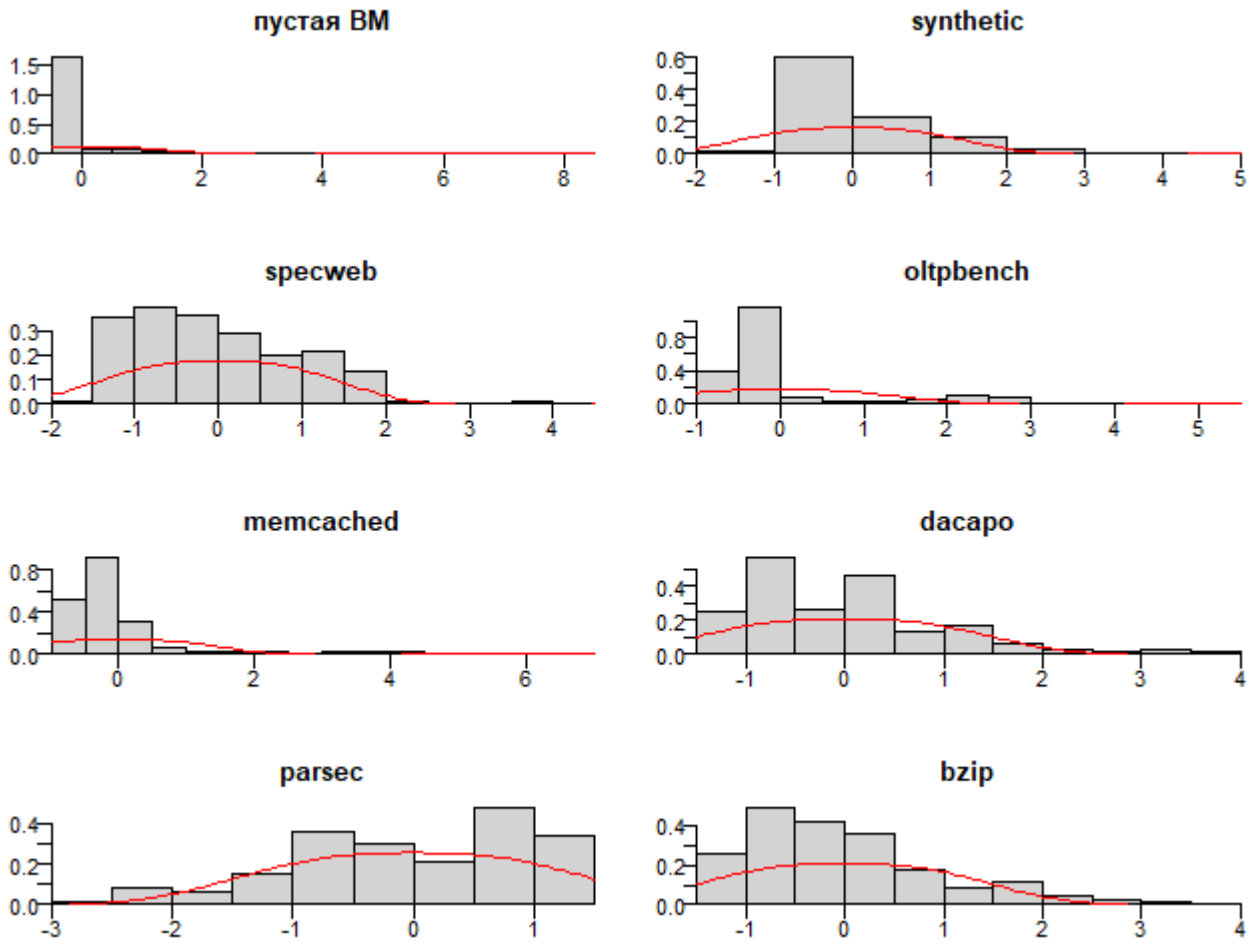

Рисунок А.15 — Аппроксимация длительности простоя ВМ рядом Грама-Шарлье для миграции с дельта-сжатием данных

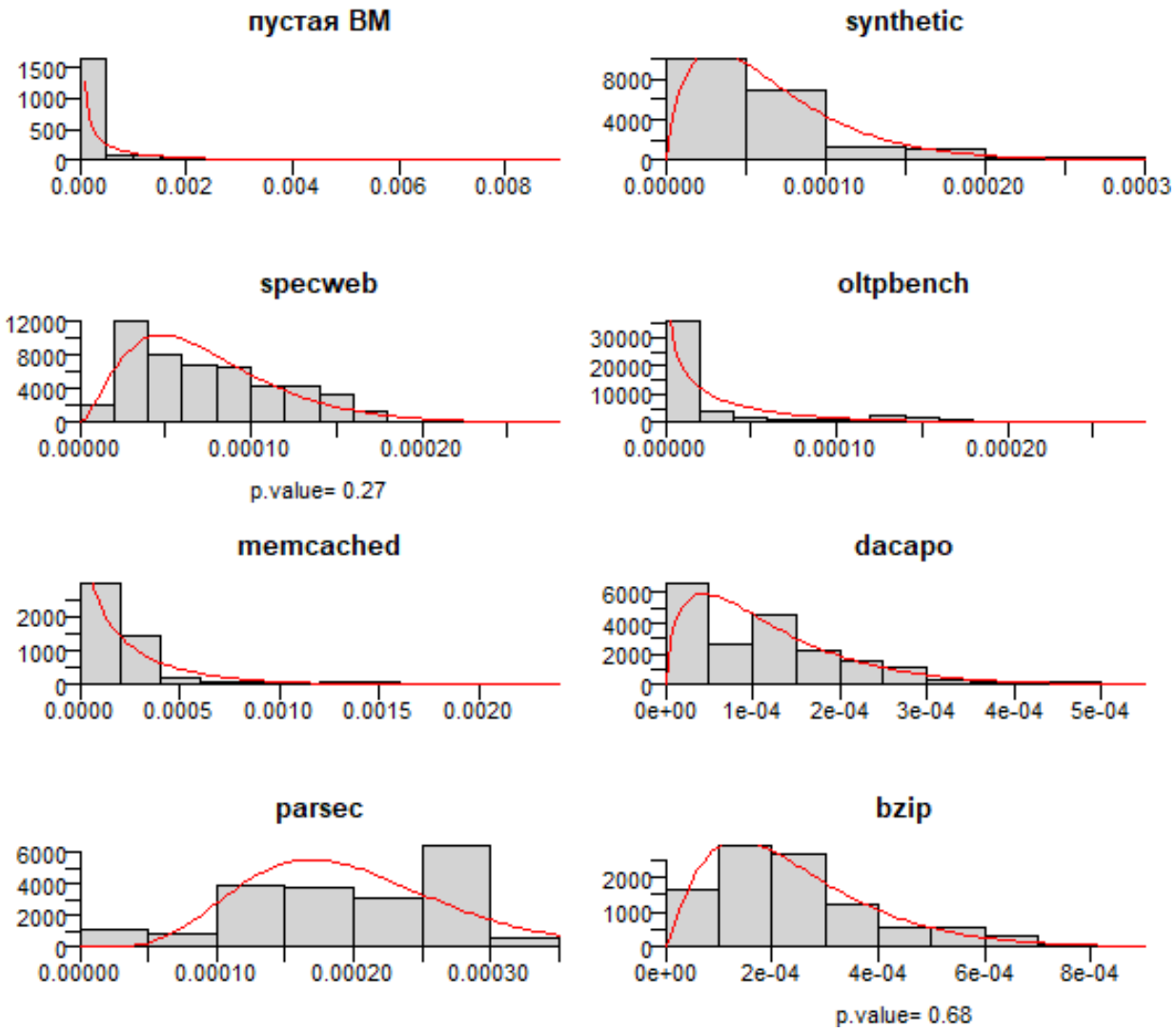

Рисунок А.16 — Аппроксимация длительности простоя ВМ рядом Лагерра для миграции с дельта-сжатием данных

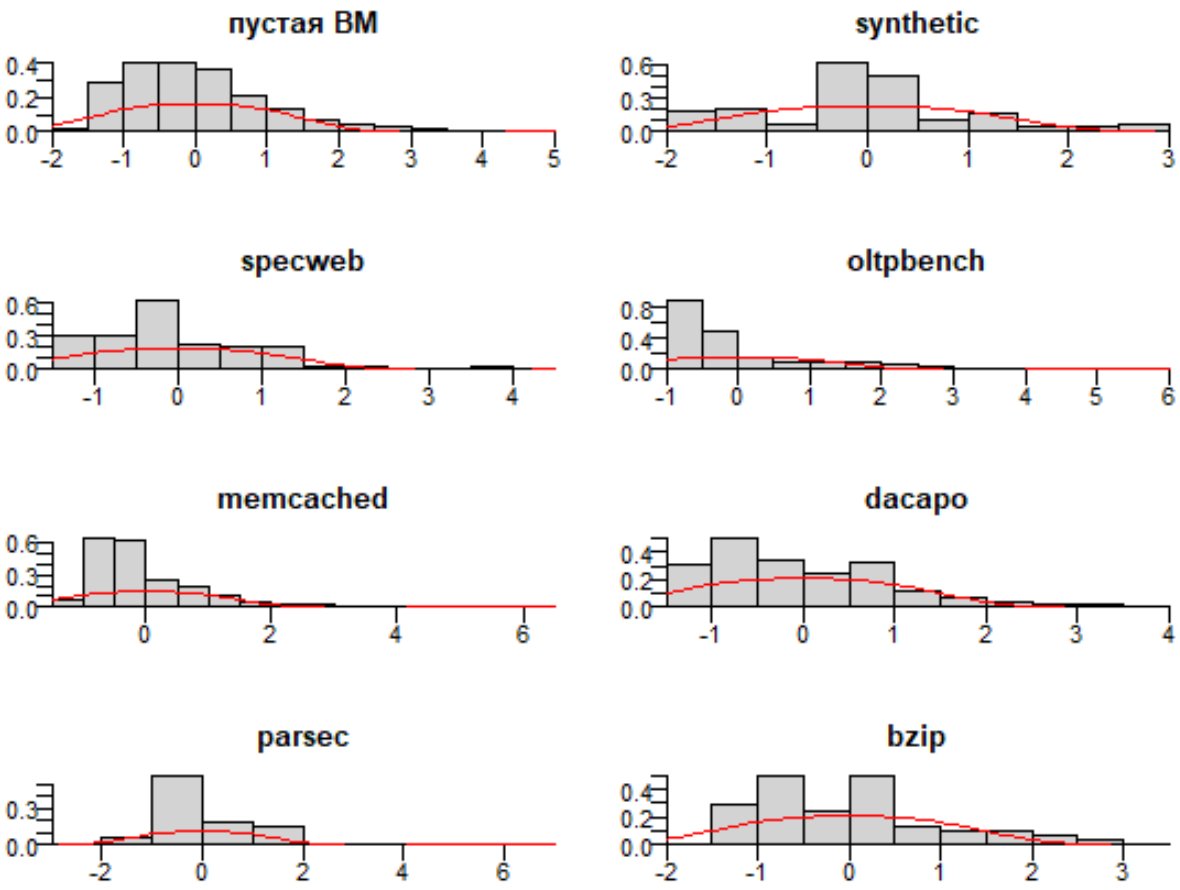

Рисунок А.17 — Аппроксимация длительности миграции ВМ рядом Грама-Шарлье для миграции с пост копированием

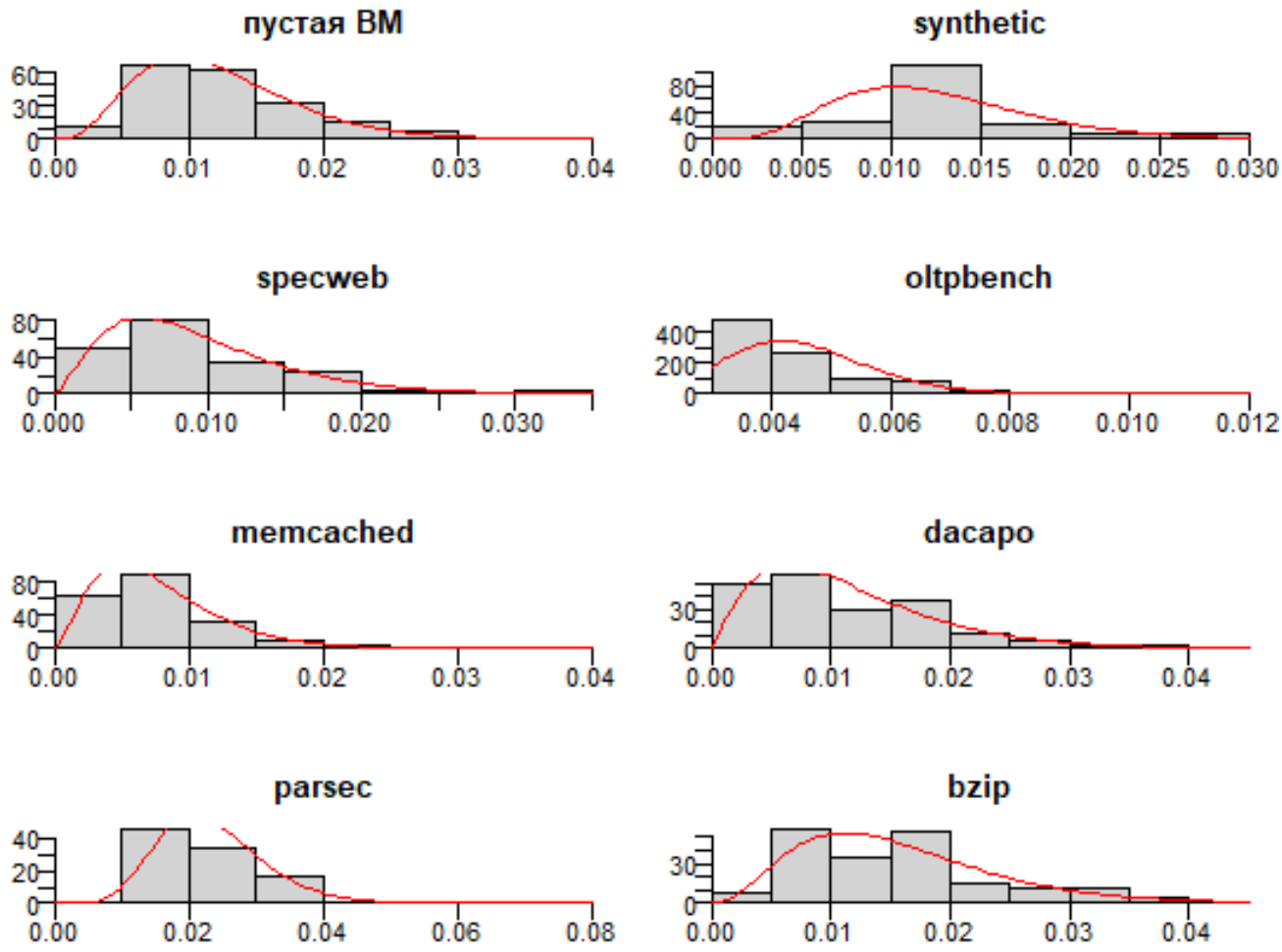

Рисунок А.18 — Аппроксимация длительности миграции ВМ рядом Лагерра для миграции с пост копированием

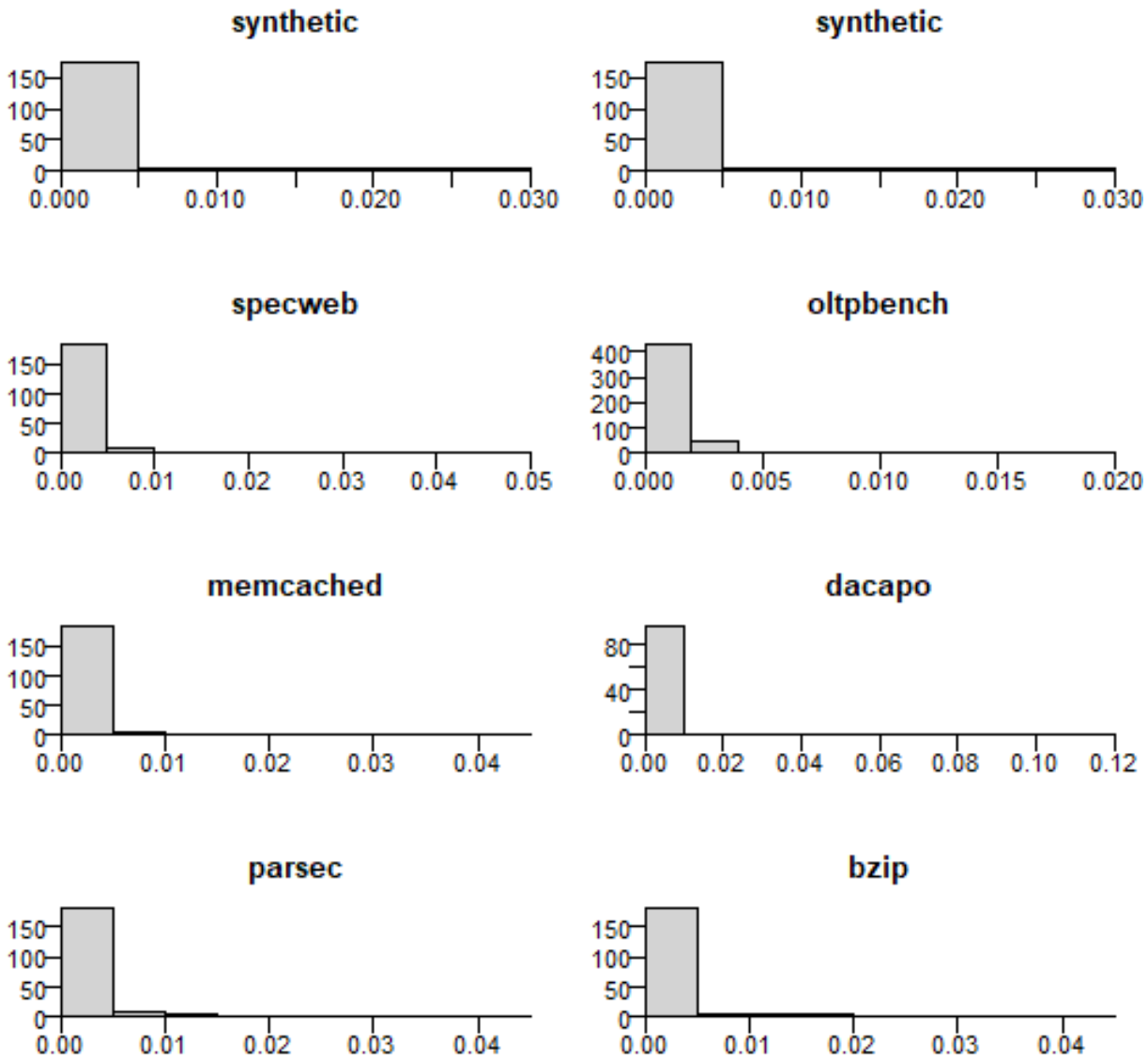

Рисунок А.19 — Гистограммы длительности простоя ВМ для миграций с пост копированием

Приложение Б

Приложение к 4-ей главе

Таблица 16 — Результаты моделирования для первых 150 экземпляров рабочих нагрузок 20110303

Алгоритм $(\alpha_1, \alpha 2)$	ESV $\cdot 10^{-3}$	E (КВт/ч)	SLAV $\cdot 10^{-5}$	SLATAH (%)	PDM (%)	Число миграций
THR-MMT-0.8 (0.9; 0.1)	5,71	20,48	2,788	13,94	0,02	5932
THR-MMT-0.8 (0.8; 0.2)	7,81	20,67	3,78	18,9	0,02	5765
THR-MMT-0.8 (0.7; 0.3)	3,99	20,27	1,97	19,7	0,01	5935
THR-MMT-0.8 (0.6; 0.4)	3,56	20,79	1,712	8,56	0,02	5679
THR-MMT-0.8 (0.5; 0.5)	2,24	21,29	1,05	5,25	0,02	5538
THR-MMT-0.8 (0.4; 0.6)	2,74	21,5	1,275	4,25	0,03	5613
THR-MMT-0.8 (0.3; 0.7)	3,68	21,95	1,677	5,59	0,03	6105
THR-MMT-0.8 (0.2; 0.8)	3,16	20,97	1,506	5,02	0,03	5481
THR-MMT-0.8 (0.1; 0.9)	1,76	21,05	0,834	4,17	0,02	5182
THR-MMT-0.8 FFD	14,06	31,57	4,455	4,95	0,09	5370

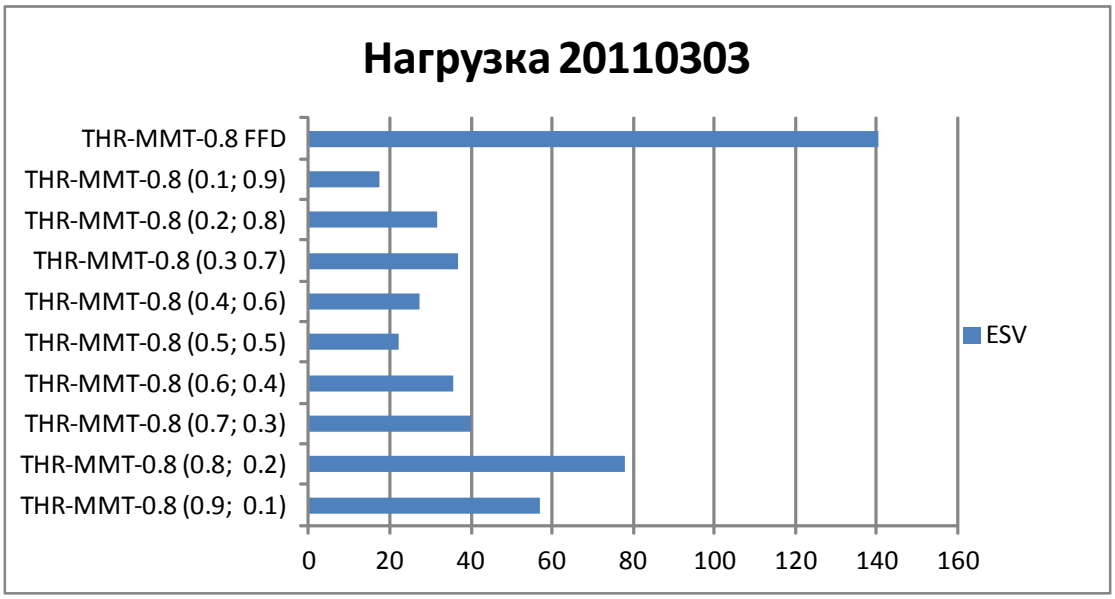

Рисунок Б.1 — Значение метрики ESV предлагаемого алгоритма размещения виртуальных машин с различными весами критериев и алгоритма FFD для рабочей нагрузки 20110303

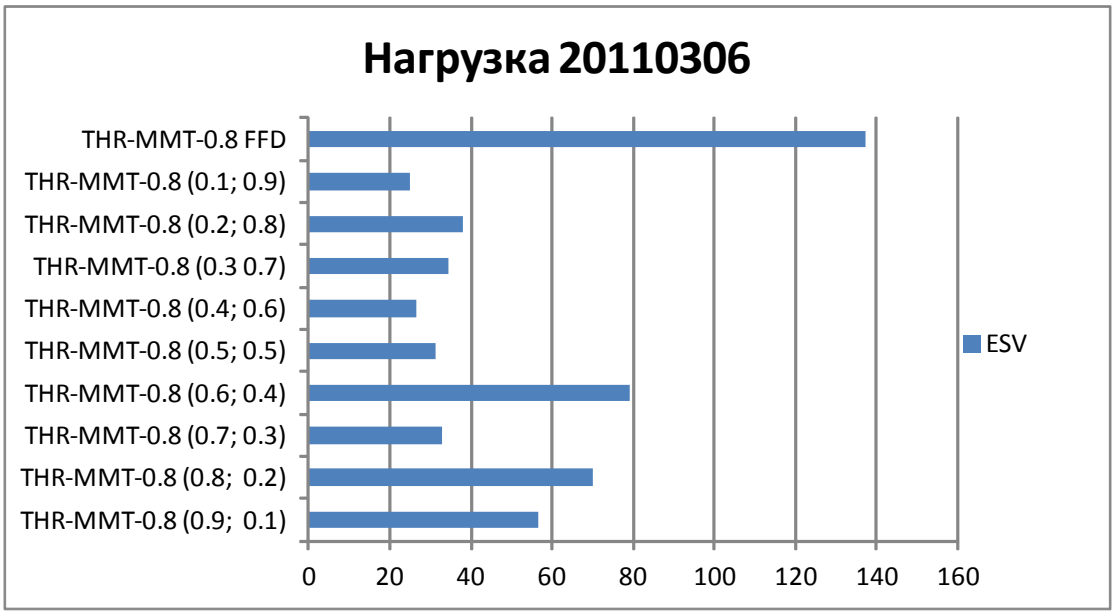

Рисунок Б.2 — Значение метрики ESV предлагаемого алгоритма размещения виртуальных машин с различными весами критериев и алгоритма FFD для рабочей нагрузки 20110306

Таблица 17 — Результаты моделирования для первых 150 экземпляров рабочих нагрузок 20110306

Алгоритм $(\alpha_1, \alpha 2)$	ESV $\cdot 10^{-3}$	E (КВт/ч)	SLAV $\cdot 10^{-5}$	SLATAH (%)	PDM (%)	Число миграций
THR-MMT-0.8 (0.9; 0.1)	56,76	17,21	3,30	16,49	0,02	4708
THR-MMT-0.8 (0.8; 0.2)	70,20	17,54	4,00	20,01	0,02	4599
THR-MMT-0.8 (0.7; 0.3)	32,71	17,29	1,89	18,92	0,01	4559
THR-MMT-0.8 (0.6; 0.4)	79,31	18,41	4,31	14,36	0,03	4829
THR-MMT-0.8 (0.5; 0.5)	31,15	18,25	1,71	5,69	0,03	4484
THR-MMT-0.8 (0.4; 0.6)	26,75	18,93	1,41	4,71	0,03	4668
THR-MMT-0.8 (0.3 0.7)	34,42	19,08	1,80	4,51	0,04	4578
THR-MMT-0.8 (0.2; 0.8)	37,89	18,83	2,01	5,03	0,04	4692
THR-MMT-0.8 (0.1; 0.9)	25,04	19,1	1,31	4,37	0,03	4728
THR-MMT-0.8 FFD	137,21	27,72	4,95	4,95	0,1	4627

Таблица 18 — Результаты моделирования для первых 150 экземпляров рабочих нагрузок 20110309

Алгоритм $(\alpha_1, \alpha 2)$	ESV $\cdot 10^{-3}$	E (КВт/ч)	SLAV $\cdot 10^{-5}$	SLATAH (%)	PDM (%)	Число миграций
THR-MMT-0.8 (0.9; 0.1)	89,20	15,69	5,685	18,95	0,03	4121
THR-MMT-0.8 (0.8; 0.2)	64,72	15,51	4,173	13,91	0,03	4025
THR-MMT-0.8 (0.7; 0.3)	78,77	16,05	4,908	16,36	0,03	3985
THR-MMT-0.8 (0.6; 0.4)	83,30	16,4	5,079	16,93	0,03	4191
THR-MMT-0.8 (0.5; 0.5)	58,85	16,44	3,58	7,16	0,05	3747
THR-MMT-0.8 (0.4; 0.6)	46,80	16,41	2,852	7,13	0,04	3697
THR-MMT-0.8 (0.3 0.7)	57,54	16,9	3,405	6,81	0,05	3993
THR-MMT-0.8 (0.2; 0.8)	53,86	16,52	3,26	6,52	0,05	3732
THR-MMT-0.8 (0.1; 0.9)	56,56	16,91	3,345	6,69	0,05	4138
THR-MMT-0.8 FFD	153,66	23,98	6,408	5,34	0,12	3935

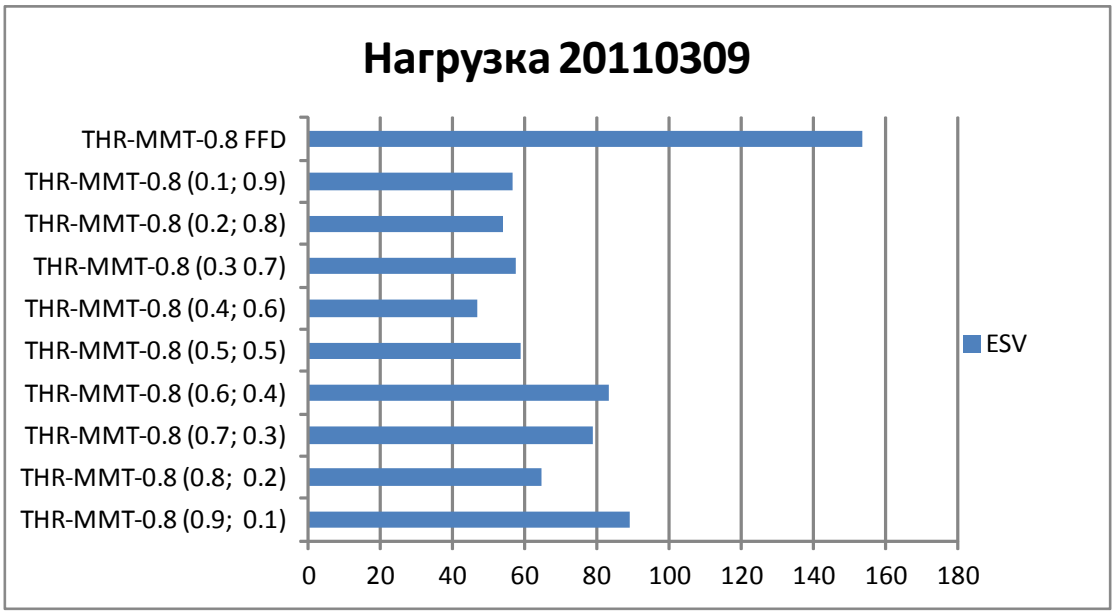

Рисунок Б.3 — Значение метрики ESV предлагаемого алгоритма размещения виртуальных машин с различными весами критериев и алгоритма FFD для рабочей нагрузки 20110309

Таблица 19 — Результаты моделирования для первых 150 экземпляров рабочих нагрузок 20110322

Алгоритм $(\alpha_1, \alpha 2)$	ESV $\cdot 10^{-3}$	E (КВт/ч)	SLAV $\cdot 10^{-5}$	SLATAH (%)	PDM (%)	Число миграций
THR-MMT-0.8 (0.9; 0.1)	26,17	14,16	1,848	9,24	0,02	4091
THR-MMT-0.8 (0.8; 0.2)	26,57	14,35	1,852	9,26	0,02	4115
THR-MMT-0.8 (0.7; 0.3)	35,97	14,6	2,464	12,32	0,02	4342
THR-MMT-0.8 (0.6; 0.4)	51,53	15,24	3,381	11,27	0,03	4237
THR-MMT-0.8 (0.5; 0.5)	33,64	15,96	2,108	5,27	0,04	3668
THR-MMT-0.8 (0.4; 0.6)	36,20	16,22	2,232	5,58	0,04	3800
THR-MMT-0.8 (0.3 0.7)	34,88	17	2,052	5,13	0,04	4266
THR-MMT-0.8 (0.2; 0.8)	31,08	16,46	1,888	4,72	0,04	3902
THR-MMT-0.8 (0.1; 0.9)	25,71	16,26	1,581	5,27	0,03	3987
THR-MMT-0.8 FFD	115,26	23,38	4,93	4,93	0,1	4183

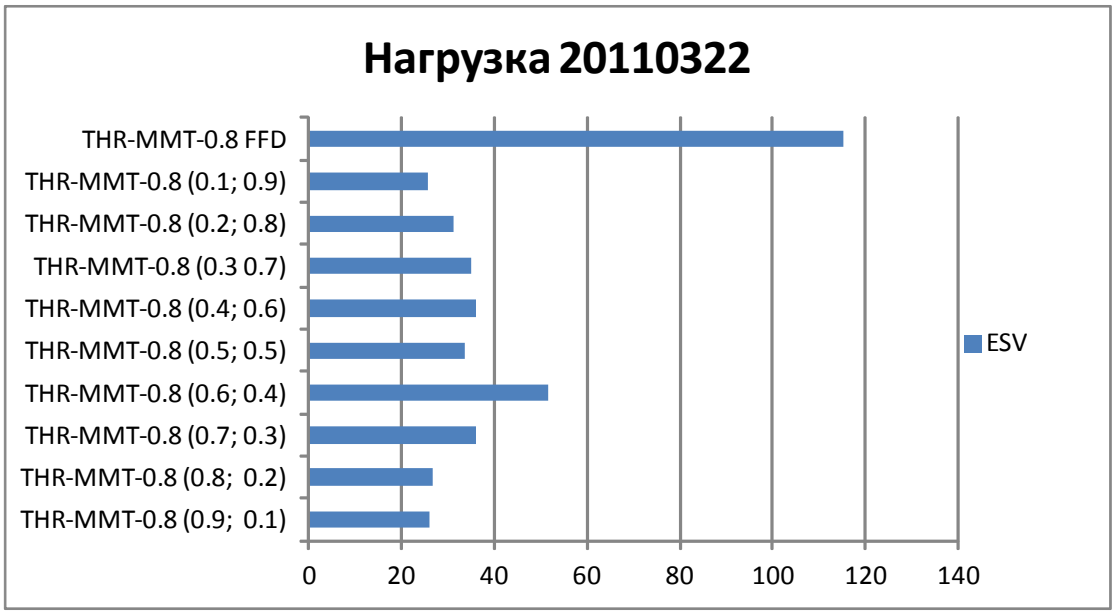

Рисунок Б.4 — Значение метрики ESV предлагаемого алгоритма размещения виртуальных машин с различными весами критериев и алгоритма FFD для рабочей нагрузки 20110322

Таблица 20 — Результаты моделирования для первых 150 экземпляров рабочих нагрузок 20110325

Алгоритм $(\alpha_1, \alpha 2)$	ESV $\cdot 10^{-3}$	E (КВт/ч)	SLAV $\cdot 10^{-5}$	SLATAH (%)	PDM (%)	Число миграций
THR-MMT-0.8 (0.9; 0.1)	51,86	16,75	3,096	15,48	0,02	5075
THR-MMT-0.8 (0.8; 0.2)	65,85	17,33	3,8	19	0,02	5396
THR-MMT-0.8 (0.7; 0.3)	37,93	17,32	2,19	10,95	0,02	4756
THR-MMT-0.8 (0.6; 0.4)	36,35	17,26	2,106	10,53	0,02	4853
THR-MMT-0.8 (0.5; 0.5)	49,45	18,59	2,66	6,65	0,04	4556
THR-MMT-0.8 (0.4; 0.6)	41,00	18,11	2,264	5,66	0,04	4523
THR-MMT-0.8 (0.3 0.7)	55,64	19,51	2,852	7,13	0,04	5244
THR-MMT-0.8 (0.2; 0.8)	35,24	19,1	1,845	6,15	0,03	5019
THR-MMT-0.8 (0.1; 0.9)	41,21	19,24	2,142	7,14	0,03	4796
THR-MMT-0.8 FFD	142,08	28,08	5,06	5,06	0,1	4740

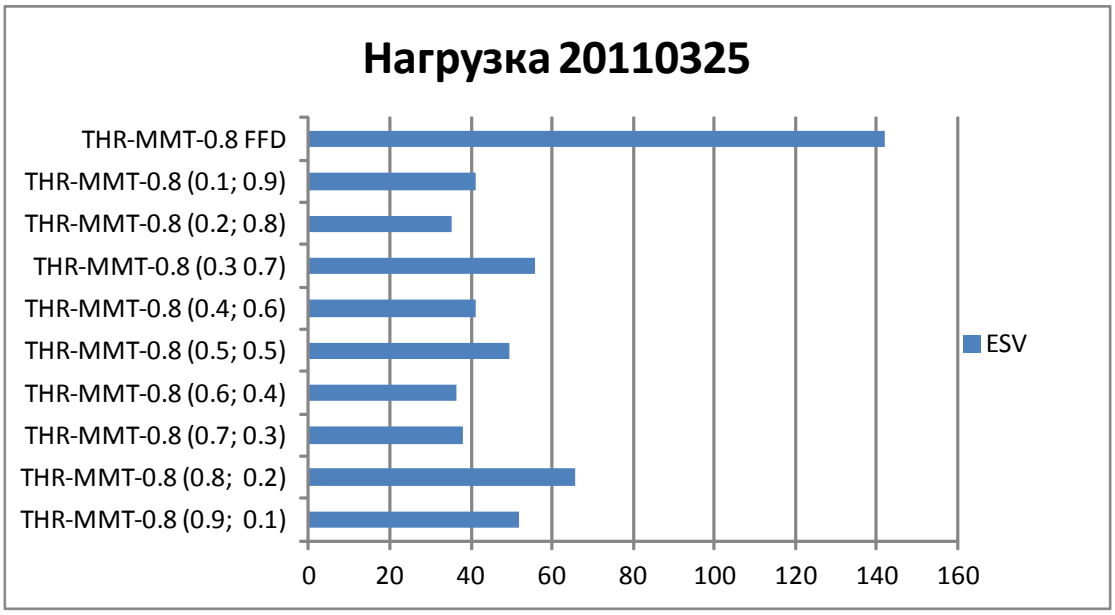

Рисунок Б.5 — Значение метрики ESV предлагаемого алгоритма размещения виртуальных машин с различными весами критериев и алгоритма FFD для рабочей нагрузки 20110325

Таблица 21 — Результаты моделирования для первых 150 экземпляров рабочих нагрузок 20110403

Алгоритм $(\alpha_1, \alpha 2)$	ESV $\cdot 10^{-3}$	E (КВт/ч)	SLAV $\cdot 10^{-5}$	SLATAH (%)	PDM (%)	Число миграций
THR-MMT-0.8 (0.9; 0.1)	88,60	18,52	4,784	23,92	0,02	5426
THR-MMT-0.8 (0.8; 0.2)	63,43	18,7	3,392	16,96	0,02	5161
THR-MMT-0.8 (0.7; 0.3)	76,91	19,17	4,012	20,06	0,02	4891
THR-MMT-0.8 (0.6; 0.4)	98,19	18,67	5,259	17,53	0,03	5408
THR-MMT-0.8 (0.5; 0.5)	32,40	19,89	1,629	5,43	0,03	4809
THR-MMT-0.8 (0.4; 0.6)	32,14	19,66	1,635	5,45	0,03	4897
THR-MMT-0.8 (0.3 0.7)	31,08	16,46	1,888	4,72	0,04	3902
THR-MMT-0.8 (0.2; 0.8)	37,74	19,81	1,905	6,35	0,03	5239
THR-MMT-0.8 (0.1; 0.9)	28,26	19,46	1,452	4,84	0,03	4664
THR-MMT-0.8 FFD	125,02	28,35	4,41	4,9	0,09	4593

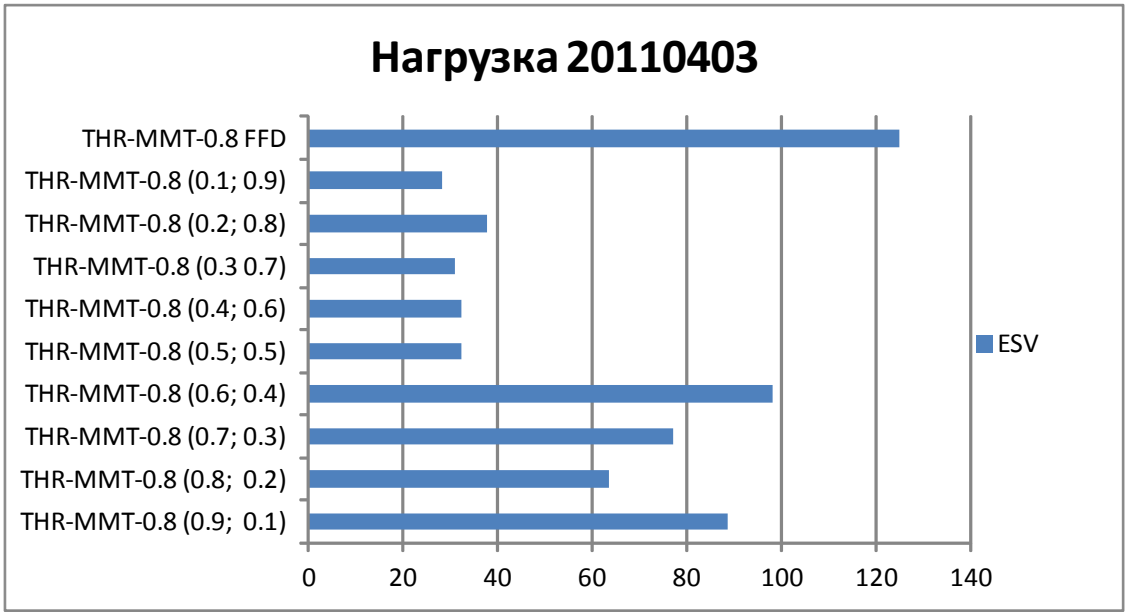

Рисунок Б.6 — Значение метрики ESV предлагаемого алгоритма размещения виртуальных машин с различными весами критериев и алгоритма FFD для рабочей нагрузки 20110403

Таблица 22 — Результаты моделирования для первых 150 экземпляров рабочих нагрузок 20110409

Алгоритм $(\alpha_1, \alpha 2)$	ESV $\cdot 10^{-3}$	E (КВт/ч)	SLAV $\cdot 10^{-5}$	SLATAH (%)	PDM (%)	Число миграций
THR-MMT-0.8 (0.9; 0.1)	30,75	17,53	1,754	17,54	0,01	4621
THR-MMT-0.8 (0.8; 0.2)	56,87	17,65	3,222	16,11	0,02	4791
THR-MMT-0.8 (0.7; 0.3)	63,76	17,92	3,558	17,79	0,02	5020
THR-MMT-0.8 (0.6; 0.4)	80,58	19,13	4,212	14,04	0,03	4964
THR-MMT-0.8 (0.5; 0.5)	35,57	18,91	1,881	6,27	0,03	4756
THR-MMT-0.8 (0.4; 0.6)	35,47	19,51	1,818	6,06	0,03	4823
THR-MMT-0.8 (0.3 0.7)	43,85	20,19	2,172	5,43	0,04	5074
THR-MMT-0.8 (0.2; 0.8)	28,79	19,83	1,452	4,84	0,03	4924
THR-MMT-0.8 (0.1; 0.9)	31,45	20,12	1,563	5,21	0,03	5271
THR-MMT-0.8 FFD	148,63	29,03	5,12	5,12	0,1	4758

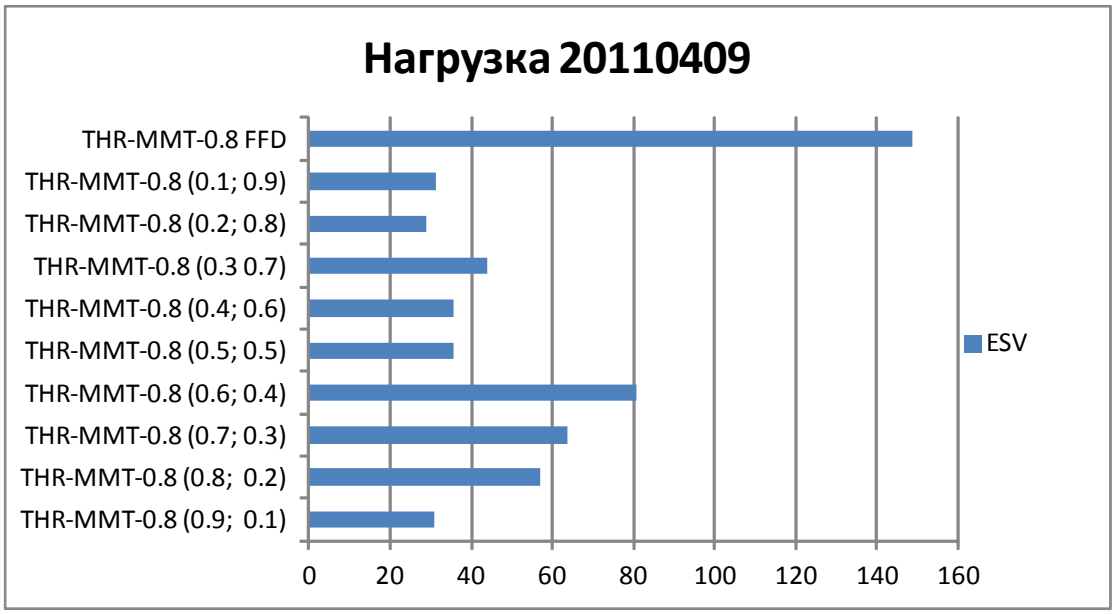

Рисунок Б.7 — Значение метрики ESV предлагаемого алгоритма размещения виртуальных машин с различными весами критериев и алгоритма FFD для рабочей нагрузки 20110409

Таблица 23 — Результаты моделирования для первых 150 экземпляров рабочих нагрузок 20110411

Алгоритм $(\alpha_1, \alpha 2)$	ESV $\cdot 10^{-3}$	E (КВт/ч)	SLAV $\cdot 10^{-5}$	SLATAH (%)	PDM (%)	Число миграций
THR-MMT-0.8 (0.9; 0.1)	30,96	15,94	1,942	9,71	0,02	4787
THR-MMT-0.8 (0.8; 0.2)	34,28	15,93	2,152	10,76	0,02	4926
THR-MMT-0.8 (0.7; 0.3)	26,20	15,92	1,646	8,23	0,02	4991
THR-MMT-0.8 (0.6; 0.4)	58,75	17,47	3,363	11,21	0,03	5488
THR-MMT-0.8 (0.5; 0.5)	26,48	17,14	1,545	5,15	0,03	4176
THR-MMT-0.8 (0.4; 0.6)	24,96	17,12	1,458	4,86	0,03	4465
THR-MMT-0.8 (0.3 0.7)	23,16	17,75	1,305	4,35	0,03	4734
THR-MMT-0.8 (0.2; 0.8)	24,54	17,37	1,413	4,71	0,03	4418
THR-MMT-0.8 (0.1; 0.9)	14,76	17,37	0,85	4,25	0,02	4487
THR-MMT-0.8 FFD	122,99	26,33	4,671	5,19	0,09	4747

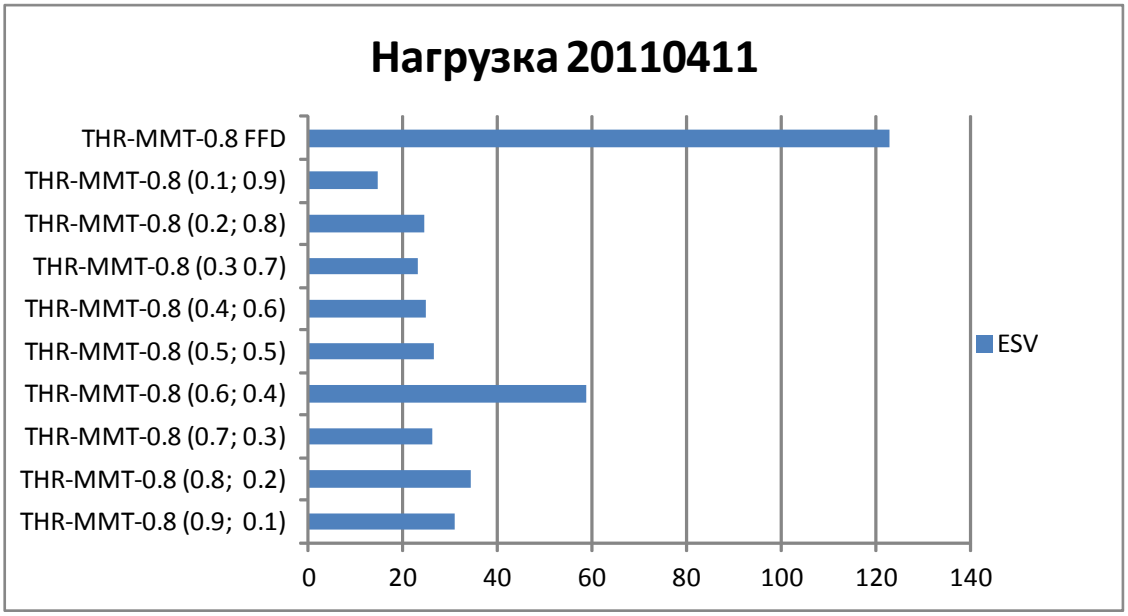

Рисунок Б.8 — Значение метрики ESV предлагаемого алгоритма размещения виртуальных машин с различными весами критериев и алгоритма FFD для рабочей нагрузки 20110411

Таблица 24 — Результаты моделирования для первых 150 экземпляров рабочих нагрузок 20110412

Алгоритм $(\alpha_1, \alpha 2)$	ESV $\cdot 10^{-3}$	E (КВт/ч)	SLAV $\cdot 10^{-5}$	SLATAH (%)	PDM (%)	Число миграций
THR-MMT-0.8 (0.9; 0.1)	76,69	19,93	3,848	19,24	0,02	5720
THR-MMT-0.8 (0.8; 0.2)	67,30	19,9	3,382	16,91	0,02	5853
THR-MMT-0.8 (0.7; 0.3)	110,99	19,94	5,566	27,83	0,02	6511
THR-MMT-0.8 (0.6; 0.4)	57,14	20,12	2,84	14,2	0,02	5614
THR-MMT-0.8 (0.5; 0.5)	33,85	20,74	1,632	5,44	0,03	5325
THR-MMT-0.8 (0.4; 0.6)	43,31	20,83	2,079	6,93	0,03	5945
THR-MMT-0.8 (0.3 0.7)	40,61	20,89	1,944	6,48	0,03	5397
THR-MMT-0.8 (0.2; 0.8)	29,03	20,9	1,389	4,63	0,03	5397
THR-MMT-0.8 (0.1; 0.9)	26,37	20,83	1,266	6,33	0,02	5561
THR-MMT-0.8 FFD	158,29	31,47	5,03	5,03	0,1	5271

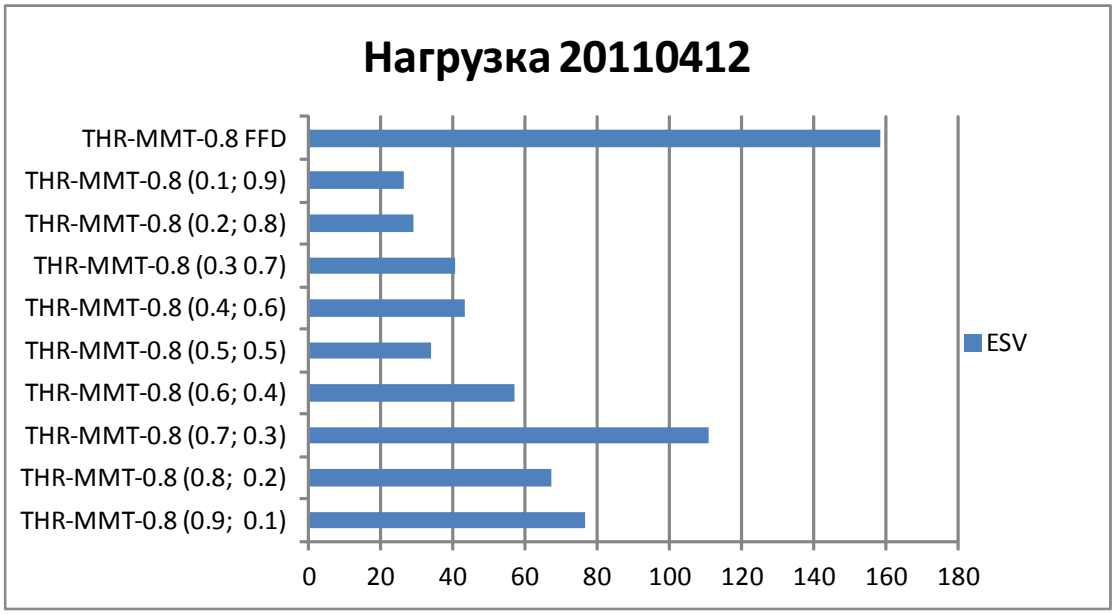

Рисунок Б.9 — Значение метрики ESV предлагаемого алгоритма размещения виртуальных машин с различными весами критериев и алгоритма FFD для рабочей нагрузки 20110412

Таблица 25 — Результаты моделирования для первых 150 экземпляров рабочих нагрузок 20110420

Алгоритм $(\alpha_1, \alpha 2)$	ESV $\cdot 10^{-3}$	E (КВт/ч)	SLAV $\cdot 10^{-5}$	SLATAH (%)	PDM (%)	Число миграций
THR-MMT-0.8 (0.9; 0.1)	45,19	12,46	3,627	12,09	0,03	3953
THR-MMT-0.8 (0.8; 0.2)	48,53	12,19	3,981	13,27	0,03	4286
THR-MMT-0.8 (0.7; 0.3)	47,70	12,45	3,831	12,77	0,03	4051
THR-MMT-0.8 (0.6; 0.4)	37,50	13,24	2,832	9,44	0,03	4072
THR-MMT-0.8 (0.5; 0.5)	30,00	13,89	2,16	5,4	0,04	3122
THR-MMT-0.8 (0.4; 0.6)	30,98	13,71	2,26	5,65	0,04	3464
THR-MMT-0.8 (0.3 0.7)	32,56	14,38	2,264	5,66	0,04	3686
THR-MMT-0.8 (0.2; 0.8)	28,85	14,28	2,02	5,05	0,04	3552
THR-MMT-0.8 (0.1; 0.9)	25,75	14,18	1,816	4,54	0,04	3331
THR-MMT-0.8 FFD	126,52	21,26	5,951	5,41	0,11	4157

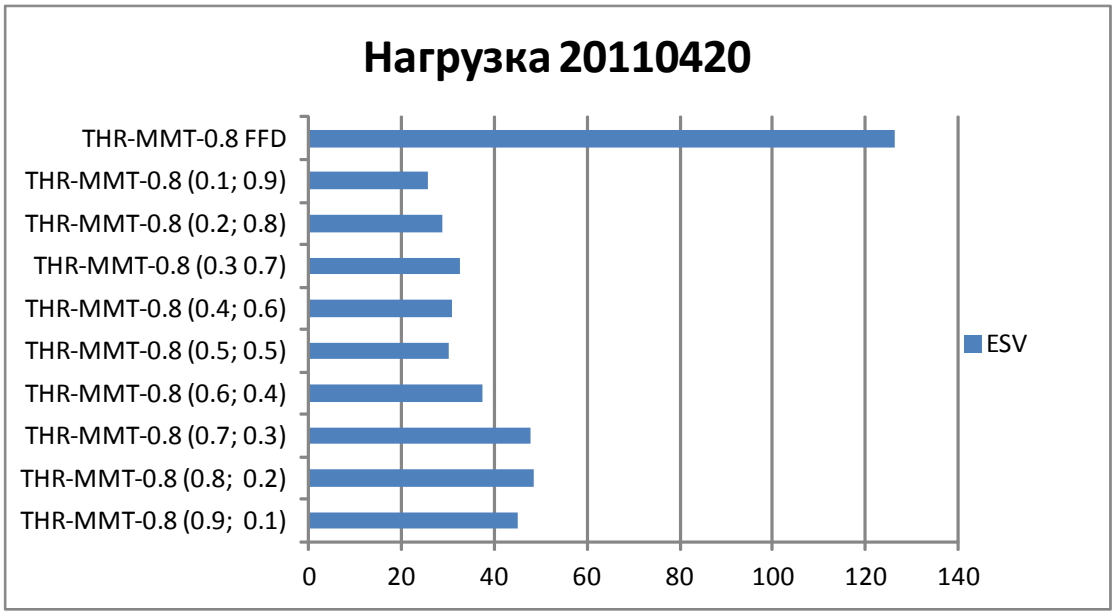

Рисунок Б.10 — Значение метрики ESV предлагаемого алгоритма размещения виртуальных машин с различными весами критериев и алгоритма FFD для рабочей нагрузки 20110420

Приложение В

Приложение к 5-ой главе

Рисунок В.1 — Этапы первоначального размещения виртуальных машин

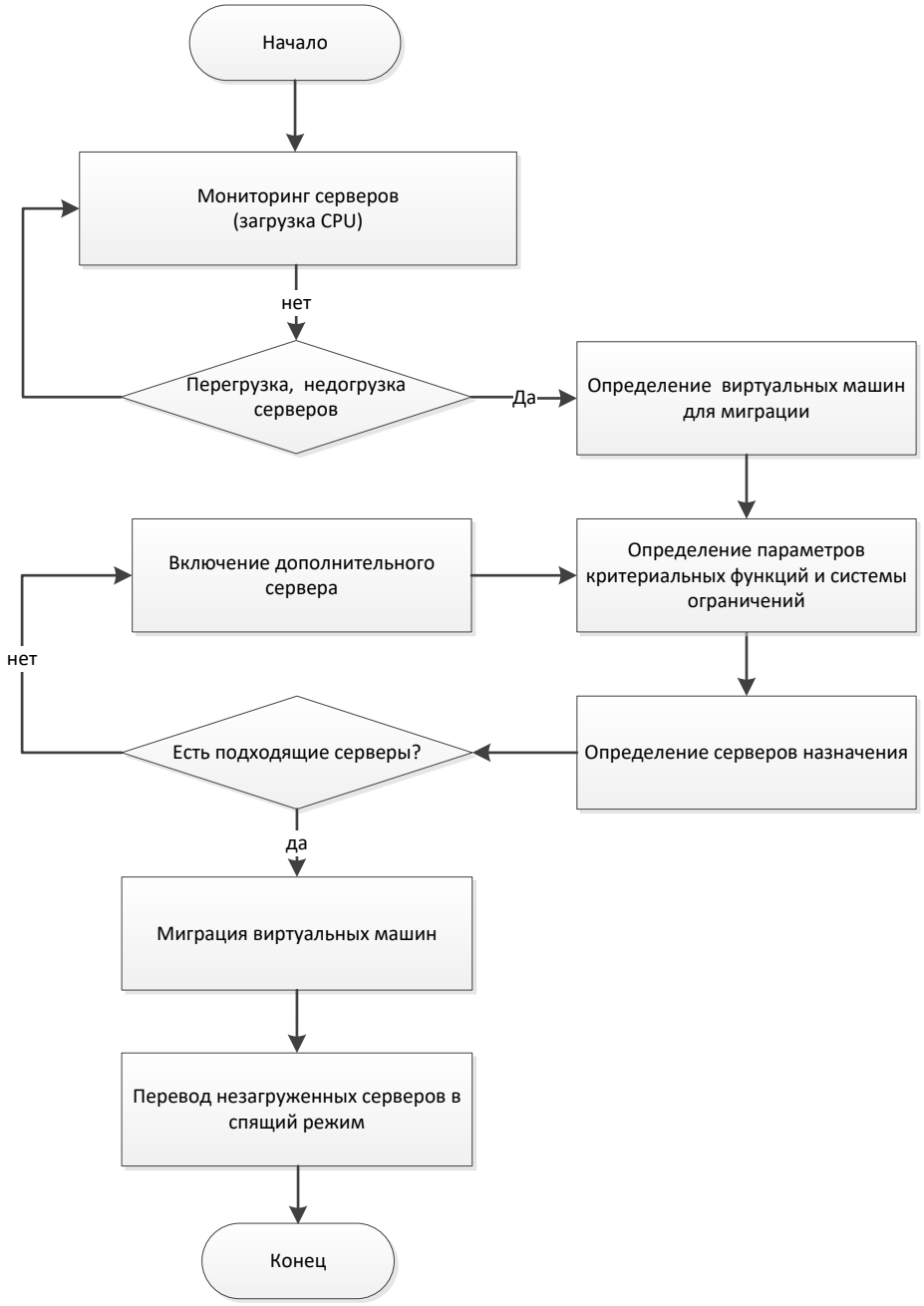

Рисунок В.2 — Этапы динамического размещения виртуальных машин

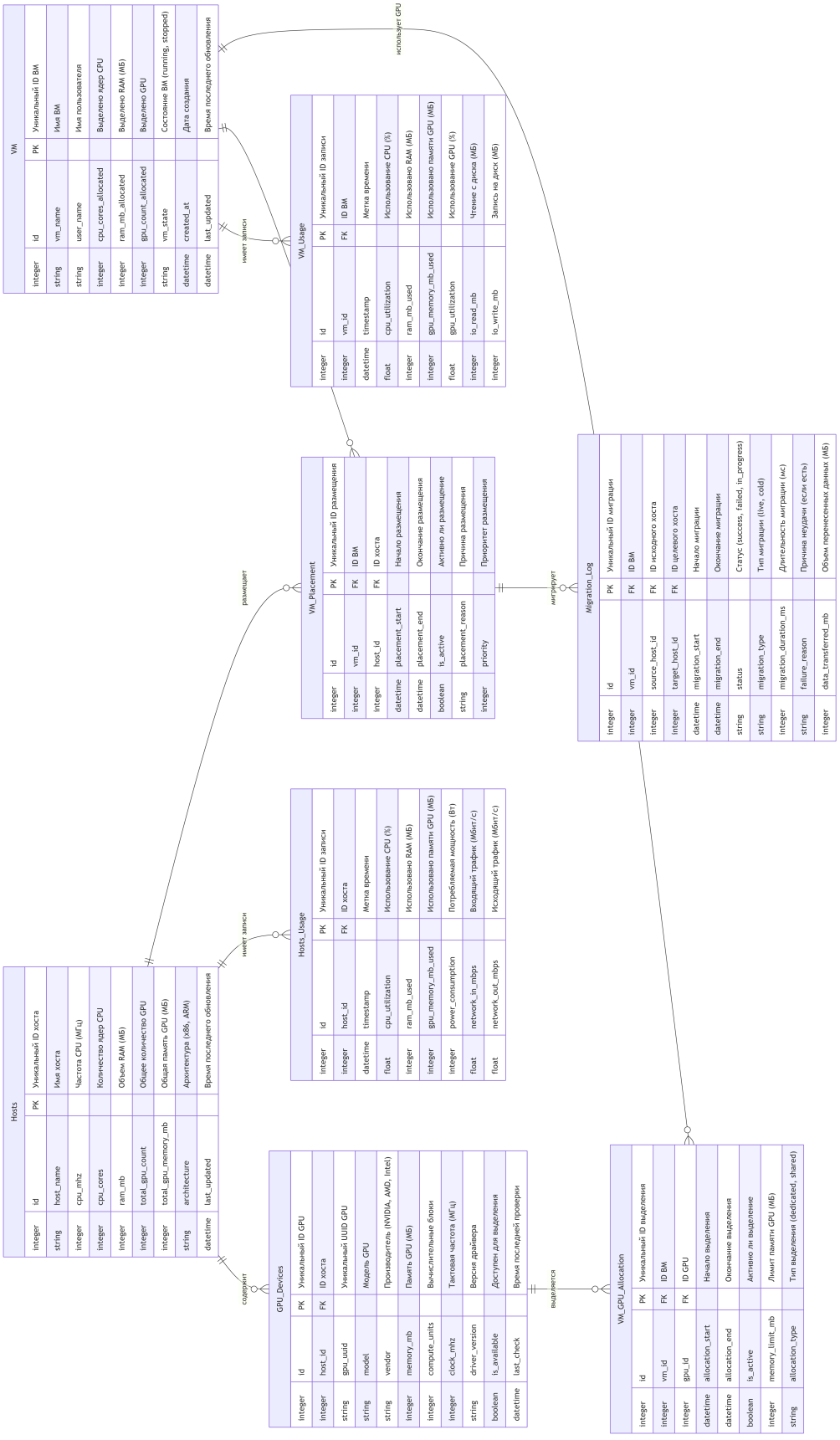

Рисунок В.3 — Схема базы данных глобального контроллера

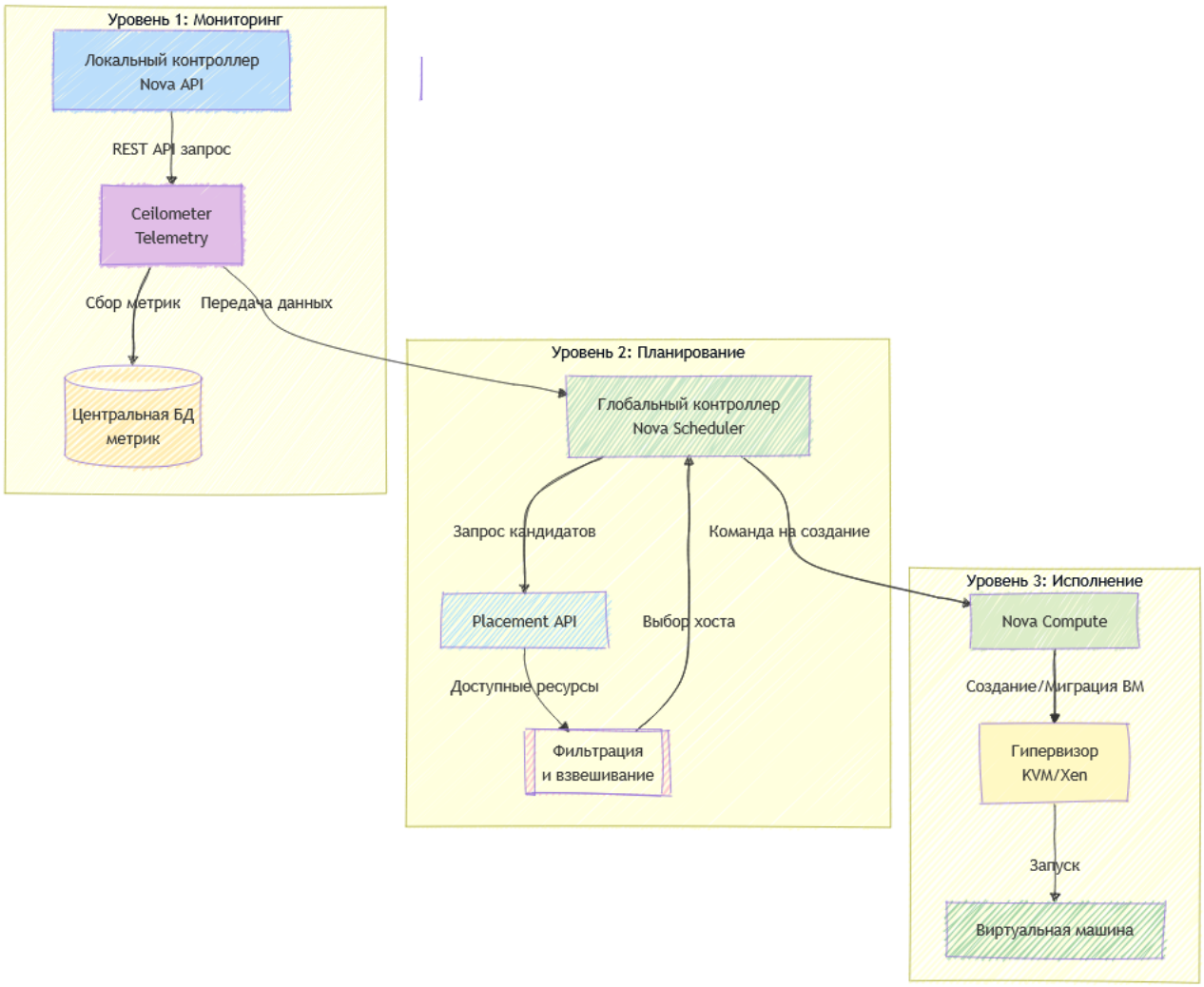

Рисунок В.4 — Блок-схема работы планировщика ресурсов интегрированного в платформу OpenStack

Приложение Г

Свидетельства о регистрации программы для ЭВМ

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2018666780

Программа для прогнозирования перегрузки серверов с использованием комбинаторного метода группового учета аргументов на языке программирования Java

Правообладатель: ***ОРДЕНА ТРУДОВОГО КРАСНОГО ЗНАМЕНИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ "МОСКОВСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ СВЯЗИ И ИНФОРМАТИКИ" (RU)***

Авторы: ***Тутов Андрей Владимирович (RU), Тутова Наталья Владимировна (RU), Ворожцов Анатолий Сергеевич (RU)***

Заявка № **2018664122**

Дата поступления **07 декабря 2018 г.**

Дата государственной регистрации в Реестре программ для ЭВМ ***20 декабря 2018 г.***

Руководитель Федеральной службы по интеллектуальной собственности

Г.П. Ивлиев

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2025693891

Программа для динамического планирования размещения виртуальных машин по физическим серверам в центрах обработки данных

Правообладатель: ***Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования «Московский технический университет связи и информатики» (RU)***

Автор(ы): ***Тутов Андрей Владимирович (RU)***

Заявка № **2025692950**

Дата поступления **18 ноября 2025 г.**

Дата государственной регистрации в Реестре программ для ЭВМ ***02 декабря 2025 г.***

Руководитель Федеральной службы по интеллектуальной собственности

ДОКУМЕНТ ПОДПИСАН ЭЛЕКТРОННОЙ ПОДПИСЬЮ
Сертификат 00a570e4f7add8d531b4b8818e75f29506
Владелец **Зубов Юрий Сергеевич**
Действителен с 04.09.2025 по 28.11.2026

Ю.С. Зубов

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2025697058

«Программа для первоначального размещения виртуальных машин в гетерогенном центре обработки данных»

Правообладатель: ***Федеральное государственное бюджетное учреждение науки Институт проблем управления им. В.А. Трапезникова Российской академии наук (RU)***

Авторы: ***Фархадов Маис Паша оглы (RU), Тутов Андрей Владимирович (RU)***

Заявка № **2025696307**
Дата поступления **18 декабря 2025 г.**
Дата государственной регистрации
в Реестре программ для ЭВМ ***22 декабря 2025 г.***

Руководитель Федеральной службы
по интеллектуальной собственности

Ю.С. Зубов

Приложение Д

Акты о внедрении

«УТВЕРЖДАЮ»
Проректор по научной работе Ордена Трудового Красного Знамени федерального государственного бюджетного образовательного учреждения высшего образования «Московский технический университет связи и информатики»
д.т.н., профессор Ю.Л. Леохин
«__» _____ 2025 г.

АКТ

об использовании результатов диссертационной работы Тутова Андрея Владимировича на тему «Модели и методы распределения информационных и вычислительных ресурсов гетерогенных центров обработки данных» в учебном процессе кафедры «Бизнес-информатика» МТУСИ

Комиссия в составе начальника отдела планирования и организации учебного процесса МТУСИ В.А. Кузнецовой, руководителя центра заочного обучения по программам магистратуры Е.Г. Кухаренко и доцента кафедры «Бизнес-информатика» А.Г. Ерохина, составила настоящий акт о том, что результаты диссертационной работы А.В. Тутова внедрены в учебный процесс кафедры «Бизнес-информатика» МТУСИ, в частности:
– метод многокритериального динамического размещения виртуальных машин по критериям нарушения соглашений об уровне сервиса и эффективности использования вычислительных ресурсов;
– имитационные модели распределения ресурсов облачного ЦОД, которые позволяют оценить эффективность алгоритмов многокритериального динамического размещения виртуальных машин.

Форма внедрения — курсовое проектирование по дисциплине «Вычислительные системы» для направления магистратуры 09.04.01 «Информатика и вычислительная техника» (программа «Архитектура информационных систем»).

Начальника отдела планирования и организации учебного процесса — В.А. Кузнецова

Руководитель ЦЗОПМ — Е.Г. Кухаренко

Доцент кафедры «Бизнес-информатика» — А.Г. Ерохин

МТС Ведите бизнес вперёд

Публичное акционерное общество

Публичное акционерное общество
«Мобильные Телесистемы»

ул. Марксистская, 4, Москва, 109147
Тел. +7 (495) 911-71-51, факс +7 (495) 911-65-69
факс +7 (495) 911-65-69
www.corp.mts.ru

«УТВЕРЖДАЮ»

Руководитель группы ПАО МТС

«01» декабря 2025 г.

А К Т

внедрения результатов диссертационной работы Тутова А.В.

Настоящий акт составлен о том, что отдельные результаты диссертационной работы Тутова А.В. на тему «Модели и методы распределения информационных и вычислительных ресурсов гетерогенных центров обработки данных» внедрены в тестовую среду Публичного акционерного общества «МТС». Опробованы метод первоначального размещения группы виртуальных машин и алгоритмы динамического распределения, в части методов прогнозирования перегрузки и простоя физических серверов и многокритериального определения узлов назначения.

Применение указанных методов показало снижение энергопотребления вычислительной подсистемы гетерогенного ЦОД в среднем до 7% за счет отключения незагруженных серверов при выполнении требований на время отклика и стабильности работы сервисов. По результатам составлены рекомендации по внедрению результатов диссертационной работы в облачную инфраструктуру нового поколения ПАО «МТС» на базе платформы OpenStack.

Руководитель группы В.А. Костебелова

ООО "АЭР – Технологические Решения"
Российская Федерация, 123112, г. Москва,
Муниципальный округ Пресненский вн.тер.г.,
Пресненская наб., д.10
ОГРН: 1217700509311
ИНН: 9703055296
Тел.: +7 (495) 139 00 08

АКТ

о внедрении результатов диссертационной работы Тутова Андрея Владимировича «Модели и методы распределения информационных и вычислительных ресурсов гетерогенных центров обработки данных», представленной на соискание ученой степени кандидата технических наук по специальности 2.3.8 «Информатика и информационные процессы»

Настоящий акт составлен о том, что результаты диссертационной работы Тутова А. В. «Модели и методы распределения информационных и вычислительных ресурсов гетерогенных центров обработки данных» были апробированы в системе управления ресурсами центра обработки данных ООО «АЭР-Технологические решения».

Предложенные алгоритмы первоначального и динамического размещения виртуальных машин показали свою эффективность. Практические результаты диссертационной работы Тутова А. В. могут быть использованы разработчиками облачных платформ и провайдерами для управления ресурсами гетерогенных вычислительных систем в центрах обработки данных.

Технический директор
ООО «АЭР-Технологические решения» А. А. Орлов
«21» октября 2025 г.

Билайн®
Бизнес

ПАО <ВымпелКом>
ул. 8 Марта, д.10, стр.14
Москва, 127083

Телефон
+7 (495) 725 0700

Факс
+7 (495) 725 0700

Акт о внедрении результатов диссертационной работы

Тутова Андрея Владимировича на тему

«Модели и методы распределения информационных и вычислительных ресурсов гетерогенных центров обработки данных»

ПАО «Вымпелком» (бренд Билайн) в лице Руководителя службы по работе с клиентами Лысой В.В. составила настоящий акт в том, что результаты диссертационного исследования Тутова Андрея Владимировича внедрены в практическую деятельность оператора связи.

Компания располагает большим парком серверов в собственных ЦОД, поэтому проявляет большой интерес к работам, предлагающим более эффективные и продуктивные подходы к вопросам управления и обслуживания. В бизнесе компании активно применяются графические серверы, на их основе формируются перспективные разработки и предлагается широкий спектр новых услуг.

Результаты диссертационной работы в области метода расчета длительности миграции виртуальных машин были представлены в виде программных модулей с открытым исходным кодом и используются ПАО «Вымпелком» для анализа и проверки соглашений об уровне обслуживания.

В рамках облачного направления Beeline Cloud были опробованы и внедрены методы статического подхода при размещении виртуальных машин, так, алгоритм муравьиных колоний был задействован при проектировании схемы размещений на графических серверах.

Планируется расширить применение результатов работы на ключевой облачный стек Beeline Cloud 2.0 — облачную платформу нового поколения для цифровой трансформации бизнеса.

Данный акт выдан для представления в диссертационный совет.

Лысая В.В. _____

BWT BARRIER
BEST WATER FILTER TECHNOLOGIES

А К Т
о внедрении результатов диссертационной работы
Тутова Андрея Владимировича
«Модели и методы распределения информационных и вычислительных ресурсов гетерогенных центров обработки данных»
на соискание ученой степени кандидата технических наук

АО «БВТ Барьер Рус» – лидирующий производитель фильтров для воды в России. Это технологическая компания, в основе продуктов которой лежат сложные уникальные, трудновоспроизводимые технологии. Сегодня БАРЬЕР выпускает десятки видов продукции: от простых фильтров-кувшинов и домашних проточных фильтров до профессиональных систем очистки воды для загородных домов, промышленного производства, социальных объектов, энергетики и транспорта

В диссертационной работе Тутова А.В. предложены модели и методы распределения информационных и вычислительных ресурсов, которые в полном объеме используются в деятельности АО «БВТ Барьер Рус». Применение алгоритмов обеспечивает бесперебойную работу инфокоммуникационной инфраструктуры, надежность и безопасность гетерогенной среды, превентивное решение возможных проблем. Перспективным направлением исследования является последующая интеграция результатов работы с разрабатываемой корпоративной нейросетью, что повысит интеллектуальность, автономность и масштабируемость системы в условиях высокой нагрузки.

Актуальность настоящей диссертационной работы подтверждается совокупностью технологических и экономических вызовов, стоящих перед современным предприятием в условиях цифровой трансформации. Для АО «БВТ Барьер рус» это возможность развертывания своего частного облака и внедрения самых современных сервисов, как Интернет вещей и машинное обучение.

Генеральный директор В.А. Рязанов

АКЦИОНЕРНОЕ ОБЩЕСТВО «БВТ БАРЬЕР РУС» ОГРН 1025005156061 АО «БВТ БАРЬЕР РУС» BARRIER RUS МОСКОВСКАЯ ОБЛ.

Акционерное общество «БВТ БАРЬЕР РУС»
ИНН 5001017207, КПП 503101001, ОГРН 1025000510061
РФ, 142400, Московская область, Богородский городской округ,
город Ногинск, территория Ногинск-Технопарк, д. 2
www.barrier.ru

Тел.: +7 495 661 21 21
E-mail: info@barrier.ru