**Федеральное государственное казенное военное
образовательное учреждение высшего образования
«Академия Федеральной службы охраны Российской Федерации»**

На правах рукописи

Бумажкина Наталья Юрьевна

СПЕЦИАЛЬНОЕ МАТЕМАТИЧЕСКОЕ И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРОЦЕССА РАЗМЕЩЕНИЯ ВИРТУАЛЬНЫХ МАШИН В ГЕТЕРОГЕННЫХ ЦЕНТРАХ ОБРАБОТКИ ДАННЫХ

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

Диссертация на соискание ученой степени
кандидата технических наук

Научный руководитель:
доктор технических наук
Белов Андрей Сергеевич

Орел – 2024

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность темы. Одной из сложившихся тенденций в предметной области обработки данных является развитие и совершенствование методов и технологий интеллектуального анализа данных (data mining). В настоящее время они находят широкое применение, как в достаточно развитой предметной области облачных вычислений – динамического (парадигма «по запросу» – on-demand) предоставления пользователям возможности удаленного выполнения их задач, так и в активно развивающейся области систем искусственного интеллекта, поддерживающих большие языковые модели, генеративно-состязательные и другие типы моделей нейронных сетей.

Технологическая реализация решения подобных задач стала возможной благодаря развитию архитектурных решений центров обработки данных (ЦОД), обеспечивающих технологии интеллектуального анализа данных требуемыми вычислительными (compute) и коммуникационными (communication) ресурсами (далее, ресурсами), такими как процессорное время, оперативная память, хранилище данных, пропускная способность каналов связи. Особенности каждого из этих видов ресурсов не позволяют свести единицы их использования к одному параметру, а требуют необходимости рассматривать их многомерное представление.

В предметной области ЦОД важным исследовательским вопросом является повышение эффективности предоставления его ресурсов пользователям. Он может решаться, как за счет масштабирования аппаратной инфраструктуры, включая создание распределенных ЦОД, так и за счет внедрения в существующую инфраструктуру средств виртуализации ресурсов. Подобные ЦОД именуются виртуализированными (виртуальными) или программно-определяемыми (SDDC – software defined data center). Основой виртуализированного ЦОД (ВЦОД) являются гипервизоры виртуальных машин – программные модули управления, которые позволяют

нескольким виртуальным машинам одновременно использовать ресурсы одной серверной платформы (далее физической машины).

В процессе обслуживания множества пользовательских запросов ВЦОД, требуют решения проблем динамического перераспределения ресурсов физических машин между множеством выполняющихся виртуальных машин. Такое перераспределение может быть необходимо, как с целью балансировки загрузки виртуализированных ресурсов множества физических машин при возникновении так называемых «горячих точек» (hotspot) – резкого увеличения потребностей в ресурсах со стороны пользовательских запросов, так и с целью повышения эффективности их использования ВЦОД в целом, в частности, общего энергопотребления (перевод подмножества физических машин в неактивный режим), и/или высвобождения ресурсов такого подмножества для решения дополнительных пользовательских или служебных задач. В ряде исследований, посвященных структурно-параметрической оптимизации ВЦОД, такой процесс именуется консолидацией виртуальных машин (VM consolidation). Обобщенным показателем его эффективности является коэффициент использования физических машин: $K_F^U = \frac{N_F^{InA}}{N_F^{Full}}$ – соотношение числа неактивных физических машин ВЦОД к их общему числу. Он вычисляется по окончании времени цикла опроса физических машин и вычисления нормализованного значения показателя использования (utilization) виртуализированных ресурсов на каждой из них системой мониторинга службы администрирования ВЦОД. Базовым способом решения консолидации является динамический процесс размещения, а также переразмещения (replacement) виртуальных машин между физическими машинами. В современных ВЦОД механизмом переразмещения виртуальных машин является их миграция (VM migration), которая может быть живой (live) или отложенной. При этом в условиях роста масштаба ВЦОД, обусловленного, например, повышением потребностей в

виртуализированных ресурсах со стороны пользовательских запросов делает вопросы эффективного использования виртуализированных ресурсов ВЦОД очень актуальными. Что, в свою очередь, требует совершенствования применяемых для решения задачи переразмещения виртуальных машин моделей, методой и алгоритмов.

Существенный вклад в развитие предметной области методов и алгоритмов переразмещения виртуальных машин внесли: С.М. Алексанков, А.С. Ворожцов, Н.В. Тутова, М.А. Лореш, В.П. Соловьев, R. Buyya, M. Marzolla, E. Feller, A. Ashraf, T. Pahikkala, C. Morin, P. Liljeberg, S. Sharma, A. Esnault.

При этом большинство механизмов переразмещения виртуальных машин, реализованных в современных гипервизорах, основаны на итерационных процедурах, в которых используются варианты жадных алгоритмов, функционирующих в условиях ряда ограничений, и не в полной мере учитывающих, как многомерность виртуализированных ресурсов, так и особенности реализации конкретных механизмов живой миграции. Также проблемой существующих решений является отсутствие учета гетерогенной структуры ВЦОД – одновременного функционирования в их рамках нескольких типов гипервизоров, отличающихся, прежде всего, алгоритмами живой миграции.

Таким образом, актуальность темы диссертационного исследования связана с тем, что современные гетерогенные ВЦОД требуют исследования, разработки или модификации их математического и программного обеспечения переразмещения виртуальных машин с целью решения проблемы повышения эффективности использования физических машин, предоставляющих виртуализированные ресурсы.

Тематика диссертационной работы соответствует научному направлению ФГКВОУ ВО «Академия Федеральной службы охраны

Российской Федерации» «Повышение эффективности функционирования распределенных вычислительных систем».

Целью работы является повышение эффективности использования ресурсов гетерогенных виртуализированных ЦОД за счет разработки математического и программного обеспечения переразмещения виртуальных машин.

Задачи исследования. Для достижения поставленной цели необходимо решить следующие задачи:

1. С позиции системной методологии провести анализ проблематики процесса размещения и переразмещения виртуальных машин в гетерогенных ЦОД, включая вопросы особенности реализации алгоритмов живой миграции современных гипервизоров.
2. Разработать модель многомерного представления виртуализированных ресурсов, учитывающую нормированное значение их использования алгоритмами живой миграции и обеспечивающую возможность расчета их дисбаланса, определяющего необходимость выполнения процесса переразмещения виртуальных машин.
3. Разработать алгоритм переразмещения виртуальных машин на основе реализации метода решения задачи многомерной комбинаторной оптимизации.
4. Разработать архитектуру программного комплекса поддержки процесса переразмещения виртуальных машин в гетерогенных ЦОД.
5. Провести экспериментальное исследование по оцениванию эффективности использования ресурсов гетерогенных виртуализированных ЦОД на основе предложенного алгоритма решения задачи переразмещения виртуальных машин для различных вариантов структуры виртуализированных ЦОД и выполняемых на ней виртуальных машин.

Объект исследования – гетерогенный виртуализированный ЦОД.

Предмет исследования – модели и алгоритмы обеспечения процесса переразмещения виртуальных машин в гетерогенных ЦОД.

Методы исследования. При решении поставленных в диссертации задач использовались методы системного анализа, методы дискретной оптимизации, эвристические методы решения задач оптимизации, методы математического моделирования, математической статистики и планирования экспериментов.

Тематика работы соответствует следующим пунктам паспорта специальности 2.3.5. Математическое и программное обеспечение вычислительных систем, комплексов и компьютерных сетей (технические науки): п. 3 «Модели, методы, алгоритмы, языки и программные инструменты для организации взаимодействия программ и программных систем»; п.9 «Модели, методы, алгоритмы и программная инфраструктура для организации глобально распределенной обработки данных».

Научная новизна работы. В диссертации получены следующие результаты, характеризующиеся научной новизной:

- модель многомерного представления виртуализированных ресурсов центра обработки данных, отличающаяся дополнительным учетом нормированного значения использования ресурсов, необходимых для процесса живой миграции и позволяющая рассчитать вектор дисбаланса ресурсов для определения паросочетаний «виртуальная машина-физическая машина»;

- алгоритм процесса переразмещения виртуальных машин, основанный на метаэвристике муравьиной колонии, отличающийся учетом при расчете матрицы миграции виртуальных машин гетерогенной структуры ЦОД и дополнительных ресурсов для алгоритмов живой миграции, и обеспечивающий получение квазиоптимальной матрицы миграции виртуальных машин для существующей структуры гетерогенных ЦОД;

- архитектура программного комплекса поддержки процесса переразмещения виртуальных машин в гетерогенных ЦОД, отличающаяся реализацией механизмов встраивания в действующие программные системы.

Положения, выносимые на защиту:

- модель многомерного представления виртуализированных ресурсов центра обработки данных позволяет оценить вектор дисбаланса ресурсов для определения паросочетаний «виртуальная машина-физическая машина»;

- алгоритм процесса переразмещения виртуальных машин обеспечивает получение квазиоптимальной матрицы миграции виртуальных машин для существующей структуры гетерогенного ЦОД;

- архитектура программного комплекса поддержки процесса переразмещения виртуальных машин в гетерогенных ЦОД обеспечивает формирование плана миграции виртуальных машин, основанного на дисбалансе ресурсов физических машин.

Теоретическая значимость исследования состоит в развитии средств специального математического и программного обеспечения, учитывающих гетерогенную структуру ЦОД и влияние накладных расходов алгоритмов живой миграции виртуальных машин, которые позволяют достичь приемлемого значения коэффициента использования физических машин в сравнении с существующими решениями. Положения и выводы, содержащиеся в данной работе, могут быть использованы в развитии программных средств управления виртуализированными ЦОД.

Практическая значимость исследования Реализация разработанного алгоритма процесса переразмещения виртуальных машин в виде средств специального программного обеспечения, позволяет осуществлять его интеграцию в системы управления гетерогенных виртуализированных ЦОД. Разработанные средства специального математического обеспечения могут быть использованы в проектных и научно-исследовательских организациях,

занимающихся разработкой средств виртуализации. На программное средство получено свидетельство о государственной регистрации.

Результаты внедрения. Основные результаты внедрены в виде специального программного модуля в состав программного обеспечения административного управления IT-инфраструктурой технологической компании ООО «СОВИТ» г. Москва, а также внедрены в образовательный процесс Академии ФСО России (дисциплина – «Администрирование операционных систем»).

Апробация результатов диссертационного исследования. Основные положения диссертационной работы докладывались и обсуждались на следующих конференциях: XXIX-th International Open Science Conference «Modern informatization problems in the technological and telecommunication systems analysis and synthesis (MIP-2024'AS)» (USA, 2024), XXIX Международная научно-техническая конференция студентов, аспирантов и молодых ученых «Научная сессия ТУСУР – 2024» (ТУСУР, Томск, 2024 г.), а также на научных семинарах кафедры информатики и вычислительной техники Академии ФСО России (2022-2024 гг.).

Обоснованность и достоверность результатов диссертационного исследования обусловлены корректным использованием реализованных формальных методов исследования и подтверждены результатами сравнительного анализа данных вычислительных и натурных экспериментов.

Публикации. По результатам диссертационного исследования опубликовано 6 научных работ, в том числе 4 – в изданиях, рекомендованных ВАК РФ (из них 1 – в изданиях WoS и одно свидетельство о регистрации программы для ЭВМ). В работах, опубликованных в соавторстве и приведенных в конце автореферата, лично автором получены следующие результаты: [78,54] – аналитическая модель многомерного представления виртуализированных ВКР, [116,118] – алгоритм переразмещения виртуальных машин, основанный на метаэвристике муравьиной колонии, [117,20] –

архитектура программного комплекса поддержки процесса переразмещения ВМ в гетерогенных ЦОД.

Объем и структура работы. Диссертация состоит из введения, четырех глав, заключения и приложений. Объем работы – 171 страница, включая 46 рисунков, 5 таблиц. Список литературы содержит 129 наименований.

ГЛАВА 1. ИССЛЕДОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ ВИРТУАЛИЗИРОВАННЫХ ЦЕНТРОВ ОБРАБОТКИ ДАННЫХ И СУЩЕСТВУЮЩИХ ПРОБЛЕМ УПРАВЛЕНИЯ ИХ ВИРТУАЛИЗИРОВАННЫМИ РЕСУРСАМИ

1.1 Анализ предметной области виртуализированных центров обработки данных

Развитие и совершенствование современных технологий обработки больших объемов данных (обобщенное название предметной области – big data) [1,2] неразрывно связано с развитием и совершенствованием вычислительных инфраструктур, обеспечивающих поддержку этих технологий. Сложность методов и алгоритмов таких предметных областей обработки больших данных, как интеллектуальный анализ данных (data mining) [3], распределенные вычисления (distributed computing) [4], а также, получившие в последнее время, широкое распространение системы искусственного интеллекта (artificial intelligence) [5] требует соответствующей распределенной, высокопроизводительной, высоконагруженной и динамически масштабируемой вычислительной инфраструктуры.

В настоящее время общепринятым подходом к разработке подобной инфраструктуры является создание ЦОД [7,8] – взаимосвязанных в рамках единой структуры промышленного масштаба вычислительных систем, а также поддерживающих их телекоммуникационных систем, систем хранения данных и сервисных систем, связанных с электропитанием, охлаждением, пожаробезопасностью и информационной безопасностью.

Так, например, поддержка процесса обучения и эксплуатации системы искусственного интеллекта (ИИ) Google Gemini [10] реализована на базе распределенной системы ЦОД компании Google, состоящей из шести взаимосвязанных ЦОД (4 – Северная Америка, 1 – Европа, 1 – Юго-Восточная Азия), поддерживающих специализированную структуру фреймворка Google

Cloud [11], базирующуюся на технологии multicost [12] – внутрекластерной и межкластерной взаимосвязи многопроцессорных систем акселераторов ИИ (TPU - Tensor Processing Unit), именуемых TPU SuperPods (сборка из 4096 TPU), объединяемых телекоммуникационной системой Google Jupiter [13], которая базируется на оптической коммуникационной сети OCS [13].

Известная своими передовыми разработками в области генеративного ИИ компания OpenAI [14], использует систему ЦОД компании Microsoft – Azure Global Infrastructure, поддерживающую более 300 ЦОД в 60 регионах [15].

Вычислительная инфраструктура Yandex Cloud базируется на распределенной системе из четырех ЦОД в Московской области, по одному в Рязанской и Владимирской областях, а также ЦОД в Финляндии и Казахстане [16].

При этом, рост и совершенствование распределенных систем ЦОД связаны не только с развитием их аппаратной вычислительной и телекоммуникационной составляющих, но также и с совершенствованием архитектурных, а также программных и программно-аппаратных решений. В области архитектурных решений превалирующей концепцией архитектуры современных ЦОД является концепция облачных вычислений (cloud computing) [16], а в области программных/программно-аппаратных решений базовой является технология виртуализации вычислительных и коммуникационных ресурсов ЦОД.

1.1.1. Обобщенное представление виртуализированных центров обработки данных

Концепция облачных вычислений – вид распределенной обработки и хранения данных, имеющая целью оперативного предоставления потребителям по требованию безопасных, качественных, масштабируемых и автоматически конфигурируемых ИТ-услуг на основе модели оплаты по мере использования (pay-as-you-go model) [18]. Национальный институт стандартов и технологий США (NIST - National Institute of Standards and Technology) определяет облачные вычисления, как совокупность трех моделей услуг [19]:

1. SaaS (Software as a Service) – программное обеспечение как услуга.

2. PaaS (Platform as a Service) – платформа как услуга.

3. IaaS (Infrastructure as a Service) – инфраструктура как услуга.

Реализация указанных типов моделей осуществляется в рамках ЦОД, которым владеет или который арендует поставщик услуг. В настоящее время, в рамках совершенствования процессов организации и функционирования ЦОД в качестве основной технологии управления его вычислительными и коммуникационными ресурсами (далее ВКР) используется технология их виртуализации [20]. Подобный тип ЦОД называется виртуализированный (виртуальный) или программно-определяемый ЦОД (SDDC – software defined data center) [21]. Согласно [21] ВЦОД – это программная надстройка в рамках физического ЦОД, представляющая собой совокупность виртуализированных ВКР, таких как серверы, системы хранения данных (СХД), сетевые компоненты (телекоммуникационная инфраструктура), а также система управления ими, которые развертываются и управляются как единое целое. ВЦОД имеет такую же функциональность, как и физический ЦОД, обладая при этом дополнительными преимуществами, предоставляемыми технологией виртуализации [21]. Фактически, ВЦОД, используя технологию виртуализации, формирует абстрактное представление физического ЦОД. При этом вся инфраструктура физического ЦОД, а именно: процессоры (процессорные ядра), оперативная память, системы хранения данных, сетевое оборудование, средства автоматизации и обеспечения информационной безопасности представляются в ВЦОД, как набор (пул) ВКР.

Как указано в [22], в общем случае, концепция ВЦОД образуется на основе обобщенного подхода к виртуализации вычислительных ресурсов и уже достаточно хорошо сформировавшихся концепций программно-определяемых сетей (SDN) [23] и программно-определяемых СХД (SDS) [24].

Как и в случае SDN и SDS, подобный подход позволяет представить виртуализированный ЦОД, в виде платформ «ИТ как Услуга» (ITaaS) [25].

В ряде источников подобное представление детализируется в зависимости от функционального назначения ВЦОД. Так в [27] компания «Softline» представляет

его в виде услуги «ЦОД как Инфраструктурная Услуга» (SDDC as a IaaS), обеспечивая потребителям инфраструктуру, которая будет соответствовать требованиям закона ФЗ-152 [28]. В [28] компания «АйТеко» представляет ВЦОД, как услугу «ЦОД как Услуга» (DCaaS – Data Center as a Service), которая предлагается потребителям для оптимизации издержек на поддержку корпоративной ИТ-инфраструктуры, для оперативного развертывания ИТ-конфигураций с целью тестирования новых процессов, а также формирования резервной среды хранения корпоративных данных. Компания HPE (Hewlett Packard Enterprise) в [29] представляет виртуализированный ЦОД как услугу «Объекты/Оборудование/Средства ЦОД как Услуга» (FaaS – Facilities as a Service). Потребителю предлагается модульный виртуализированный ЦОД, который и функционирует только в режиме операционных расходов потребителя (OPEX - Operation Expenditure). Схематично услуга FaaS представлена на рисунке 1.1.

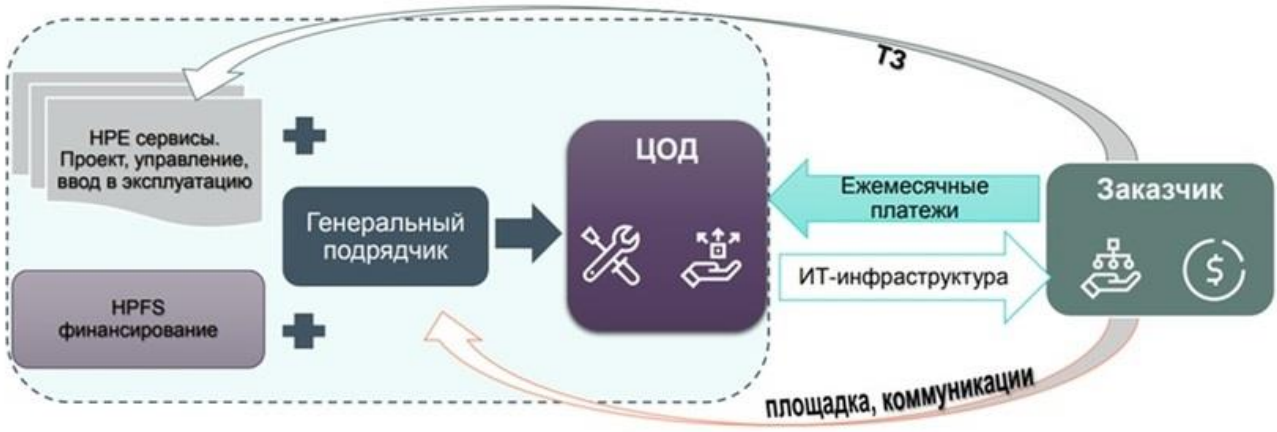


Рисунок 1.1 – Обобщенное представление виртуализированного ЦОД в виде услуги FaaS (источник компания HPE)

Очевидно, что подобные представления ВЦОД имеют много общего с технологией облачных вычислений, реализующей концепцию предоставления услуг различного типа. Однако, технология виртуализации, лежащая в основе ВЦОД и технология облачных вычислений имеют существенные различия: технология виртуализации позволяет на базе одной физической аппаратной

вычислительной и/или коммуникационной платформы сформировать несколько вычислительных (коммуникационных) сред, использующих только часть аппаратных ВКР. Технология же облачных вычислений – это платформа, которая, используя телекоммуникационную систему, предоставляет удаленный доступ потребителя к набору аппаратных ВКР или программных услуг, реализуемых на его основе, обеспечивая, по мере необходимости, масштабирование этого набора.

В [30,31] делается обобщение состава компонентов ВЦОД. В общем виде они представлены на рисунке 1.2.

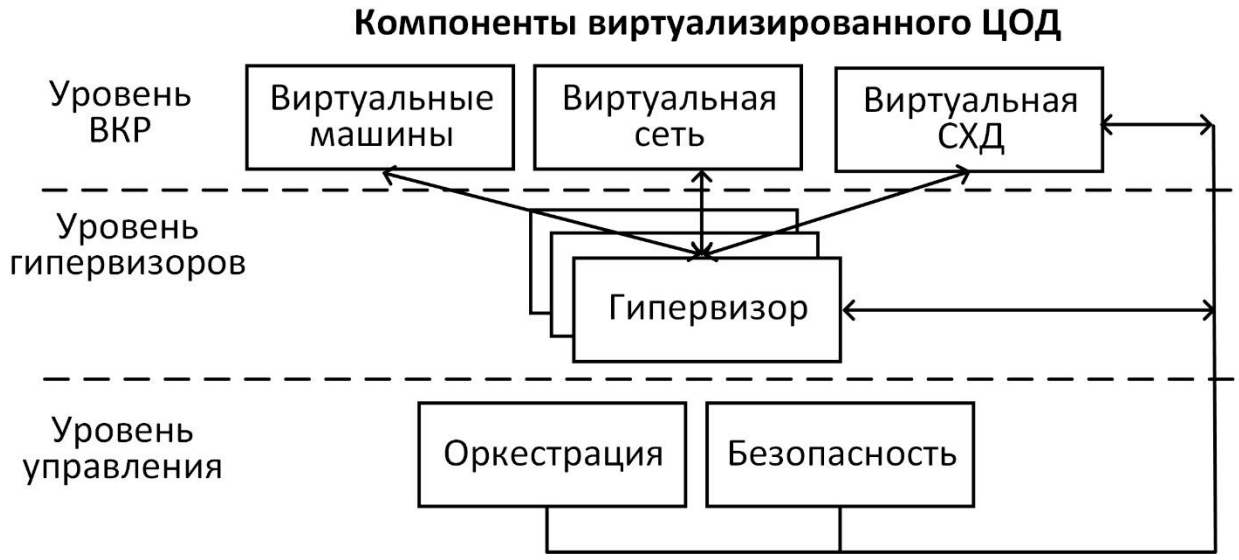


Рисунок 1.2 – Обобщенное представление структурных компонентов виртуализированного ЦОД

Из рисунка 1.2 видно, что к компонентам ВЦОД относят:

- виртуальные машины (ВМ) – программно-эмулируемые физические машины (ФМ), использующие часть ВКР ФМ. На базе ВМ, как и ФМ устанавливается операционная система (ОС) и набор требуемого системного и прикладного программного обеспечения (ПО). Обычно, в зависимости от вида технологии виртуализации [32] к ВКР ВМ относят: виртуализированные процессоры (CPU) или процессорные ядра (CPU Core) (обычно именуются VCPU) и выделенный объем оперативной памяти ФМ.

- виртуальная сеть – это набор аппаратного или программно-определяемого сетевого оборудования, обеспечивающего ВМ средствами обмена данными с

другими ВМ, ФМ внутри виртуализированного ЦОД или находящихся во внешних сетях. Обычно виртуальная сеть включает в себя: виртуальные коммутаторы, виртуальные маршрутизаторы, систему управления сетевой инфраструктурой, систему сетевого мониторинга.

- виртуальная СХД – программно-определяемый набор СХД множества ФМ и/или отдельно реализуемой СХД на основе технологий NAS или SAN [33], используемый множеством ВМ, в качестве собственных или совместно используемых СХД. Наряду с гибким распределением аппаратного ресурса СХД, она обеспечивает такие функции, как создание моментальных снимков (snapshot) ВМ, а также репликацию и дедупликацию системных и пользовательских данных.

- гипервизор или монитор ВМ (VMM – Virtual Machine Monitor) – это программный модуль управления, который позволяет нескольким ВМ выполняться на ВКР одной ФМ, обеспечивая их эффективное использование [34]. Он скрывает детали физических ВКР и предоставляет виртуализированные ВКР для ПО более высокого уровня. Кроме того, он обеспечивает нескольким ВМ совместно использовать ВКР ФМ.

Примерами гипервизоров серверных платформ (т.н. гипервизоры первого типа) являются: Xen VMM (разработчик Компьютерная лаборатория Университета Кембриджа) [35], KVM Hypervisor (разработчик KVM Community) [36], Microsoft Hyper-V (разработчик компания Microsoft) [37], ESXi (разработчик компания VMware (принадлежит компании Qualcomm) [38]. В рамках ВЦОД может функционировать несколько гипервизоров разных производителей. Подобный ВЦОД именуется гетерогенным. В случае применения только одного типа гипервизора речь идет о гомогенном ВЦОД. Оба варианта имеют свои достоинства и недостатки. Чаще всего гетерогенные ВЦОД формируются на основе уже существующих решений, например, в рамках уже развернутых корпоративных облачных платформ. Гомогенные ВЦОД обычно предлагаются разработчиком конкретного гипервизора, либо формируются на основе единого проекта.

- уровень управления ВЦОД и оркестровки ВКР – это программные модули управления для автоматизации подготовки, настройки и мониторинга

вышерассмотренных компонентов ВЦОД. Он предназначен для управления жизненным циклом ВМ, оптимизации использование ВКР ЦОД в целом и обеспечения их высокой доступности.

- уровень обеспечения безопасности – это программные и программно-аппаратные модули, такие как виртуальные межсетевые экраны, системы обнаружения и предотвращения вторжений (IDPS) и механизмы разграничения доступа. На этом уровне решается задача контролируемого разграничения доступа к компонентам ВЦОД, а также предотвращение несанкционированного доступа к ним.

Пример реализации указанных уровней ВЦОД на основе гомогенного решения, использующего продукты VMware (принадлежит компании Broadcom), [39] представлен на рисунке 1.3.

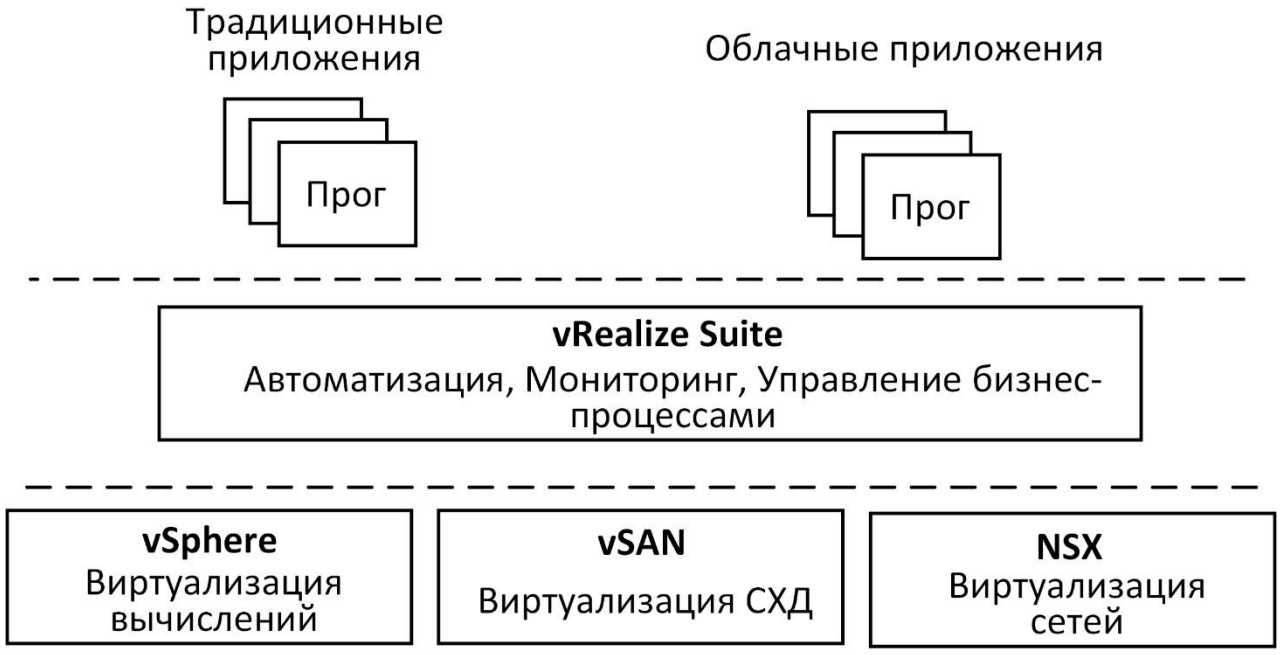


Рисунок 1.3 – Пример реализации гомогенного виртуализированного ЦОД на основе продуктов VMware

К основным преимуществам ВЦОД в [22,40,41] относят:

- простоту управления ВКР, приложениями и пользовательскими запросами, что достигается использованием ежиной платформы управления и мониторинга состояния ЦОД;

- уменьшение временных затрат на развертывание системного и прикладного ПО, что достигается использованием единых политик управления, а также применением технологий создания моментальных снимков (snapshots) состояния ВМ;

- повышение эффективности использования ВКР ЦОД за счет процессов автоматизации (использование шаблонов развертывания вычислительных и сетевых инфраструктур), а также оркестрации (динамическое управление нагрузкой);

- независимость от выбора поставщика облачных услуг на базе традиционного ЦОД. Связана, как с возможностью быстрого переноса структуры виртуализированного ЦОД на ВКР более экономически выгодного поставщика, так и возможность ее развертывания одновременно на ВКР нескольких поставщиков;

- обеспечение безопасности на основе, как единых политик, предоставляемых поставщиком облачных услуг, так и собственных политик, реализуемых внутри виртуальной инфраструктуры.

Пользователи ВЦОД получают следующие преимущества [41]:

- гибкость оплаты по факту использования виртуализированных ВКР;

- масштабируемость для удобного добавления или удаления виртуализированных ВКР по мере необходимости;

- повышенные возможности аварийного восстановления данных и функциональности ПО;

- возможность недорого развертывания нескольких виртуальных инфраструктур с целью тестирования эффективности их использования в реальных бизнес-процессах.

В [42] приводится пример перевода ИТ-инфраструктуры компании Carrier Global (специализируется на интеллектуальных решениях в области климата и энергетики, предлагает широкий спектр продуктов и услуг в области систем

отопления, вентиляции и кондиционирования, охлаждения, пожарной безопасности, безопасности и автоматизации зданий). Компания преобразовала свою ИТ-инфраструктуру за счет использовании виртуальных центров обработки данных и сервисов миграции в облако от Amazon Web Services (AWS) [42]. Целью такого преобразования был перенос в течение 3 лет более 4000 единиц рабочей нагрузки в ВЦОД. В ходе этого преобразования компания:

- свернула использование 1 300 аппаратных серверов, поддерживающих 350 приложений, и вывела из эксплуатации 370 аппаратных серверов;

- закрыла 14 традиционных ЦОД;

- привела к единому стандарту сетевую инфраструктуру, объединяющую 675 филиалов.

Указанные выше преимущества определяют высокий рост рынка ВЦОД, обусловленный, в частности, развитием и совершенствованием технологий ИИ, требующих достаточно быстрого реагирования на растущие запросы пользователей. В этом смысле возможности динамического масштабирования виртуализированных ВКР, а также быстрого развертывания готовых вычислительных инфраструктур для обучения нейронных сетей различного типа, значительно превосходят возможности традиционных ЦОД, как по показателям оперативности, так и по показателям ресурсоемкости.

В [43,44] приводятся аналитические обзоры рынка ВЦОД, отражающие прогнозы развития прибыли основных поставщиков этого решения, к которым относятся компании: VMware, Microsoft, Citrix Systems, Amazon Web Services, Cisco Systems, AT&T, Fujitsu, Radiant Communications, HPE, Huawei, HCL, IBM.

На рисунке 1.4 представлена тенденция роста прибыли мирового рынка ВЦОД на период до 2032 года.

Как указывается в [43] ключевым фактором такого роста является растущий спрос на унифицированное и централизованное администрирование ЦОД. Унифицированное и централизованное администрирование ЦОД позволяет организациям управлять своей виртуализированной ИТ-инфраструктурой, используя единый интерфейс. Такой подход обеспечивает всестороннюю

видимость виртуализированной инфраструктуры и позволяет быстрее устранять неполадки и оптимизировать производительность. Кроме того, факторами роста являются: растущий спрос на более эффективное использование ВКР и источников данных (что характерно для отрасли ИИ), централизованное управление безопасностью и соблюдение нормативных требований, а также общее снижение операционных затрат на развертывание и эксплуатацию ЦОД.

Рынок российских решений ВЦОД имеет специфические особенности, связанные, в первую очередь с решением проблемы импортозамещения серверных технологий, а также телекоммуникационной инфраструктуры, являющихся основой традиционных ЦОД. В подобных условиях основными направлениями развития этого рынка являются технологии VDI (Virtual Desktop Infrastructure - удаленный рабочий стол) и контейнеризации приложений и сервисов [44].

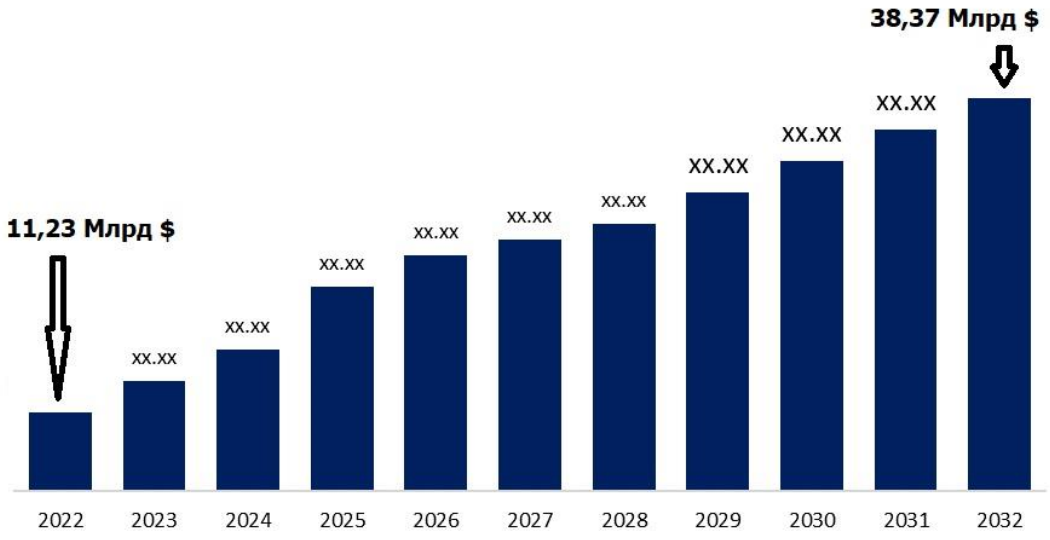


Рисунок 1.4 – Диаграмма роста операционной прибыли рынка виртуализированных ЦОД на период до 2032 года (источник Sphericals Insinghts)

В аналитических отчетах группы компаний «Астра» [45] и компании «Базис» (Basys) [46] отмечается, что в 2023 году рынок виртуализации в целом вырос на 15-20% по сравнению с предыдущим периодом, и оценивается в 10 млрд руб. против 8 млрд руб. в 2022 году, а серверной виртуализации, в частности, в 6 млрд рублей. Рынок инфраструктурных сервисов (серверная виртуализация, контейнерная

виртуализация, VDI, SDS, SDN и др.) в целом вырос до 57 млрд рублей против 46 млрд руб. в 2022 году.

На рисунке 1.5 приведена диаграмма роста прибыли ведущих российских компаний, связанных с рынком виртуализации в течение 2023 года.

Очевидно, что вне зависимости от представлений ВЦОД, рассмотренных выше, основным фактором, определяющим разработку и совершенствования такого вида ЦОД является экономическая составляющая.

В [47,48] она выражается показателем «совокупная стоимость владения» (total cost of ownership, TCO) – величина целевых затрат, которые несет владелец с момента начала вступления во владение до момента выхода из владения и исполнения владельцем полного объема обязательств, связанных с владением [49]. Обобщенное представление структурных элементов, формирующих TCO ЦОД, представлена на рисунке 1 [50]. На рисунке 1.6 красным цветом выделены структурные элементы ЦОД, рассматриваемые в рамках исследования.

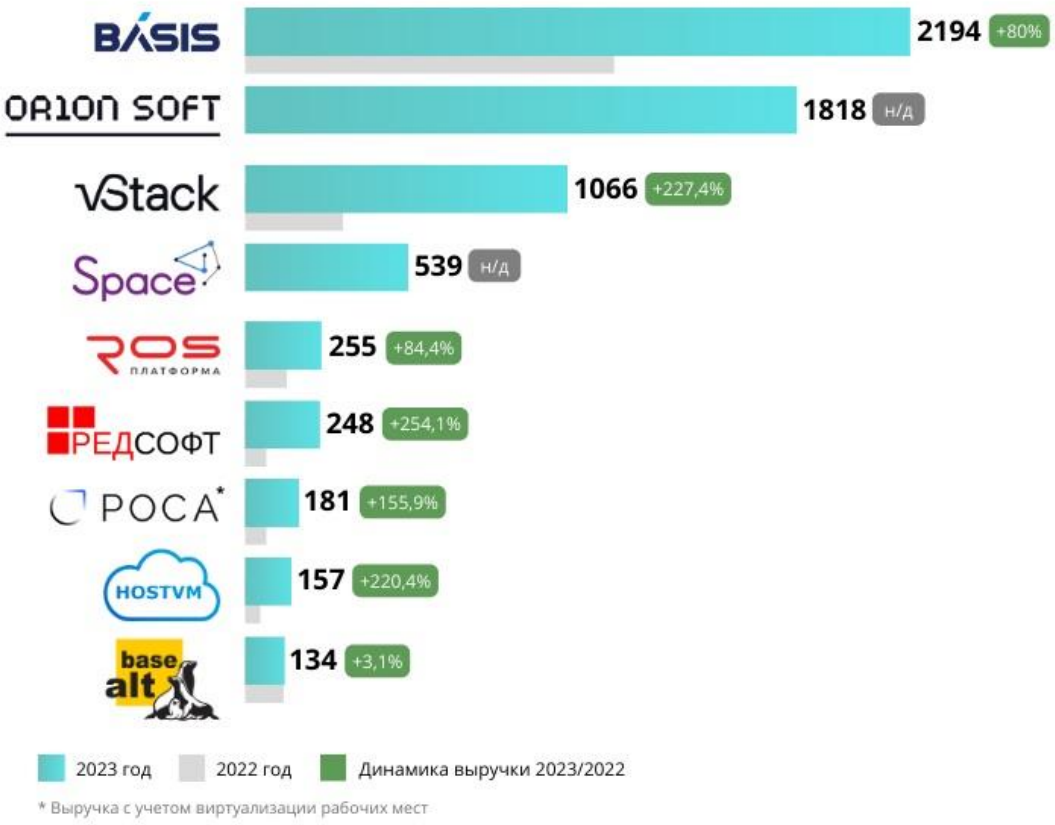


Рисунок 1.5 – Диаграмма роста операционной прибыли российских компаний производителей средств виртуализации за 2023 год (в млн. рублей)

Как видно, к ним относятся, как элементы капитальных затрат, формирующие ВКР, так и элементы операционных затрат, связанные с процессом миграции различного рода данных.

Показатель ТСО является комплексным. Ряд исследований, связанных с оцениванием эффективности функционирования ЦОД по этому показателю, рассматривают два базовых направления [51,52,53]:

1. Энергоэффективность реализуемого ЦОД процесса выделения и эксплуатации ВКР, связанная с совокупным потреблением энергии серверными платформами, системами хранения данных и телекоммуникационными системами ЦОД.

Рисунок 1.6 – Структурные элементы, формирующие ТСО ЦОД

2. Эффективность использования базовых ВКР серверных платформ ЦОД, в первую очередь таких как:

- утилизация (коэффициент использования) процессоров (процессорных ядер) серверной платформы (CPU demand);

- утилизация (коэффициент использования) оперативной памяти серверной платформы ЦОД (Memory demand).

С точки зрения эксплуатации ВЦОД именно эти коэффициенты использования играют определяющую роль в процессах оптимизации его структуры для обеспечения сокращения операционных расходов.

Для решения таких оптимизационных задач требуется дать обобщенное представление виртуализированных ВКР.

1.1.2. Обобщенное представление виртуализации вычислительных и телекоммуникационных ресурсов ЦОД

В [54] под виртуализацией ВКР понимается формирование и выделение требуемых пользователям вычислительных ресурсов не на основе аппаратной составляющей ЦОД, а с использованием технологий серверной виртуализации. Подобный подход обеспечивает решение комплекса задач, связанных с балансировкой нагрузки поступающей на отдельные узлы распределенной схемы ЦОД, в зависимости от интенсивности пользовательских запросов на использование ресурсов в различные временные промежутки и в различных территориальных точках размещения узлов ЦОД. Кроме того, технология виртуализации аппаратных вычислительных ресурсов обеспечивает решение проблемы эффективного использования вычислительных ресурсов аппаратных серверов, составляющих технологическую основу ЦОД.

В общем виде подход предоставления пользовательских услуг с использованием технологии виртуализации представлен на рисунке 1.7.

Из рисунка 1.7 видно, что базовой концепцией технологии виртуализации ВКР является отделение уровня ПО: $ПО_1 - ПО_m$, эксплуатируемого множеством пользователей ЦОД, от уровня ФМ: $ФМ_1 - ФМ_k$ серверных платформ и телекоммуникационной составляющей ЦОД ($ТКС_{ЦОД}$) путем формирования на основе этих физических ВКР множества ВМ: $ВМ_1 - ВМ_n$, а также виртуализированной телекоммуникационной структуры (ВТКС). При этом каждая

ВМ содержит полный стек системного и промежуточного ПО и обеспечивается заданными в ее первоначальной конфигурации виртуализированными ВКР (в первую очередь процессорными ядрами и требуемым объемом оперативной памяти). Очевидно, в общем случае мощности множеств ПО, ФМ и ВМ не равны. Функции формирования уровня виртуализированных ВКР выполняет гипервизор $ГВ_{ЦОД}$, относящийся, как правило, к классу гипервизоров первого типа [55].

Также из рисунка 1.7 видно, что множество $ПО_1 - ПО_m$ следует рассматривать, как совокупность подмножеств $М_{ПО}$, элементами которых являются отдельные модули $ПО_1 - ПО_m$, требующие независимого выделения ВКР, взаимосвязи которых определяют логику функционирования $ПО_1 - ПО_m$. При этом, в зависимости от текущего значения свободных и занятых виртуализированных ВКР множества $ВМ_1 - ВМ_n$ для запуска разных модулей $М_{ПОm}$, принадлежащих одному и тому же $ПО_m$, могут использоваться разные ВМ.

Как и в случае эксплуатации ПО на физической структуре ЦОД, множество пользователей $Пол_1 - Пол_m$ предъявляют соответствующие требования к качеству предоставляемой им услуги, реализуемой на виртуализированных ВКР. Эти требования определяются в рамках соглашения SLA (Service Level Agreement – соглашение об уровне сервиса) между пользователями и провайдером ЦОД. Обычно, в качестве показателей качества предоставляемой услуги рассматриваются: время отклика ПО или его отдельных модулей, а также пропускная способность ВТКС).

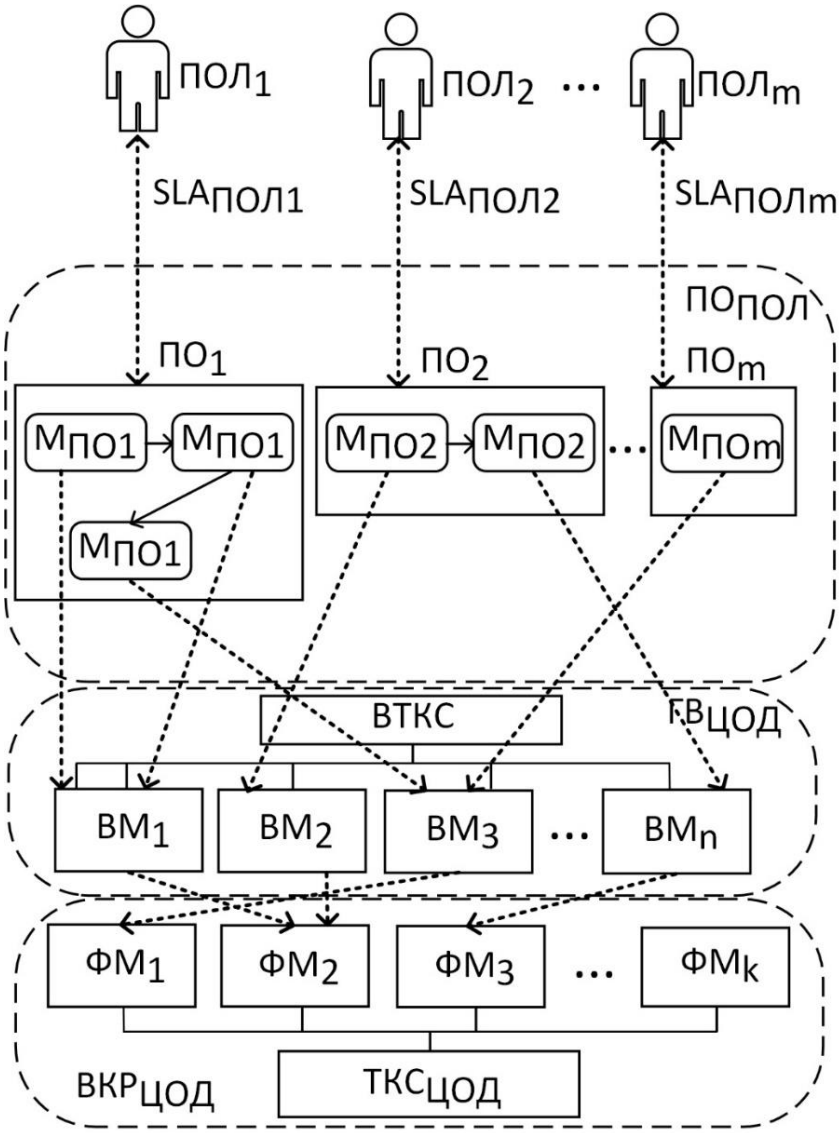


Рисунок 1.7 – Принцип динамического распределения ВКР ЦОД на основе технологии их виртуализации

Примеры конкретной реализации обобщенной схемы виртуализации ВКР ЦОД (рисунок 1.7) представлены для гипервизора первого типа KVM [36] (рисунок 1.8) и гипервизора типа 1+ Microsoft Hyper-V [37] (рисунок 1.9).

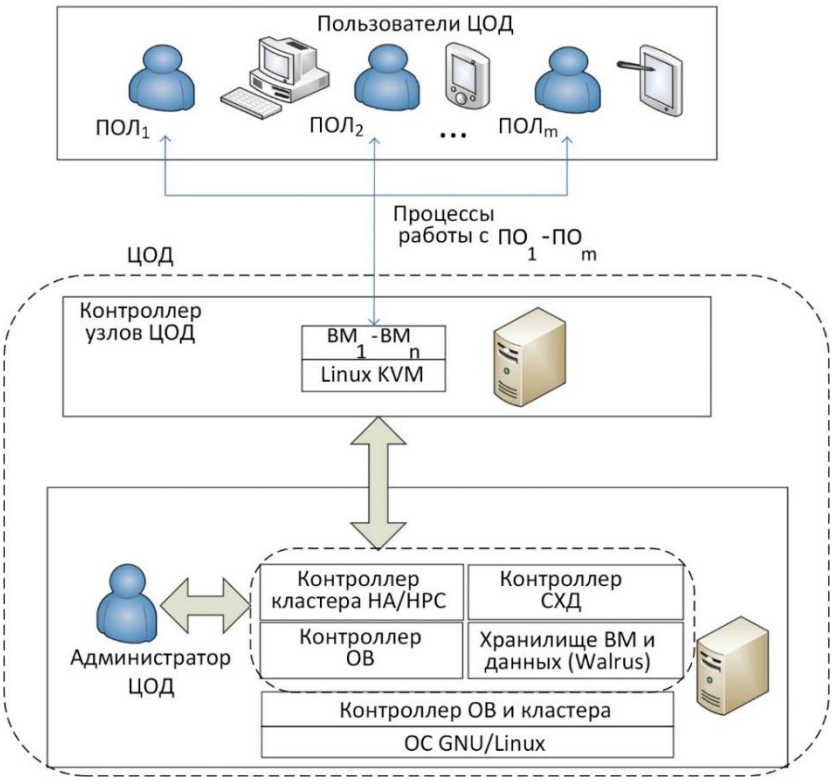


Рисунок 1.8 – Схема реализации технологии виртуализации ВКР ЦОД с использованием гипервизора Linux KVM

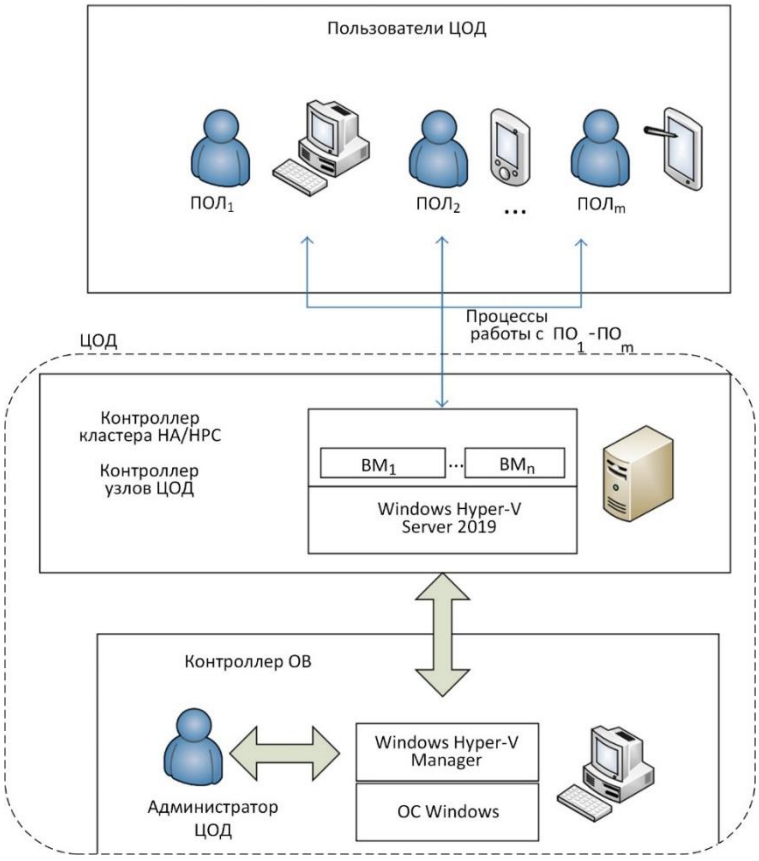


Рисунок 1.9 – Схема реализации технологии виртуализации ВКР ЦОД с использованием гипервизора Microsoft Hyper-V

Таким образом, реализация услуг с использованием технологии виртуализации представляется многоуровневой структурой, взаимосвязи компонентов каждого уровня в которой (пунктирные стрелки на рисунке 1.7) формируются динамически, в зависимости от множества внешних и внутренних факторов.

Очевидно, что в рамках обобщенной проблемы управления подобными виртуализированными ВКР появляется множество частных проблем, связанных с их использованием по разным критериям эффективности процессов формирования и предоставления услуг пользователям ЦОД.

Запуск, приостановка и завершение выполнения пользователями о ПО или его отдельных модулей может существенно влиять на увеличение или уменьшение значений отдельных виртуализированных ВКР, что, в свою очередь, может сказываться на изменении требований ВМ к значениям аппаратных ВКР.

В связи с этим актуальными являются исследовательские задачи, обеспечивающие комплексную поддержку рассмотренных выше методологических основ функционирования ЦОД, то есть решение проблем динамического переразмещения (replacement) множества ВМ с разными потребностями в виртуализированных ВКР между множеством ФМ (узлов) ЦОД в зависимости от особенностей текущего или прогнозируемого потока пользовательских запросов и/или условий соглашения о качестве их обслуживания SLA.

1.2 Исследование проблемы переразмещения виртуальных машин в виртуализированных центрах обработки данных

В [56] дается представление процесса переразмещения ВМ, этапы которого обобщенно представлены на рисунке 1.10.

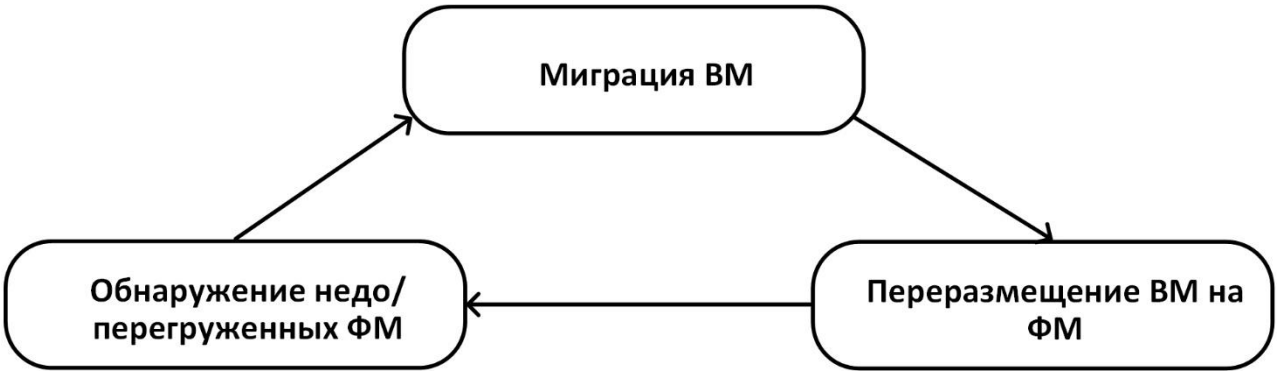


Рисунок 1.10 – Обобщенное представление процесса переразмещения (replacement) ВМ

В [58] представленная на рисунке 1.10 схема детализируется с рассмотрением двух этапов функционирования системы управления ВЦОД:

1. Этапа планирования (статическое размещение множества ВМ на множестве ФМ).

2. Этапа эксплуатации (статическое/динамическое переразмещение подмножества ВМ на подмножестве ФМ).

Схема, детализирующая указанные этапы представлена на рисунке 1.11.

Рисунок 1.11 – Детализация этапов процесса переразмещения (replacement) ВМ

Из рисунка 1.11 видно, что процесс переразмещения ВМ состоит из этапов:

- определения требований каждой ВМ к ВКР на начальном этапе ее функционирования;

- конфигурирование ВМ в соответствии с требованиями текущей рабочей нагрузки (VM Sizing);

- начальное размещение (placement) множества сконфигурированных ВМ на множестве ФМ (VM-PM Mapping).

Указанные этапы реализуются в ходе планирования конфигурации виртуализированного ЦОД.

В ходе его эксплуатации рассматриваются следующие этапы:

- реконфигурирование ВМ в соответствии с изменением рабочей нагрузки (Workload VM Resizing);

- мониторинг использования ВКР множества ФМ множеством реконфигурированных ВМ с целью для обнаружения горячих точек (hotspot) (ФМ перегруженные и недогруженные с точки зрения ВКР);

- переназначение подмножества ВМ на подмножество ФМ (VM-PM Remapping);

- миграция подмножества ВМ на подмножество ФМ в соответствии с планом переразмещения.

Из рассмотренного выше можно выделить следующие проблемы эффективного процесса переразмещения ВМ:

- обнаружение ФМ, недогруженных с точки зрения ВКР: требуется для принятия решения о возможности переноса всех ВМ, выполняющихся на них и их перевод в режим пониженного энергопотребления (задача минимизации количество активных ФМ);

- обнаружение ФМ, недогруженных с точки зрения ВКР: требуется для принятия решения о возможности переноса подмножества ВМ с них на подмножество других активных ФМ или повторно активированных ФМ (задача обеспечения QoS множества пользовательских запросов);

- миграция ВМ: выполнение процесса миграции ВМ с минимальным временем простоя и потреблением ВКР во время процесса миграции;

Большинство исследований, посвященных проблеме переразмещения ВМ [57,58,59], ориентированы на рассмотрение проблем обнаружения недо/перегруженных ФМ с точки зрения ВКР, не в полной мере учитывая при этом особенности процесса их миграции, которые оказывают влияние на эффективность процессе переразмещения ВМ.

1.3 Исследование методов и алгоритмов миграции виртуальных машин

Как было рассмотрено на рисунке 1.11, при обнаружении недо/перегруженных ФМ (этапы 5 и 6) одна или подмножество ВМ должны мигрировать между ФМ. Как было указано в п. 1.2, одной из целей этой миграции является необходимость устранения нарушения SLA [61].

В [62] подробно рассмотрены виды миграции ВМ, включая их критерии и целевое назначение. В обобщенном виде эти виды представлены на рисунке 1.12.

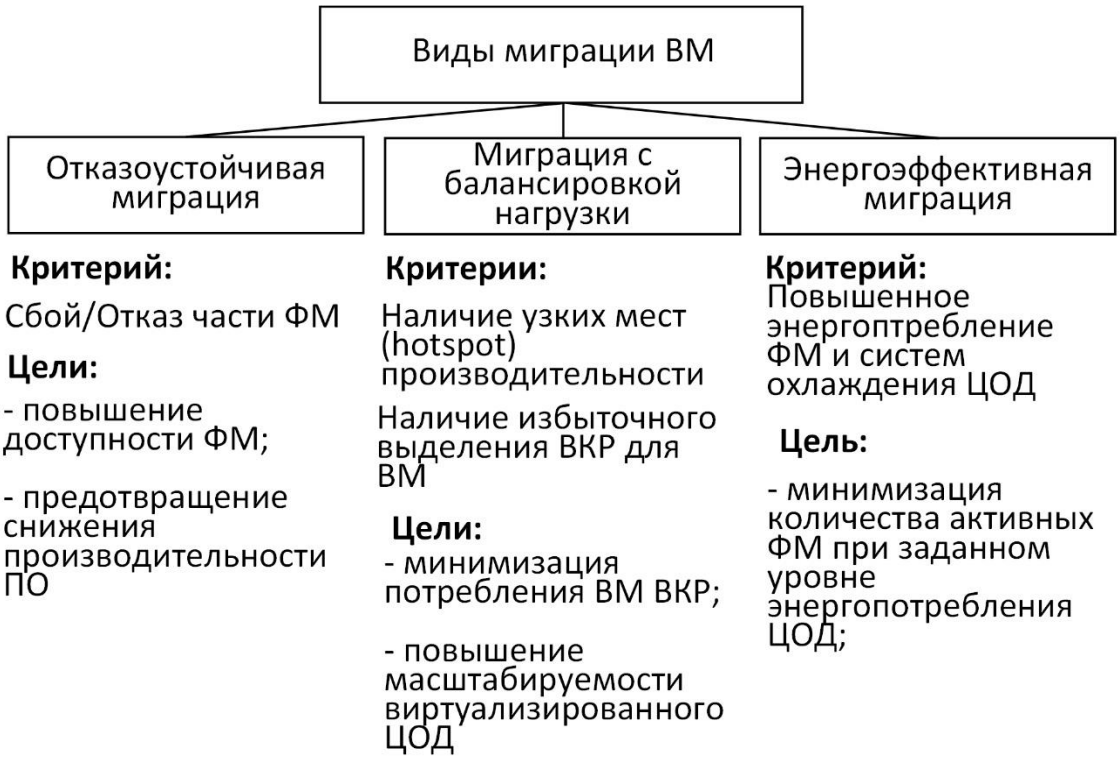


Рисунок 1.12 – Виды миграции ВМ, их критерии и целевое назначение

Очевидно, что вне зависимости от вида миграции этот процесс сводится к передаче состояния виртуализированных ВКР ВМ между исходной и целевой ФМ. К состоянию виртуализированных ВКР в [62, 64] относят:

- состояние процессора и/или процессорных ядер (VCPU State);
- содержимое оперативной памяти (Memory content);
- содержимое хранилищ виртуальных дисков (Storage content).

Как показано в [7] современные ВЦОД поддерживают единое хранилище виртуальных дисков, доступное через телекоммуникационную систему ЦОД в режиме «чтение-запись» для всего множества ВМ, размещенных на различных ФМ.

Состояние VCPU State фактически определяет контекст регистров процессора (процессорного ядра), что не является критичным с точки зрения передачи между исходной и целевой ФМ.

Наиболее критичным с точки зрения оперативности процесса переразмещения ВМ является содержимое страниц (pages) ее оперативной памяти (Memory content).

В общем виде процесс миграции ВМ представлен на рисунке 1.13.

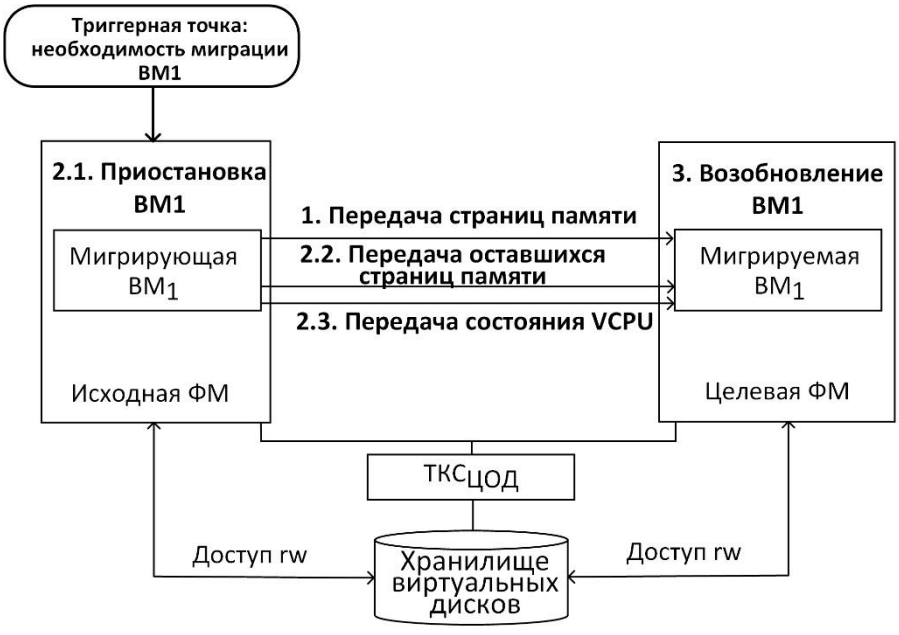


Рисунок 1.13 – Обобщенное представление процесса миграции ВМ

В рамках представленного на рисунке 1.13 процесса миграции ВМ в [63,64] его классифицируют на:

- не живую миграцию (VM Non-Live Migration);

- живую (или динамическую) миграцию (ЖМ) (VM Live Migration).

Не живая миграция выполняется в автономном режиме. Это означает, что ВМ должны быть приостановлены на исходной ФМ и состояние их виртуализированных ВКР должно быть скопировано на целевой ФМ, после чего ВМ могут быть возобновлены.

Для передачи состояния Memory Content не живая миграция использует метод остановки и копирования (stop-and-copy), при котором выполнение ВМ должно быть приостановлено для копирования ее страниц виртуализированной оперативной памяти на целевую ФМ (этап 1 на рисунке 1.13), и ВМ может быть возобновлена только после выполнения этой операции копирования.

В [65] под ЖМ ВМ понимается перенос состояния ее виртуализированных ВКР между аппаратными ВКР множества ФМ, расположенных либо внутри одного узла ЦОД, либо на территориальной распределенных узлах ЦОД.

Существующие технологии ЖМ ВМ в основном различаются особенностями передачи состояния Memory Content, которые базируются на концепции организации памяти современных операционных систем (ОС) серверных платформ, поддерживающих программно-аппаратную технологию страничного распределения памяти (Page Memory) [66, 67]. Согласно этой концепции, ресурс Memory content представлен двумя множествами страниц памяти:

1. Множеством свободных страниц F (от Free).

2. Множеством занятых («грязных») страниц D (от Dirty).

Под занятой («грязной») страницей памяти понимается страница, в которую записана часть контекста $C_{ПО}$, выполняемого ПО или его отдельных модулей:

- код ПО (модулей);

- код библиотек, необходимых для функционирования кода ПО;

- данные конфигурации ПО (модулей, библиотек);

- данные, обрабатываемые ПО (модулями, библиотеками).

Очевидно, что мощность множества $|D|$ является переменной величиной, значение которой зависит от:

- текущего значения мощности множества $|F|$ в момент времени загрузки ПО (модулей, библиотек) в память ФМ;

- значения контекста $C_{ПО}$ в момент времени загрузки ПО (модулей, библиотек) в память;

- значений динамически подгружаемого контекста $C_{ПО}^{t_n}$ в моменты времени t_n.

В обобщенном виде принцип ЖМ ВМ представлен на рисунке 1.14.

Из рисунка 1.14 видно, что в процессе реализации ЖМ используются два типа ВМ:

1. Мигрируемая (исходная) ВМ, состояние которой требует переноса (миграции).

2. Целевая ВМ – расположенная на целевой ФМ, которая при запуске будет реализовывать функции мигрируемой ВМ.

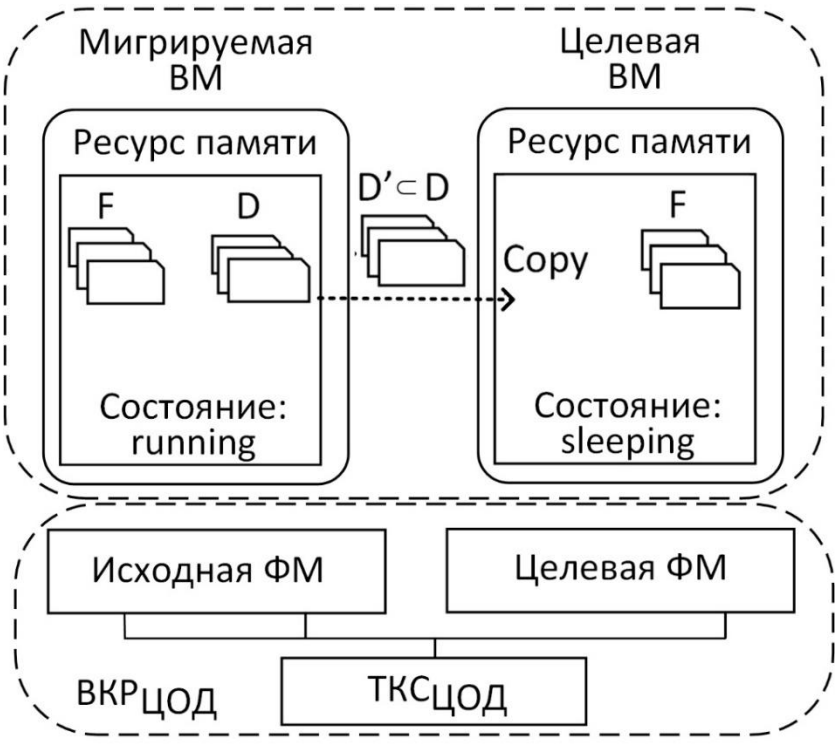


Рисунок 1.14 – Обобщенное представление технологии живой миграции ВМ

Очевидно, что состояние целевой ВМ на момент запуска должно быть не активным (sleeping). В противном случае гипервизор ВМ не получит доступа к множеству F ее страниц памяти. В свою очередь состояние мигрируемой ВМ должно быть активным (running), поскольку в процессе ЖМ должен быть передан актуальный контекст $C_{ПО}$, выполняющегося на ней, как в момент начала процесса ЖМ ВМ, так и в ходе его развития.

Базовой операцией ЖМ ВМ является копирование (Copy) множества страниц D.

При этом гипервизор ВМ, определив целевую ВМ, итерационно в каждый момент времени процесса ЖМ выбирает подмножество страниц $D' \subset D$ ресурса памяти мигрируемой ВМ, доступных для выполнения операции Copy. К элементам подмножества D' относятся страницы, к которым в момент их выбора не происходит обращения на выполнение операции записи (Write) со стороны ПО (модулей, библиотек), функционирующих на мигрируемой ВМ.

Очевидно, что длительность процесса ЖМ ВМ существенно зависит от указанной выше активности функционирования ПО (модулей, библиотек) и может занимать существенное время и может приводить к увеличению целевого показателя качества предоставляемой пользователю услуги (времени отклика ПО). Например, исследования ЖМ ВМ конкретных прикладных сервисов [68] показали, что время простоя ВМ варьируется в зависимости от реализуемого ПО шаблона рабочей нагрузки (software workload) от 60 миллисекунд (игровое ПО) для до 3 секунд в случае функционирования ПО высокопроизводительных вычислений. Исследование, представленное в [35] показало, что время отклика ПО, выполняемого на мигрируемых ВМ, существенно увеличивается в течение всего периода ЖМ.

В рамках оптимизации процесса ЖМ ВМ по показателю времени отклика ПО были разработаны алгоритмы его реализации, использующие различные подходы к реализации операции Copy.

В настоящее время существуют реализации ЖМ ВМ на основе двух подходов:

1. Предварительное копирование (pre-copy) – основанном на двухэтапной схеме, реализующей предварительную итеративную отправку содержимого страниц оперативной памяти ВМ на ФМ-источнике (этап 1), и его остановку, и копирование состояния других его ресурсов (процессор и дисковое хранилище) на целевую ФМ (этап 2);

2. Последующее копирование (post-copy) – основан на предварительной передаче минимального состояния ВМ (CPU state, состояния регистров и подсистемы ввода/вывода) и последующей итеративной передаче страниц памяти по требованию (paging on demand).

В зависимости от реализации современные гипервизоры первого типа поддерживают различные из рассмотренных выше технологий ЖМ ВМ или их гибридные решения. Обобщение особенностей функционирования различных реализаций pre-copy и post-copy алгоритмов ЖМ представлено на рисунках 1.15 и 1.16.

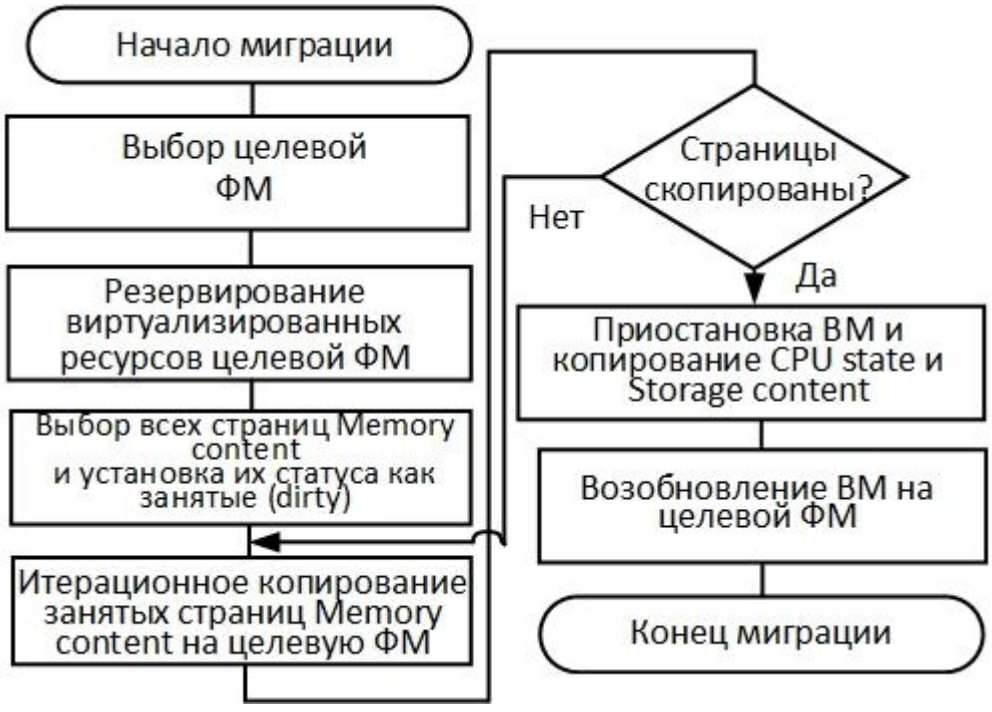


Рисунок 1.15 – Схема pre-copy алгоритма живой миграции ВМ

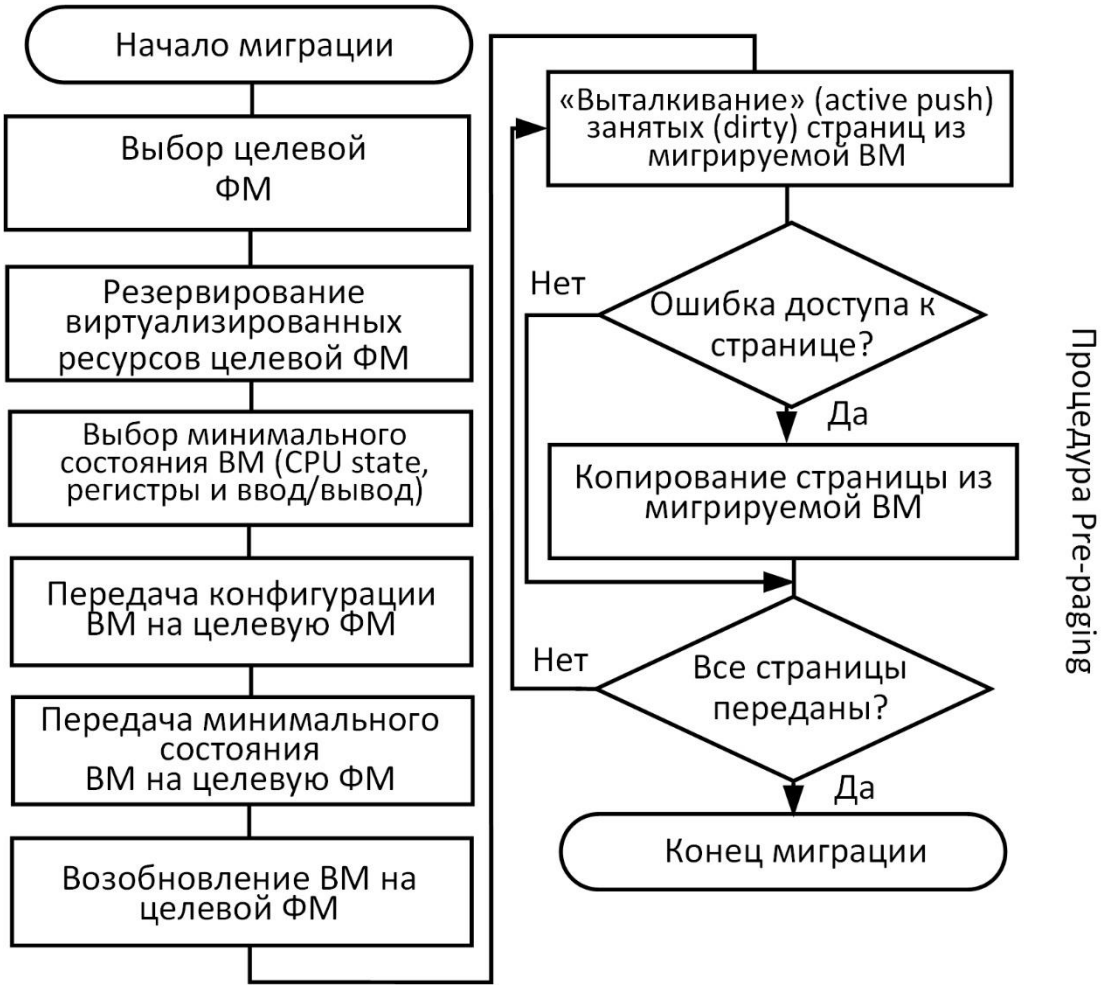


Рисунок 1.16 – Схема post-copy алгоритма живой миграции ВМ

В [65] выполнено расширенное исследование указанных алгоритмов ЖМ, включая рассмотрение алгоритма гибридной ЖМ, а также поведены экспериментальные исследования реализаций этих алгоритмов в гипервизорах Xen и KVM. На рисунке 1.17 обобщенно представлены временные диаграммы указанных алгоритмов миграции, отражающие соотношение времени простоя к общему времени миграции.

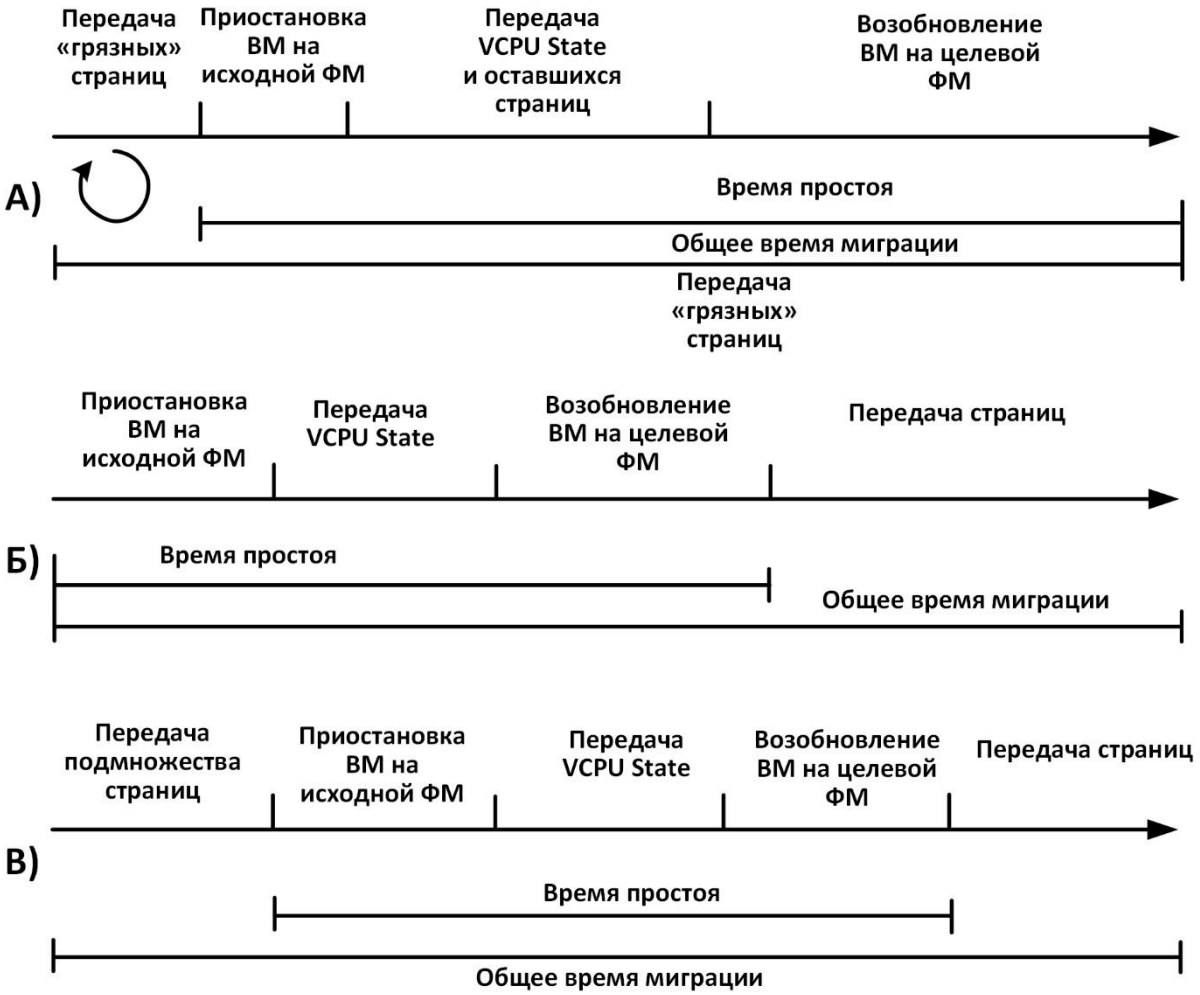


Рисунок 1.17 – Обобщенные временные диаграммы алгоритмов живой миграции: а) pre-copy, б) post-copy, в) гибридной живой миграции

В [69] также выполнено развернутое исследование особенностей реализации указанных алгоритмов с точки зрения времени простоя, приводящей к увеличению времени отклика, выполняемого на ВМ ПО.

В частности, определены базовые метрики, отражающие эффективности процесса ЖМ, реализуемого алгоритмами pre- и post-copy (таблица 1.1).

Таблица 1.1 Метрики для оценивания эффективности реализации процесса живой миграции виртуальных машин

Наименование метрики	Описание метрики
$t_{под}$ - время подготовки	Время между началом миграции и переносом контекста CPU (процессора) ВМ на целевую ВМ, во время которого мигрируемая ВМ продолжает выполняться и формировать новы страницы множества D. Для алгоритма pre-copy включает итеративную процедуру копирования всего множества страниц D. Для алгоритма post-copy это время является незначительным.
$t_{прост}$ - время простоя	Время, в течение которого выполнение мигрируемой ВМ приостановлено. В первую очередь включает в себя время передачи контекста CPU. Для алгоритма pre-copy это время передачи включает передачу последнего подмножества страниц $D' \subset D$. Для алгоритма post-copy время, необходимое ВМ для запуска на целевой ФМ.
$t_{воз}$ - время возобновления	Время между возобновлением выполнения ВМ на целевой ФМ и окончание миграции в целом. Для алгоритма pre-copy состоит из времен перепланировать целевой ВМ и удаления мигрируемой ВМ. Для алгоритма post-copy реализует основные этапы миграции (рисунок 1.18)
N_{page} - количество скопированных страниц	Общее количество скопированных страниц множества D, включая их дубликаты, по метрикам $t_{под}$, $t_{прост}$, $t_{воз}$. Алгоритм pre-copy копирует большую часть страниц за $t_{под}$. Алгоритм post-copy копирует большую часть страниц за $t_{воз}$.

На рисунке 1.18 представлены временные диаграммы реализации алгоритмов pre- (а) и post-copy (б), отображающие взаимосвязь представленных в Таблице 1.1 метрик.

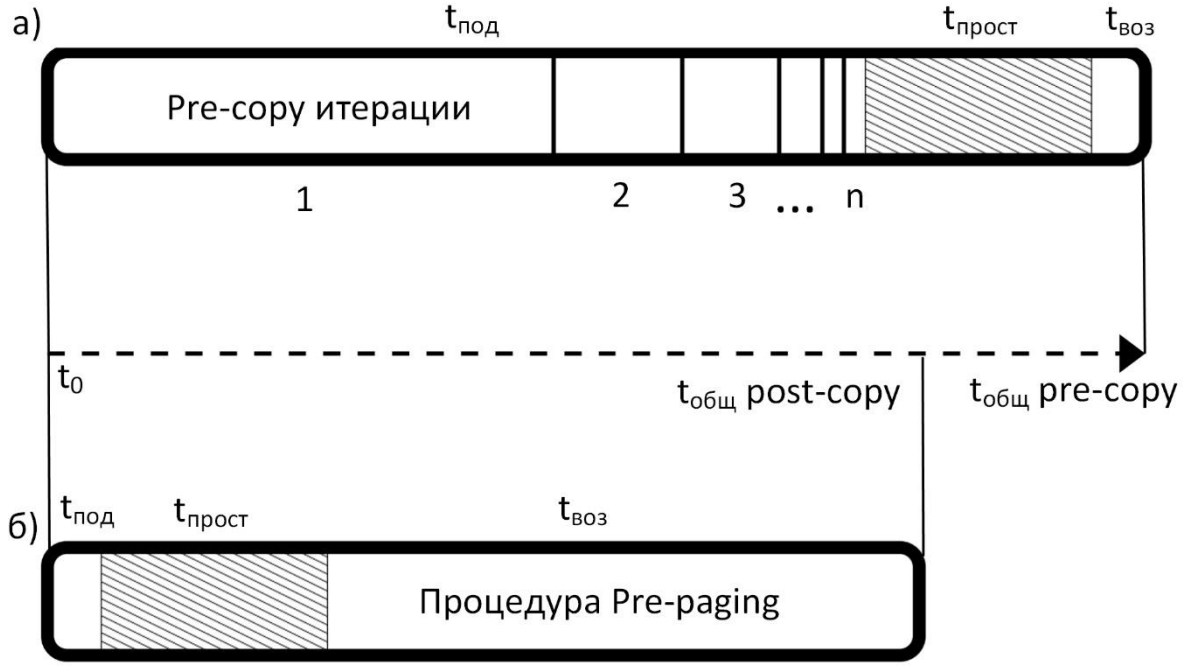


Рисунок 1.18 – Временные диаграммы реализации алгоритмов живой миграции: а) pre-copy, б) post-copy

В Приложении 1 приведены особенности алгоритмов ЖМ гипервизоров Xen VMM, KVM Hypervisor, Microsoft Hyper-V, ESXi.

В рамках процесса функционирования ЦОД реализация процесса ЖМ ВМ (вне зависимости от алгоритма ее реализации) выполняется для некоторого подмножества ВМ на основе предварительно формируемого плана переназначения (VM-PM Remapping) (этап 6, рисунок 1.11). Обычно он представляется в виде матрицы миграции (ММ) (в [69, 70] – матрицы расположения, Placement Matrix) – битовой карты, определяющей сопоставление n-й ВМ k-й ФМ.

Из рисунков 1.14-1.18 видно, что, в зависимости от используемого алгоритма ЖМ, временные характеристики этого процесса, а также характеристики ВКР, задействованных в ходе его реализации, существенно отличаются. Они определяют накладные расходы, составляющие часть процесса администрирования ЦОД и могут оказывать влияние на общую эффективность функционирования ЦОД.

Очевидно, что, динамика функционирования ВМ в составе крупных виртуализированных ЦОД, носящая в общем случае случайный характер, в

совокупности с автоматическим решением гипервизора ВМ задачи балансировки нагрузки на множество ФМ с использованием рассмотренных выше алгоритмов ЖМ ВМ, с течением времени могут приводить к фрагментации множества ВМ по множеству распределенных ВКР ФМ (рисунок 1.7). Обычно такая проблема фрагментации возникает в течение времени эксплуатации сформированной структуры виртуализированных ВКР в зависимости от интенсивности и особенностей задач, решаемых пользователями этих ВМ.

Таким образом, важным является рассмотрение методов, поддерживающих процесс переразмещения ВМ с учетом влияния, оказываемого на этот процесс алгоритмов ЖМ.

1.4 Исследование методов переразмещения виртуальных машин

Процесс переразмещения (replacement) множества ВМ в виртуализированных ЦОД существенно зависит от цели этого переразмещения. В [71,72,73] представлена обширная исследовательская база, связанная с проблемами переразмещения ВМ.

В частности, в [74] обобщаются методы, обеспечивающие поддержку процесса переразмещения. В общем виде они представлены на рисунке 1.19.

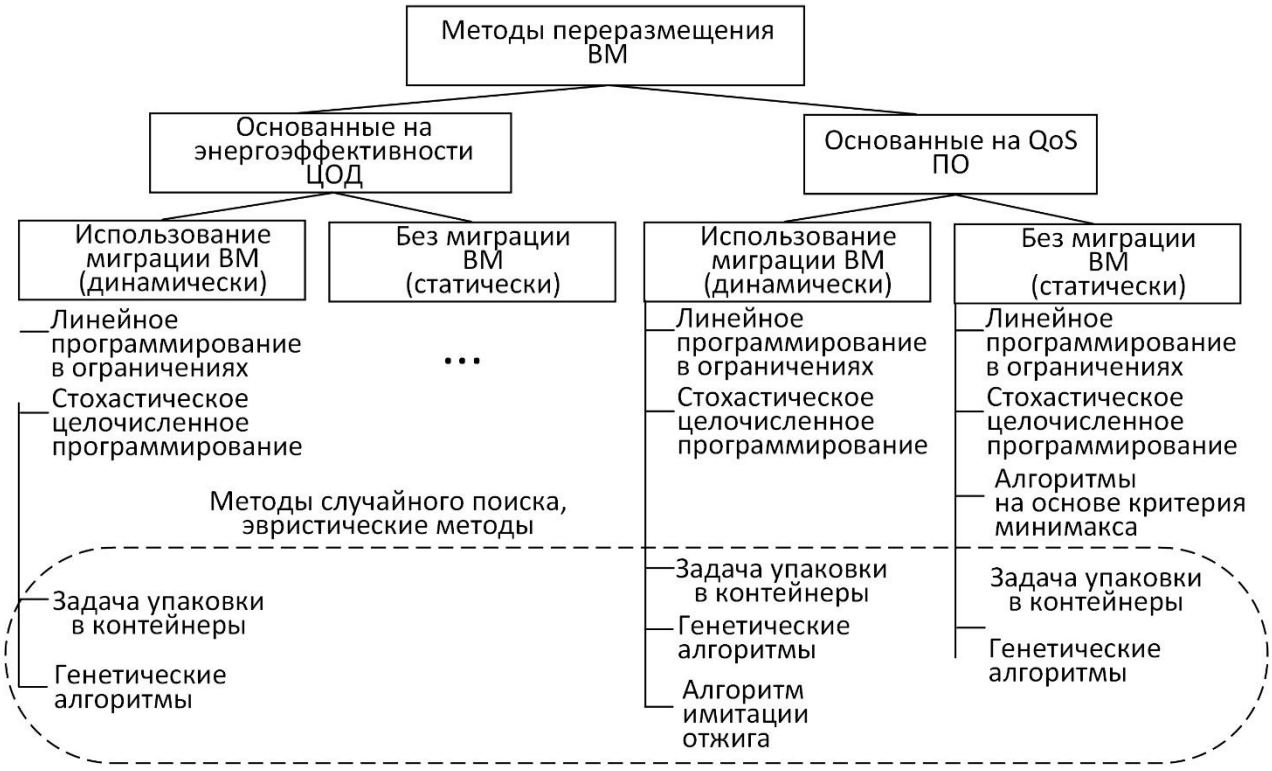


Рисунок 1.19 – Классификация методов поддержки процесса переразмещения ВМ

Из рисунка 1.19 видно, что в общем случае поддержку процесса переразмещения ВМ можно в целом разделить на две категории, отражающие цели переразмещения:

1. Методы, основанные на энергоэффективности ЦОД: основная цель этих методов – сопоставить множество ВМ множеству ФМ таким образом, чтобы некоторое подмножество ФМ (активные ФМ) использовалось с максимальной эффективностью, а оставшееся подмножество ФМ (не задействованные ФМ) могло быть, либо переведено в спящий режим, либо отключено (в зависимости от условий текущей нагрузки).

2. Методы, основанные на QoS ПО: основная цель этих методов – сопоставить множество ВМ множеству ФМ таким образом, чтобы максимизировать качество (QoS) обслуживания запросов пользователей ПО, функционирующего на множества ВМ.

Также из рисунка 1.19 следует, что реализовать процесс переразмещения в рамках указанных классов методов возможно, как на основе алгоритмов ЖМ ВМ

(п. 1.3), так и без их использования (только для класса методов, основанных на QoS ПО).

Наглядное представление варианта решения задачи динамического переразмещения ВМ с целью повышения энергоэффективности ВЦОД представлено на рисунке 1.20.

Как видно из рисунка 1.20 (а) мощность множества $|\text{ФМ}|=4$, при этом мощность подмножества $\text{ФМ}_\text{А}$ (активных ФМ) максимальна и равна $|\text{ФМ}_\text{А}|=4$, мощность подмножества $\text{ФМ}_\text{П}$ (не задействованных ФМ) равна $|\text{ФМ}_\text{П}|=0$. При этом значение показателя утилизации виртуализированных ВКР $\text{UR}_{\text{ФМ}_\text{А}} = \sum_{n=4} \text{ФМ}_n = 63{,}8$.

В результате переразмещения ВМ (рисунок 1.20 (б)) мощность подмножества $\text{ФМ}_\text{А}$ стала равна $|\text{ФМ}_\text{А}|=3$, значение показателя $\text{UR}_{\text{ФМ}_\text{А}} = \sum_{n=3} \text{ФМ}_n = 85$. Мощность подмножества $\text{ФМ}_\text{П}=1$.

Ниже обобщенно рассматриваются особенности конкретных методов переразмещения ВМ.

- методы, основанные на решении задачи линейного программирования в ограничениях (Constraint Linear Programming). Класс этих методов применяется в условиях детерминированности входных параметров (например, известных требований к ВКР со стороны множества ВМ). Подобные методы обеспечивают получение оптимального решения, однако, время, необходимое для его получения существенно возрастает по мере увеличения количества вводимых ограничений, что делает эти методы приемлемыми на этапах планирования первоначального размещения ВМ (рисунок 1.11);

- методы, основанные на решении задачи стохастического целочисленного программирования. Находят свое применение в условиях априорной неопределенности входных параметров, однако известных (или имеющих

возможность вычисления) их распределений вероятности. Обычно применяются метод в случае неопределенности двух и более входных параметров;

- методы основанные на решении задачи упаковки контейнеров (bin packing) [74,75]. Класс этих методов относится к эвристическому подходу решения оптимизационной задачи, что означает невозможность получения оптимального решения. Однако, это не является недостатком, поскольку в условиях решения задачи переразмещения ВМ с высокой вариативностью входных параметров оптимальное решение задастую не требуется. Вместо этого требуется квазиоптимальное решение для некоторого периода времени, связанного с вариативностью параметров. Очевидно, что в силу рассмотрения ВКР вектора ресурсов следует рассматривать класс методов многомерной (векторной) упаковки (Multi-dimensional Vector Packing Bin Problem).

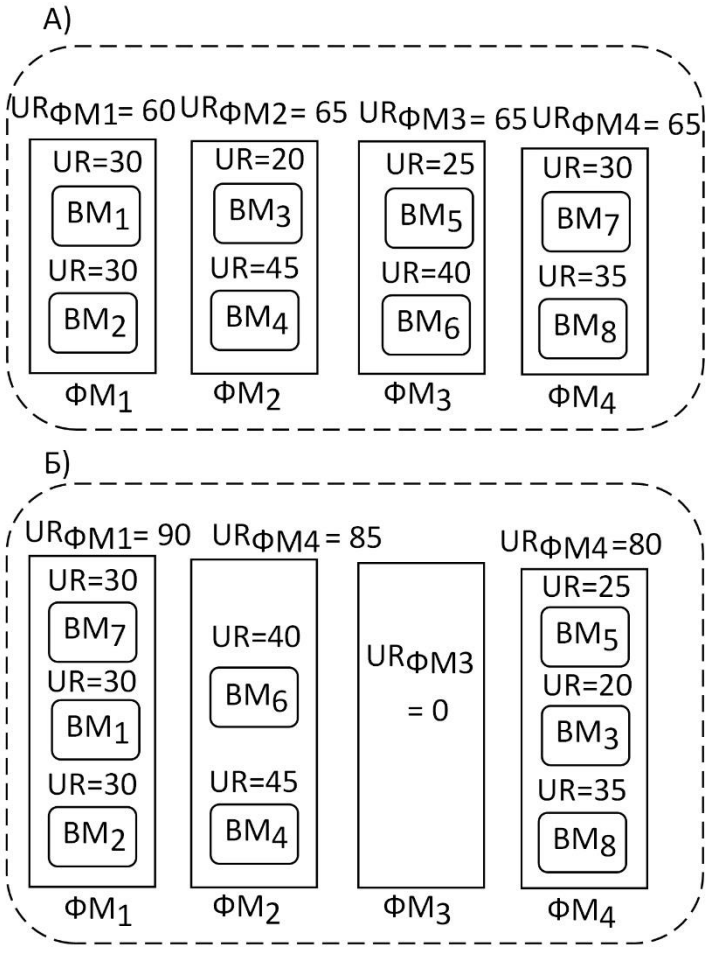


Рисунок 1.20 – Пример переразмещения ВМ с целью повышения энергоэффективности ЦОД

- генетические алгоритмы являются одним вариантов оптимизации комбинаторного решения задачи упаковки контейнеров в условиях ограничений.

Как и в случае минимизации затрат на производство и операции, задачи линейного программирования используется на этапе планирования размещения множества ВМ, потребности в ВКР которых не изменяются на достаточно длительных промежутках времени.

Очевидно, что каждый из представленных методов показывает свою эффективность при определенных условиях. Важной исследовательской задачей является выбор метода, который соответствует потребностям как потребителей, так и поставщиков услуг ВЦОД.

1.5 Обобщенное представление модели виртуализированного центра обработки данных

В общем случае, вне зависимости от рассмотренных методов, формально модель ВЦОД может быть представлена в рамках теоретико-множественного и теоретико-матричного подходов [76,77].

Определим следующие множества:

F – множество ФМ виртуализированного ЦОД;

V – множество ВМ, обеспечивающих обслуживание запросов пользователей виртуализированного ЦОД;

R – множество виртуализированных ВКР элементов множества F.

В [78,79] предлагается следующая редукция множества типов виртуализированных ВКР $R=\{R_1, R_2,...,R_d\}$ в соответствии с принципами их использования современными гипервизорами ВМ.

Элементы этого множества представлены следующими типами ВКР:

- состоянием процессора и/или процессорных ядер (CPU State);
- содержимым оперативной памяти (Memory content);
- содержимым дисковых хранилищ (Storage content).

При этом при рассмотрении алгоритмов ЖМ ВМ (pre- и post-copy) (п. 1.3) была отмечена важность виртуализированных коммуникационных ресурсов ЦОД, представленных сетевым вводом-выводом данных (Network I/O) в/из дисковых хранилищ, реализуемым обычно в виде SAN или NAS блочных устройств. Примером реализации такого типа ВКР является Amazon EBS (Elastic Block Storage) [80].

Таким образом, d-мерное представление виртуализированных ВКР в [81,82] предлагается свести к трехмерному представлению:

- CPU State – далее C, где C_i – элемент множества CPU State (например, ядро многоядерного процессора);

- Memory content – далее M, где M_j – элемент множества Memory content (например, страница оперативной памяти);

- Network I/O – далее N, где N_k – элемент множества Network I/O (например, пропускная способности SAN/NAS, выражаемая в блоках данных, передаваемых по сети в единицу времени).

Таким образом, множество виртуализированных ВКР R можно представить как вектор-функцию, аргументами которой являются векторные переменные:

- c_{fr} – объема указанных выше типов виртуализированных ресурсов $r=\{C,M,N\} \in R$ каждой ФМ – элементом множества $f \in F$;

- u_{vr} – использования (утилизации) указанных выше типов виртуализированных ресурсов $r=\{C,M,N\} \in R$ каждой ВМ – элементом множества $v \in V$. То есть, вектор-функцию R можно представить как трехмерное векторное поле $\mathbb{R}$ массивов векторных переменных c_{fr} и u_{vr} (выражение 1.1).

$$\begin{aligned} c_{fr} &\in \mathbb{R} \\ u_{vr} &\in \mathbb{R} \end{aligned} \tag{1.1}$$

Множества F, V и R, а также массивы векторных переменных c_{fr} и u_{vr} определяют структурно-параметрические характеристики ВЦОД с первоначальным распределением множества ВМ по множеству ФМ и,

следовательно, являются входными параметрами при моделировании конкретного варианта ВЦОД.

Результаты моделирования определяются следующими бинарными переменными:

- $y_f \in \{1,0\}$ – переменная задействования ФМ, принадлежащей множеству F. Принимает значение 1, если сопоставленная с ней ФМ активна - задействована для размещения хотя бы одной ВМ из множества V и принимает значение 0 в противном случае. Определяет массивы активных и свободных ФМ;

- $x_{fv} \in \{1,0\}$ – переменная сопоставления конкретной ВМ (элемента множества V) конкретной ФМ (элементу множества F).

Используя рассмотренное формальное представление виртуализированного ЦОД сформулируем задачу переразмещения ВМ используя два класса методов: линейного программирования в ограничениях и методов случайного поиска в постановке задачи упаковки контейнеров.

1.5.1 Формальная постановка задачи переразмещения виртуальных машин с использованием методов линейного программирования в ограничениях

Определим целевую функцию как сумму всех значений переменной $y_f = 1$ для всех элементов множества F (всех ФМ, задействованных для размещения ВМ).

Тогда можно сформулировать следующую задачу линейного программирования:

$$\min \sum_{f \in F} y_f \tag{1.2}$$

Определим для выражения 1.2 следующие ограничения:

$$\sum_{f \in F} x_{fv} = 1 \tag{1.3}$$

для $\forall v \in V$, а также:

$$\sum_{v \in V} u_{vr} x_{fv} \le c_{fr} y_f \tag{1.4}$$

для $\forall f \in F, \forall r \in R$.

Таким образом, целевая функция (выражение 1.2) направлена на минимизацию количества задействованных ФМ, а ограничения определяют, что каждая ВМ сопоставлена с конкретной ФМ и при этом использование виртуализированных ВКР каждой ВМ на конкретной ФМ не превышают имеющийся на ней объем виртуализированных ВКР.

Очевидно, что в подобной постановке задача линейного программирования в ограничениях наиболее эффективна для статического (первоначального) распределения множества ВМ по множеству ФМ с целью минимизации подмножества активных ФМ. При этом массив u_{vr} определяет пиковое значение потребностей в виртуализированных ВКР каждой ВМ.

Однако в условия функционирования ВЦОД она также может быть использована за счет реализации итерационной процедуры (шага переразмещения). При этом массив u_{vr} будет определять текущее значение потребностей в виртуализированных ВКР каждой ВМ.

Также в условиях функционирования ВЦОД можно рассматривать ограничение, связанное с запретом переразмещения ВМ, потребности в виртуализированных ВКР которых не изменяются с течением времени, что позволяет использовать подмножество активных ФМ, на которых они размещены в качестве основы для решения задачи переразмещения.

Указанная особенность может быть смоделирована введением следующих бинарных параметров:

- $s_v \in \{0,1\}$ – фиксированного (неизменяемого) использования ВМ виртуализированных ВКР. Принимает значение 0, если потребности ВМ в ВКР не изменяются и 1 в противном случае;

- $x'_{fv} \in \{0,1\}$ – определяет предыдущее сопоставление ВМ и ФМ. Принимает значение 1, если на предыдущем шаге переразмещения ВМ $v \in V$ была распределена на ФМ $f \in F$, и 0 в противном случае.

Таким образом, можно сформулировать дополнительное ограничение задачи переразмещения (выражение 1.2):

$$-s_v \le (x_{fv} - x'_{fv}) \le s_v \qquad (1.5)$$

для $\forall f \in F, \forall v \in V$.

Выражение 1.5 определяет, что если ВМ $v \in V$ изменяет свои потребности в виртуализированных ВКР, то значение переменной $s_v = 1$. При этом разница между текущим и предыдущим сопоставлением этой переменной находится в интервале [−1, 1]. В противном случае ВМ $v \in V$ имеет фиксированную (неизменяемую) потребность в виртуализированных ВКР, а значит, что значение переменной $s_v = 0$. Из этого следует, что разница между текущим и предыдущим сопоставлениями этой переменой должна быть равна 0, что означает равенство массивов x_{fv} и x'_{fv} этой ВМ для всех ФМ $f \in F$.

Очевидно, что для использования параметра x'_{fv} для следующего шага переразмещения, он сохраняется на предыдущем шаге.

Фактически, включение в модель дополнительного ограничения (выражение 1.5) гарантирует, что ВМ с фиксированной потребностью в виртуализированных ВКР не подлежат переразмещению и остаются на ВКР, которые им были выделены на этапе первоначального размещения.

Очевидно, что с ростом масштаба ВЦОД, повышением динамики изменения потребностей множества ВМ в виртуализированных ВКР, обусловленной особенностями запросов к ним пользователей, а также возможной динамикой изменения мощностей множеств F, V и R получение результатов решения задачи переразмещения в постановке задачи линейного программирования в ограничениях становится сложно выполнимой, как с точки зрения времени решения, так и потребных для решения вычислительных ресурсов (особенно в условиях многошагового процесса переразмещения). В связи с этим наряду с постановкой задачи методами математического моделирования рассматривается решение этой задачи методами случайного поиска, дающими приближенный (квазиоптимальный) результат за приемлемое время.

1.5.2 Формальная постановка задачи переразмещения виртуальных машин с использованием методов случайного поиска

Как и в п. 1.5.1, определим множества F и V. Определим множество виртуализированных ВКР всего множества F, как R, а количество типов этих ВКР, как d.

Тогда виртуализированные ВКР элемента $F_i \in F$ представим d-мерным вектором полноты ВКР:

$$P_i = \langle P_i^1, ... P_i^m, ..., P_i^d \rangle, \quad (1.6)$$

где P_i^m – полнота m-го ВКР $R_m \in R$, которым располагает элемент $F_i \in F$.

Соответственно, потребность в m-м ВКР со стороны элемента $V_j \in V$ представим d-мерным вектором потребности (demand) в ВКР:

$$D_j = \left\langle D_j^1, ... D_j^m, ..., D_j^d \right\rangle, \quad (1.7)$$

где D_j^m – потребность в m-м ВКР $R_m \in R$, которым располагает элемент $F_i \in F$.

Определим d-мерный вектор утилизации ВКР и матрицу миграции элементов множества V по элементам множества F. Вектор утилизации ВКР элемента $F_i \in F$ представим как:

$$U_i = \left\langle U_i^1, ... U_i^m, ..., U_i^d \right\rangle, \quad (1.8)$$

где U_i^k вычисляется с использованием выражения 1.9:

$$U_i^m = \sum D_j^m \text{ для } \forall MM_{i,j}=1, \quad (1.9)$$

где $MM_{i,j}$ – матрица миграции элемента $V_j \in V$ на элемент $F_i \in F$, которую в операторном виде представим выражением 1.10:

$$MM_{i,j}=\begin{cases}1, \text{ если } V_j \to F_i \\ 0, \text{ в противном случае}\end{cases} \quad (1.10)$$

Определим булевый вектор A размещения (allocation) элементов $V_j \in V$ на элементах $F_i \in F$, где каждый элемент вектора a_i принимает значение 1 в случае, если хотя бы один элемент множества V размещен на элементе $F_i \in F$. Представим его в операторном виде:

$$a_i=\begin{cases}1, \text{ если } \sum_{j=1}^{n} MM_{i,j} \geq 1 \\ 0, \text{ в противном случае}\end{cases} \quad (1.11)$$

Определим целевую функцию:

$$\min f(A)=\sum_{i=0}^{k} a_i, \quad (1.12)$$

со следующими ограничениями:

$$\sum_{j=1}^{n} D_j^m MM_{i,j} \leq P_i^m, \forall i \in \{1,...,n\}, \forall m \in \{1,...,d\} \quad (1.13)$$

$$\sum_{i=1}^{n} a_{i,j} \leq 1, \forall j \in \{1,...,k\}. \quad (1.14)$$

В рамках исследования выдвигается гипотеза, что использование методов детерминированного поиска, являющихся основой существующих алгоритмов и административных методик решения задачи переразмещения ВМ, не является целесообразным в силу того, что вид целевой функции (выражение 1.12) в общем случае неизвестен, а в частных случаях может иметь сложную структуру.

Это определяет необходимость выбора метода решения задачи из класса методов случайного поиска, к которым относятся методы:

- стохастического программирования;
- эвристической и квазиэвристической оптимизации;
- методы на основе использования искусственных нейронных сетей.

Выбор и обоснование варианта используемого метода случайного поиска и конкретных алгоритмов его реализации, а также проведение соответствующих численных и имитационных экспериментов, подтверждающих эффективность его использования является важной исследовательской задачей.

1.5.3 Особенности задачи переразмещения виртуальных машин в гетерогенных ЦОД

Представленные в п.п. 1.5.1 и 1.5.2 постановки задачи переразмещения ВМ с целью повышения энергоэффективности виртуализированного ЦОД имеет существенное структурное ограничение, связанное с гомогенностью элементов множества (множества ФМ). Под гомогенностью ФМ (см. п. 1.1) будем рассматривать наличие у всех элементов множества F гипервизоров (ГВ) одного типа. Например, виртуализированный ЦОД может быть спроектирован на основе гипервизора Xen или использовать гипервизор Hyper-V, входящий в состав серверной ОС Windows Server.

При этом, существуют варианты гетерогенной структуры виртуализированных ЦОД, в которых разные подмножества элементов множества F поддерживают ГВ различных типов. Такая структура накладывает дополнительные ограничения на решение задачи переразмещения ВМ. К ним можно отнести:

1. Полную или частичную несовместимость алгоритмов ЖМ, реализованных в различных ГВ.

2. Разницу в значениях параметров процесса ЖМ (Таблица 1.1) одинаковых типов алгоритмов (pre-copy и post-copy), реализованных в различных ГВ.

Первое ограничение накладывает особенности формирования паросочетаний ВМ-ФМ в матрице миграции (выражение 11), ограничивая выбор ФМ, поддерживающих одинаковый тип ГВ. Второе ограничение оказывает влияние на уровень накладных расходов процесса ЖМ (использования виртуализированных ВКР ФМ для организации процесса ЖМ).

Так в [81] исследуется сравнительная оценка таких накладных расходов для двух типов ГВ – Xen и KVM. На рисунке 1.21 представлены гистограммы, отображающие сравнительную оценку: реализации процесса ЖМ (показатель TMT – Total Migration Time) (рисунок 1.21 а)) и времени простоя ВМ (показатель DT – Down Time) (рисунок 1.21 б)).

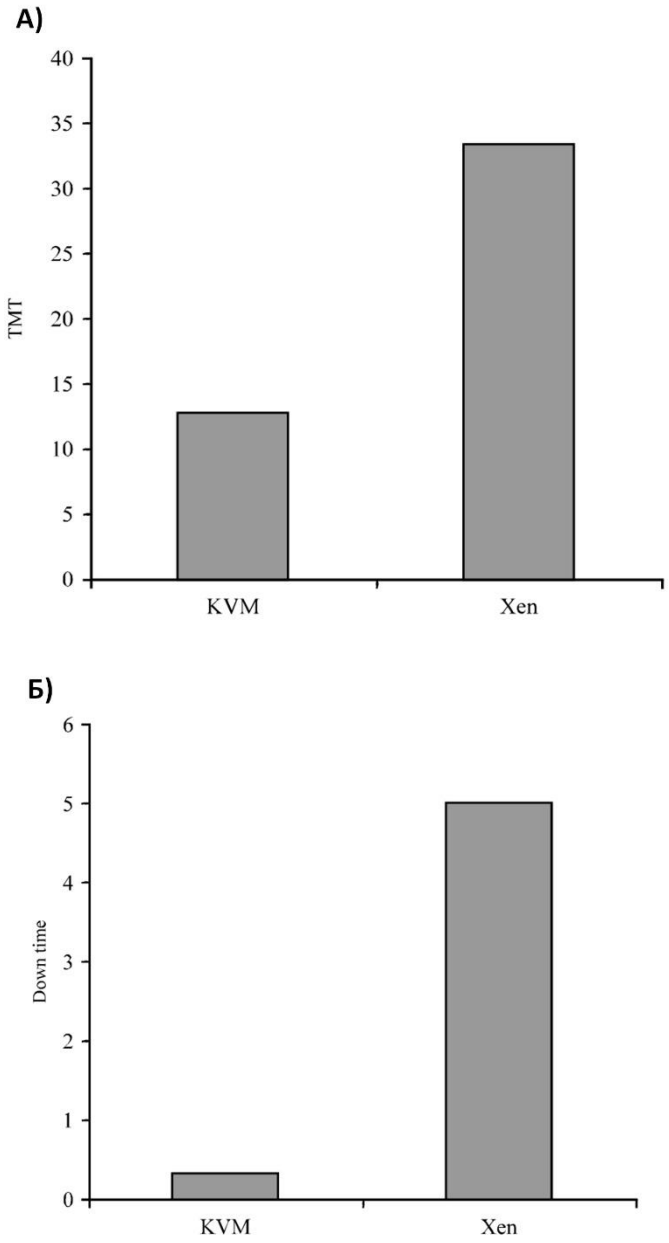


Рисунок 1.21 – Сравнительная оценка временных параметров процесса живой миграции гипервизоров Xen и KVM: а) общего времени миграции, б) времени простоя ВМ

На рисунке 1.22 представлена зависимость объема «грязных» страниц памяти (показатель Dirty Memory) от общего объема мигрируемых данных (показатель Migration data) для рассматриваемых типов ГВ. Наблюдаемая разница связана с особенностями реализации алгоритмов ЖМ: для Xen – pre-copy и для KVM – post-copy.

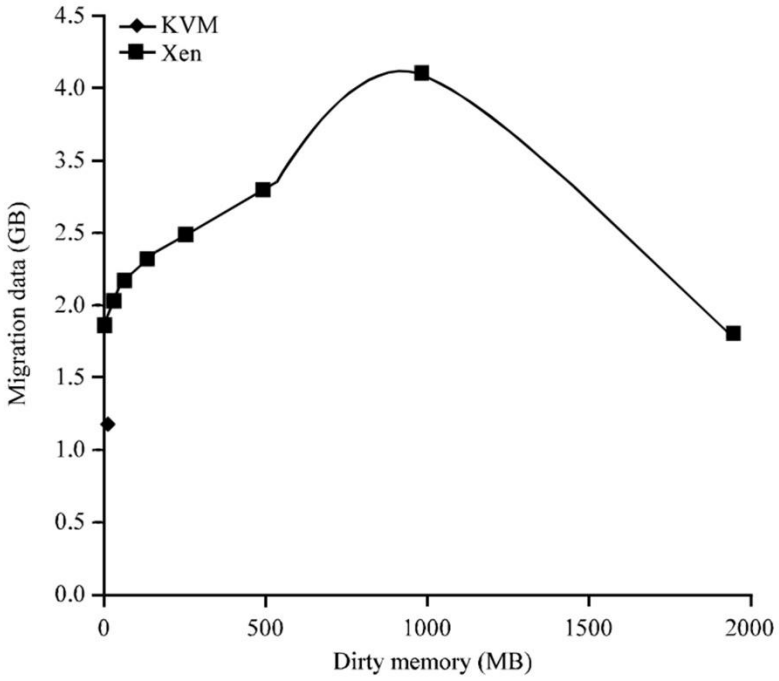


Рисунок 1.22 – Сравнительная оценка гипервизоров Xen и KVM по соотношению в передаваемом в процессе живой миграции объеме данных ВМ содержимого «грязных» страниц памяти

Указанные особенности реализации алгоритмов ЖМ различными типами ГВ требуют учета, как в процессе моделирования виртуализированных ресурсов ЦОД, так и в процессе разработки алгоритма переразмещения ВМ.

1.6 Постановка задачи исследования

Исходя из рассмотренных в п.п. 1.1-1.5 особенностей организации ВЦОД, в частности, ЦОД с гетерогенной структурой, особенностей реализации алгоритмов живой миграции ВМ, а также особенностей методов переразмещения ВМ в

процессе эксплуатации ВЦОД предлагается следующая постановка задачи исследования:

Объект исследования – виртуализированный ЦОД гетерогенного типа.

Предмет исследования – процесс переразмещения виртуальных машин в виртуализированном ЦОД гетерогенного типа на основе их живой миграции.

Цель исследования – разработка математического и программного обеспечения для повышения эффективности использования виртуализированных вычислительных и коммуникационных ресурсов в гетерогенном виртуализированном ЦОД (два или более типов гипервизоров) за счет реализации процесса переразмещения ВМ.

Исходя из цели исследования, научная задача исследования представляется совокупностью следующих задач.

1. Провести анализ состояния проблемы динамического переразмещения ВМ на основе их живой миграции между физическими машинами ЦОД.

2. Разработать модель виртуализированных вычислительных и коммуникационных ресурсов виртуализированного ЦОД, учитывающую их накладные расходы на процесс живой миграции ВМ.

3. На основе обобщения существующих эвристических и метаэвристических методов решения задачи многомерной векторной упаковки контейнеров обоснованно выбрать и модифицировать метод, применимый к задаче переразмещения ВМ и на его основе разработать алгоритм поддержки процесса переразмещения ВМ, обеспечивающий достижение цели исследования по критерию пригодности.

4. Разработать архитектуру программной реализации системы поддержки процесса переразмещения ВМ в гетерогенных виртуализированных ЦОД.

5. Провести численные эксперименты по оцениванию качества функционирования разработанного алгоритма поддержки процесса переразмещения ВМ на примере мониторинга узлов фрагмента транспортной сети с коммутацией пакетов для различной структуры виртуализированных ЦОД гетерогенного типа.

1.7 Выводы по главе

Таким образом, в данной главе рассмотрены особенности решения задач балансировки загрузки ФМ ВЦОД, а также консолидации ВМ, выявлены их отличия с точки зрения решения задач оптимизации структурно-параматрических характеристик ВЦОД.

Проведен анализ видов миграции ВМ, определены особенности алгоритмов живой миграции ВМ с точки зрения потребления ими ресурсов физической машины. Рассмотрены существующие подходы к оцениванию эффективности алгоритмов живой миграции, обобщены метрики, используемые для оценивания эффективности этих алгоритмов.

Выполнено обобщение методов, применяемых для решения различного типа задач переразмещения ВМ, рассмотрены особенности, достоинства и недостатки рассмотренных методов. Выполнена формальная постановка задачи переразмещения ВМ как задачи линейного программирования в ограничениях и как задачи случайного поиска решения.

На основе рассмотренного анализа предметной области задачи переразмещения ВМ в ВЦОД сформулирована необходимость совершенствования их математического и программного обеспечения, связанного с решением задачи переразмещения ВМ с целью обеспечения энерго- и/или ресурсоэффективности ВЦОД, в частности гетерогенных ВЦОД.

В этой связи, было сформулировано противоречие между необходимостью повышения коэффициента использования ФМ, комбинаторной сложностью задачи переразмещения ВМ, использованием в существующих подходах переразмещения ВМ преимущественно жадных алгоритмов оптимизации, а также отсутствием в них учета гетерогенной структуры ВЦОД. На основании этого противоречия сделана постановка научной задачи по разработке математического и программного обеспечения переразмещения ВМ в гетерогенных ЦОД.

ГЛАВА 2. РАЗРАБОТКА МАТЕМАТИЧЕСКОГО ОБЕСПЕЧЕНИЯ ПРОЦЕССА ПЕРЕРАЗМЕЩЕНИЯ ВИРТУАЛЬНЫХ МАШИН В ВИРТУАЛИЗИРОВАННЫХ ЦЕНТРАХ ОБРАБОТКИ ДАННЫХ

2.1 Исследование подходов к моделированию виртуализированных ресурсов центров обработки данных для решения задач их выделения и перераспределения между множеством виртуальных машин

2.1.1. Классы задач выделения и перераспределения виртуализированных ресурсов центров обработки данных для множества виртуальных машин

Как было рассмотрено в п. 1.2, важным классом задач, обеспечивающих эффективное функционирование ЦОД, которые предоставляют множество виртуализированных ВКР для запуска и эксплуатации множества ВМ, являются задачи, связанные с выделением и перераспределением ВКР ЦОД между множеством ВМ для достижения определенных целей. Технологической основой подобных задач является механизм ЖМ ВМ, особенности и актуальные алгоритмы которого рассмотрены в п. 1.3.

Исследование источников, посвященных повышению эффективности использования виртуализированных ВКР [34,57,60,70,84], позволили выделить классы задач, связанных с решением проблемы перераспределения множества ВКР между множеством ВМ:

1. Задача статического размещения ВМ. Является базовой задачей, связанной с этапом проектирования и/или модернизации ЦОД, например, с целью масштабирования его вычислительных и коммуникационных возможностей. В подобном классе задач определено множество ВМ с сопоставленных каждому элементу множества значением требуемых ВКР. Также определено множество серверов ЦОД (далее ФМ) с сопоставленным каждому элементу множества значением имеющихся ВКР. Целью решения задачи статического размещения ВМ является получение подмножества ФМ, значения имеющихся ВКР которого, с

заданной степенью полноты покрывают значения требуемых ВКР всего множества ВМ. При этом указанное подмножество ФМ должно быть минимизировано.

2. Задача динамического размещения ВМ. Является базовой задачей, связанной с этапом эксплуатации ЦОД. В общей постановке эта задача соответствует задаче статического размещения ВМ. Отличительной особенностью является представление подмножества ФМ ЦОД, на которых в некоторый момент времени t эксплуатируется подмножество ВМ, именуемых «активные (резидентные) ВМ». В силу того, что часть ВКР указанного подмножества ФМ выделены подмножеству активных ВМ (в [85,86,87] именуются утилизированные ВКР), то в постановке задачи динамического размещения ВМ для каждого элемента подмножества ФМ указывается не общее, а текущее значение ВКР (именуются «свободные» или «не выделенные» ВКР). Дополнительным условием этой задачи является появление в момент времени t+1 подмножества ВМ (именуются «новые ВМ») с сопоставленных каждому элементу множества значением требуемых ВКР. В общем случае целью решаемой задачи является сопоставление элементов подмножества ФМ, на котором эксплуатируются активные ВМ элементам подмножества новых ВМ. При этом, как и в случае задачи статического размещения ВМ, указанное подмножество ФМ должно быть минимизировано. Существуют модификации задачи динамического размещения ВМ, связанные с динамическим переводом в моменты времени t+n некоторых элементов подмножества активных ВМ в подмножество «не активных ВМ» (ВМ, не предъявляющих требования к ВКР), а также динамическим переводом в моменты времени t+m некоторых элементов подмножества не активных ВМ в подмножество активных ВМ.

3. Задача балансировки нагрузки множества ФМ. Является расширенным в критериальной части вариантом задачи динамического размещения ВМ. Как и в случае общей постановки задачи балансировки нагрузки в распределенных системах [88], задача балансировки нагрузки множества ФМ включает этапы:

- определения в момент времени t подмножества «перегруженных ФМ» (ФМ, на которых эксплуатируется подмножество активных ВМ сопоставленных

каждому элементу множества значением требуемых ВКР, такими что их общее значение равно или превышает значения сопоставленных каждому элементу множества значением текущих ВКР);

- решения в момент времени t+1 задачи динамического размещения подмножества активных ВМ на подмножестве перегруженных ФМ и/или дополнительно используемого подмножества «резервных ФМ» с целью минимизации подмножества перегруженных ФМ.

4. Задача консолидации множества ФМ. В общей постановке является обратным вариантом задачи балансировки нагрузки ФМ. Включает этапы:

- определения в момент времени t подмножества «недогруженных ФМ» (ФМ, на которых эксплуатируется подмножество активных ВМ сопоставленных каждому элементу множества значений требуемых ВКР, такими что их общее значение значительно меньше значения сопоставленных каждому элементу множества значений текущих ВКР);

- решения в момент времени t+1 задачи динамического размещения подмножества активных ВМ на подмножестве недогруженных ФМ с целью минимизации этого подмножества ФМ. В [89,90] задача консолидации ФМ обычно рассматривается с целью снижения таких показателей эффективности функционирования ЦОД, как PUE (Power Usage Effectiveness – отношение суммарной мощности, потребляемой ЦОД к суммарной мощности, потребляемой множеством ФМ, а также DCiE (Data Center Infrastructure Efficiency – отношение суммарной мощности, потребляемой элементами множеств всех компонентов ЦОД, предоставляющих ВКР (СХД, сетевая инфраструктура, множество ФМ) к общей мощности, потребляемой ЦОД.

Очевидно, что для решения указанных выше задач требуется разработка и исследование эффективности использования, соответствующего особенностям каждой задачи математического и программного обеспечения. Однако, вне зависимости от класса задач, можно рассматривать единый методологический подход к моделированию виртуализированных ресурсов ЦОД. Это связано с

единым для каждой задачи представлением виртуализированных ВКР в качестве наличного, выделенного или запрашиваемого ВМ ресурса.

2.1.2 Подходы к представлению виртуализированных ВКР в векторной форме

Обобщенный подход к формальному представлению виртуализированных ВКР в ЦОД достаточно широко представлен в [81,82,91]. Он основан на многомерном векторном представлении наличия, выделения (утилизации) и запросов на использование ВКР.

При этом определяются:

- множество ФМ ЦОД – $F=\{F_1,F_2,...,F_k\}$;

- множество ВМ, эксплуатируемых пользователями ЦОД – $V=\{v_1,v_2,...,v_n\}$;

- на множестве F определяется множество ВКР $R=\{R_1, R_2,...,R_d\}$, где d – количество типов ВКР.

Тогда ВКР элемента $F_i \in F$ можно представить d-мерным вектором полноты ВКР:

$$P_i = \langle P_i^1,...,P_i^m,...,P_i^d \rangle \quad (2.1)$$

где P_i^m – полнота m-го типа ВКР $R_m \in R$, которым располагает элемент $F_i \in F$.

Соответственно, потребность в ВКР m-го типа со стороны элемента $v_j \in V$ представим d-мерным вектором запроса (demand) в ВКР:

$$D_i = \langle D_i^1,...,D_i^m,...,D_i^d \rangle \quad (2.2)$$

где D_j^m - запрос элементом $v_j \in V$ m-го типа ВКР.

- также на множестве F определяется d-мерный вектор выделения (utilization) ВКР для элемента $F_i \in F$ он определяется как:

$$U_i = \langle U_i^1, ..., U_i^k, ..., U_i^d \rangle \quad (2.3)$$

где U_i^k вычисляется с использованием выражения:

$$U_i^k = \sum D_j^m \quad (2.4)$$

Таким образом, с использованием представления виртуализированных ВКР в векторной форме можно определить принцип размещения требуемых ВМ на конкретные ФМ в зависимости от:

- потребностей ВМ в виртуализированных ВКР (demand, выражение 2.2);
- выделение на ФМ виртуализированных ВКР (utilization, выражение 2.3).

2.1.2.1 Формальное представление дисбаланса виртуализированных ВКР

Очевидно, что рассмотрение задачи выделения виртуализированных ВКР следует рассматривать в совокупности, в едином ресурсном пространстве их состояний. Обобщенный подход к такому представлению дается в [91,92].

Рассмотрим указанные выше типы ВКР в качестве трех измерений абстрактного объекта. Нормализуя типы ВКР по каждой оси, общий имеющийся в наличии ВКР множества F может быть представлен в виде единичного куба [92] (рисунок 2.1). В [92] такой единичный куб именуется нормализованным ресурсным кубом (Normalized Resource Cube – NRC). В терминах векторной алгебры нормализованная мощность множества виртуализированных ВКР каждого типа (C, M, N) может быть представлена единичными векторами (ортами) куба NRC.

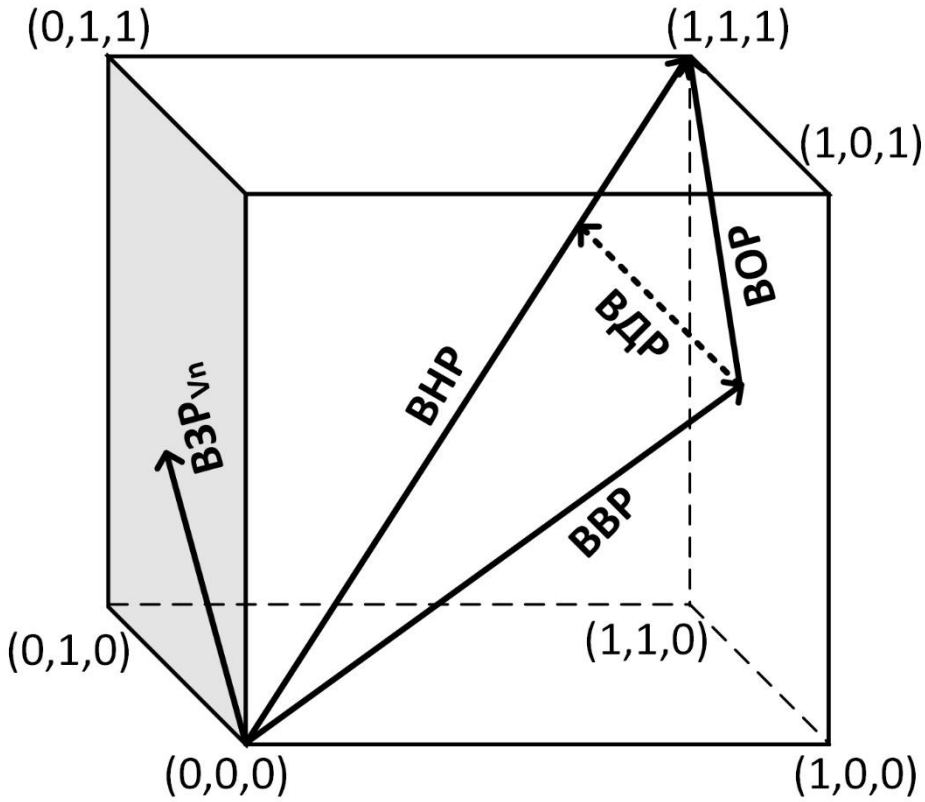


Рисунок 2.1 – Представление нормализованного ресурсного куба (NRC) и соответствующих векторов нормализованного представления трех тиров ВКР

Очевидно, что рассмотренные выше векторы наличия, выделения и запроса ВКР (выражения 2.1-2.3) могут быть представлены векторами внутри куба NRC.

Так общая полнота ВКР всего множества F рассматривается, как вектор из начала координат NRC (0, 0, 0) в точку с координатами (1, 1, 1). Определим его как вектор ВНР (вектор наличия ресурсов).

Тогда вектор выделения всех трех типов ВКР некоторому подмножеству множества V представим, как ВВР (вектор выделения ресурсов). При этом очевидно, что нормализация использования отдельных типов ВКР (C, M, N) выполняется относительно вектора ВНР.

Также, исходя из принципов векторной алгебры, остаточная полнота ВКР множества F после выделения части ВКР подмножеству множества V, представляется векторной разностью между векторами ВНР и ВВР. На рисунке 2.1 такой разносный вектор именуется ВОР (вектор оставшихся ресурсов).

Подобным образом можно определить запросы к ВКР всех трех типов конкретным элементом $v_n \in V$. Эти запросы представляют собой векторное сложение нормализованных векторов D_j (выражение 2.4) ВКР каждого типа. На рисунке 2.1 такой результирующий вектор именуется $ВЗР_{v_n}$ (вектор запроса ресурсов элементом $v_n \in V$).

Очевидно, что степень сбалансированного использования всех типов ВКР определяется близостью векторов ВНР и ВВР. При их полном совпадении предполагается что ВКР используются сбалансировано по каждой оси типа ВКР (C, M, N). Тогда степень несбалансированности использования ВКР всех типов в векторной форме можно представить вектором ВДР (вектор дисбаланса ресурсов), который представляет собой векторную разность между проекцией вектора ВВР на вектор ВНР и самим вектором ВВР (рисунок 2.1).

2.2 Моделирование накладных расходов, возникающих в процессе живой миграции виртуальных машин на основе векторной модели ресурсного куба NRC

Очевидным недостатком рассмотренных в п.п. 2.2.1-2.2.4 методов распределения элементов множества V по элементам множества F является отсутствие учета ресурсных затрат, связанных с базовым механизмом распределения – ЖМ ВМ. Исходя из рассмотренного в п. 2.1.2.1 векторного представления ресурсов множества F в трехмерном ресурсном пространстве куба NRC, вектор ВНР можно представить как сумму векторов типов ресурсов:

$$ВНР=C+M+N, \quad (2.5)$$

а вектор ВВР, как сумму векторов типов ресурсов, выделенных подмножеству активных ВМ – $F_A \subset F$:

$$ВВР=\hat{C}+\hat{M}+\hat{N} \quad (2.6)$$

Тогда вектор ВДР (дисбаланс ресурсов) в общем случае определяется как векторная разность проекций на ВНР и ВВР:

$$ВДР'=\sqrt{(C-\Delta)^2+(M-\Delta)^2+(N-\Delta)^2} \quad (2.7)$$

где $\Delta=\frac{ВНР}{З}$.

Как было отмечено в п. 2.1.2.1, подобное представление ВКР ЦОД позволяет использовать вектор ВДР в качестве параметра при реализации алгоритма перераспределения ВКР. При этом для выбора элемента множества V (конкретной ВМ), размещаемого на элементе множества F (конкретной ФМ) следует выбирать такой элемент $v_n \in V$, который уменьшает величину вектора ВДР (задает равновесное значение C, M и N измерений куба NRC). Указанная величина ВДР может быть вычислена согласно выражению:

$$ВДР'=\sqrt{(C-\Delta)^2+(M-\Delta)^2+(N-\Delta)^2} \quad (2.8)$$

Исследование алгоритмов ЖМ ВМ типов pre- и post-copy (п. 1.3) показало, что процедуры алгоритмов ЖМ также задействуют ВКР ФМ. Поскольку, вне зависимости от типа ЖМ, базовой процедурой является копирование и передача по сети страниц памяти мигрируемой ВМ между исходной и целевой ФМ (рисунок 2), основными типами ВКР, необходимыми для ее реализации являются тип M и N.

Таким образом, в рассмотренном выше векторном подходе к моделированию виртуализированных ВКР следует учитывать ВКР, используемые тем или иным алгоритмом ЖМ ВМ. Обозначим этот тип ВКР как $ВКР^{нр}_{жм}$ – накладные расходы (overhead cost) процесса ЖМ ВМ.

В общем виде ВКР ЦОД, доступные в процессе перераспределения ВМ можно представить как:

$$ВКР_{дост} = ВКР_{общ} - ВКР^{нр}_{жм} \quad (2.9)$$

То есть величина вектора ВНР должна быть уменьшена с учетом векторного значения $ВКР^{нр}_{жм}$.

Для более точного представления $ВКР^{нр}_{жм}$ необходимо рассмотреть параметры, используемые алгоритмами ЖМ ВМ, а также определить факторы, влияющие на значение $ВКР^{нр}_{жм}$ вне зависимости от используемого типа алгоритма.

Исследование алгоритмов ЖМ ВМ типа pre- и post-copy позволило определить следующие параметры, используемые ими в процессе миграции конкретной ВМ:

- ν_M – объем ВКР типа M (используется метрика N_{page} (таблица 1.1). Исходя из определения временных метрик ЖМ, длительность их первой итерации ($t_{под}$) пропорциональна значению ν_M и оказывает влияние на длительность последующих этапов реализации алгоритма, например, на $t_{воз}$ алгоритма post-copy;

- K_{dp} – коэффициент «загрязнения» страниц памяти (dirty page). Является отношением страниц памяти, отмеченных менеджером памяти ФМ, как «грязные» (dirty – содержащие конфигурацию и данные мигрируем ВМ), к общему числу страниц памяти, выделенной ВМ. То есть, $K_{dp} = \frac{N^{dirty}_{page}}{N_{page}}$.

Параметр K_{dp} является динамическим поскольку в течение времени $t_{под}$ (рисунок 1.18, таблица 1.1) мигрируемая ВМ продолжает выполняться, а, следовательно, формировать новые значения N^{dirty}_{page}. Очевидно, что увеличение коэффициента K_{dp} приводит к пропорциональному росту времени выполнения всех этапов алгоритмов ЖМ, включая время простоя ($t_{прост}$);

- C_N – пропускная способность ВКР типа N, выделенная для реализации алгоритма ЖМ. Имеет обратную пропорциональную зависимость со значениями времен $t_{под}$ и $t_{прост}$;

- D_N – длительность (distance, расстояние) маршрута ВКР типа N между исходной и целевой ФМ. Определяется количеством точек маршрутизации между

этими типами ФМ и формирует время задержки пакетов, содержащих конфигурацию и данные мигрируемой ВМ.

Очевидно, что параметры C_N и D_N взаимозависимы, поскольку общее значение C_N складывается из частных значений пропускной способности каналов между точками маршрутизации.

Рассмотрение реализации алгоритмов ЖМ конкретных $ГВ_{ЦОД}$ (KVM, VMware ESXi) показало, что в настройках их конфигурации определяются следующие пороговые значения, связанные с рассмотренными выше параметрами:

- порог значения N_{page}^{dirty};

- количество итераций (раундов) процедур pre-copy и pre-paging (рисунок 1.18).

Учет этих параметров требуется в случае реализации алгоритмов ЖМ в ЦОД с гетерогенными $ГВ_{ЦОД}$ (имеющими разное значение этих параметров). В случае реализации ЦОД с гомогенными $ГВ_{ЦОД}$ эти параметры можно исключить из рассмотрения.

С учетом рассмотренных выше параметров процесса ЖМ ВМ можно определить следующие факторы, оказывающие влияние на значение $ВКР_{жм}^{нр}$:

- $ПД_{ЖМ}$ – процесс передачи конфигурации и данных в ходе реализации алгоритмов ЖМ. Существенно связан с рассмотренными параметрами миграции и оказывает влияние на расход в $ВКР_{жм}^{нр}$ типов M и N;

- $T_{ЖМ}$ – общее время, затраченное на миграцию конкретной ВМ (с момента начала процедур алгоритмов ЖМ до запуска ВМ на целевой ФМ). Оказывает существенное влияние на расход в $ВКР_{жм}^{нр}$ типа C, поскольку в процессе миграции производительность ВМ существенно снижается;

- $T_{ВМ}^{прост}$ – общее время простоя ВМ в процессе ее миграции. Определяет период недоступности ВМ и, как следствие, поддерживаемых ею сервисов для потребителя;

- $N_{НР}^{ВМ}$ – накладные расходы в $ВКР_{жм}^{нр}$ типа N, определяются, как произведение параметров ν_M и D_N с учетом параметров K_{dp} и C_N.

Исходя из предложенных факторов, значение $ВКР_{жм}^{нр}$ при миграции n-й ВМ на k-ю целевую ФМ, можно представить в следующем виде:

$$НР_{V_n}^{F_k} = \sigma_1 \times ПД_{ЖМ} + \sigma_2 \times Т_{ЖМ} + \sigma_3 \times Т_{V_n}^{прост} + \sigma_4 \times N_{НР}^{V_n}, \quad (2.10)$$

где $\sigma_1, \sigma_2, \sigma_3, \sigma_4 \in [0,1]$ – весовые коэффициенты, определяющие в каждом конкретном случае важность связанных с ними факторов. При этом сами факторы нормализуются относительно их возможных максимальных значений.

Предложенный подход к определению накладных расходов, связанных с ЖМ ВМ, позволяет использовать указанные параметры и факторы при расчете накладных расходов при разработке алгоритмов динамического перераспределения и объединения ВМ.

Таким образом, решение класса задач, связанных с размещением ВКР и представленных в п. 2.1.1, может быть сведения к решению задачи минимизации величины вектора ВДР.

Далее проводится анализ ряда подходов, использующих подобную методологию, с целью определения их достоинств и недостатков относительно решения научной задачи исследования, формальная постановка которой дана в п. 1.6.

2.3 Исследование существующих подходов к решению задач выделения и перераспределения виртуализированных ресурсов центров обработки данных с использованием их векторного представления

2.3.1 Метод планарного шестиугольника ресурсов

Наиболее известным методом решения указанных в п. 2.1.1 задач выделения и перераспределения виртуализированных ВКР ЦОД, использующим подход векторного представления типов ВКР на основе куба NRC, является метод планарного шестиугольника ресурсов (planar resource hexagon), далее ПШР представленный в [93,94].

Указывается, что метод ПШР является наиболее эффективным при решении задачи балансировки нагрузки и обеспечивает эффективный поиск элементов множества V для их размещения на элементах множества F в случаях дисбаланса запросов по отдельным типам ВКР. Например, для элемента $v_n \in V$, который запрашивает больше ВКР типа M по сравнению с ВКР типа C (несбалансированность по ВКР типа M) может быть выделен только элемент $F_k \in F$ в котором ВКР типа M используется меньше, чем ВКР типа C.

Метод основан на проецировании куба NRC на плоскость, перпендикулярную его главной диагонали (рисунок 2.2).

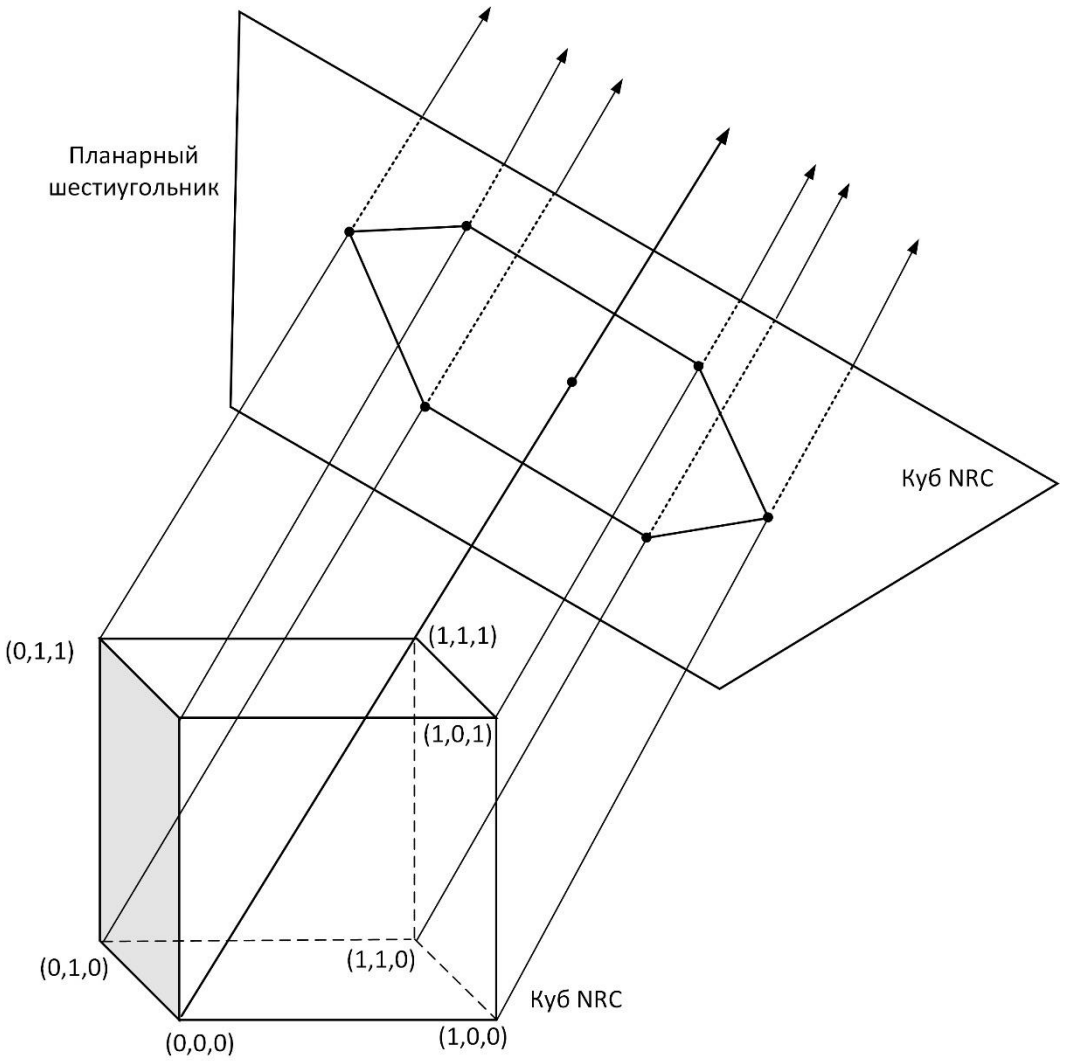


Рисунок 2.2 – Принцип получения планарного шестиугольника ресурсов для куба NRC

Очевидно, что углы куба NRC с координатами (0, 0, 0) и (1, 1, 1) будут находиться в центре координат шестиугольника ПШР.

Тогда можно делать проекцию вершин векторов ресурсов ВНР, ВВР и $ВЗР_{V_n}$ (рисунок 2.1) на плоскость проекции ПШР. При этом проекция вектора ВНР будет находиться в центре ПШР, а проекции векторов ВВР и $ВЗР_{V_n}$ будут находиться в пределах ПШР.

С целью получения размеров векторов ВВР и $ВЗР_{V_n}$, шестиугольник ПШР делится на шесть конгруэнтных друг другу треугольников (в [93] именуются треугольники ресурсов (ТР)) (рисунок 2.3).

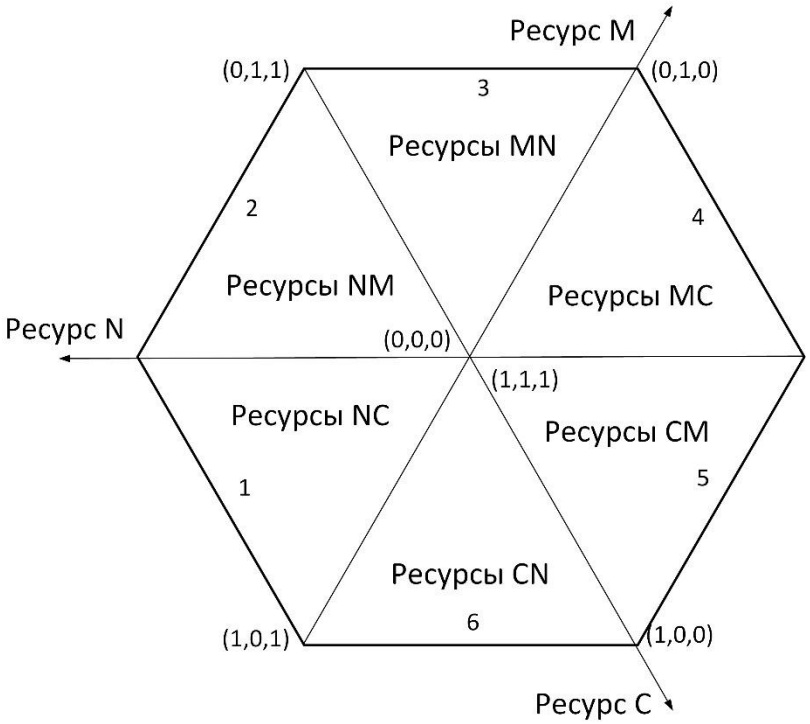


Рисунок 2.3 – Представление шестиугольника ПШР в виде конгруэнтных треугольников ТР

Из рисунка 2.3 видно, что, например, ТР № 5 – «Ресурсы CM» представляет собой область, где для вектора ВВР выполняется неравенство $C>M>N$.

Очевидно, что треугольники ТР можно использовать, как для определения дисбаланса между типами ВКР, так и для его устранения. Так, при получении равенства тех или иных типов ВКР в рамках ТР, его нужно разделить, например, на основе некоторого приоритета использования типа ВКР, определяемого потребителем ВМ $v_n \in V$ (например, ВМ с меньшим числом процессорных ядер (ВКР типа C) и большим объемом оперативной памяти (ВКР типа M).

Таким образом, при назначении элемента $v_n \in V$ на некоторый элемент $F_k \in F$ можно определить соседнее подмножество ТР, удовлетворяющее запросу по всем типам ВКР.

Очевидным недостатком использования ПШР при решении задач балансировки нагрузки или перераспределения элементов множества F, является отсутствие данных о размере вектора ВНР (центр шестиугольника ПШР). В [92]

для решения этой проблемы предлагается введение мнимых величин ВНР, относящиеся к трем классам:

1. H (High) – высокое использование ВКР (хотя бы один из типов ВКР (C, M, N) имеет высокий уровень использования).

2. M (Middle) – среднее использование ВКР.

3. L (Low) – низкое использование ВКР.

Фактически, с точки зрения геометрической интерпретации, это означает изменение нормированного размера куба NRC. Используя предложенные классы наличия ВКР, можно достаточно быстро определить нахождение точки вектора ВЗР на шестиугольнике ПШР, отнеся ее к тому или иному треугольнику ТР путем максимизации значения того или иного типа ВКР. То есть, можно рассматривать 18 уровней использования типов ВКР (шесть треугольников ТР по три класса величины вектора ВНР).

Такой подход позволяет представить степень дисбаланса типов ВКР, как расстояние ВВР каждого типа ВКР по отношению к главной диагонали куба NRC, что фактически соответствует вектору ВДР. Для его вычисления предлагается следующий подход. Пусть величина вектора ВВР равна $C\hat{x}+M\hat{y}+N\hat{z}$, где x, y и z – соответствующие оси координат трехмерного пространства. Тогда проекция этого вектора на главную диагональ НКР будет равна:

$$\left(\frac{1}{\sqrt{3}}*C+\frac{1}{\sqrt{3}}*M+\frac{1}{\sqrt{3}}*N\right)\left(\frac{1}{\sqrt{3}}\hat{x}+\frac{1}{\sqrt{3}}\hat{y}+\frac{1}{\sqrt{3}}\hat{z}\right) \quad (2.11)$$

Правая часть выражения – единичный вектор главной диагонали куба NRC, а левая – величина проекции вектора ВВР на этот вектор. Это выражение можно упростить:

$$\left(\frac{C+M+N}{3}\hat{x}+\frac{C+M+N}{3}\hat{y}+\frac{C+M+N}{3}\hat{z}\right) \quad (2.12)$$

Тогда величину вектора ВДР можно определить выражением:

$$\left(C-\frac{C+M+N}{3}\right)\hat{x}+\left(M-\frac{C+M+N}{3}\right)\hat{y}+\left(N-\frac{C+M+N}{3}\right)\hat{z} \qquad (2.13)$$

Как можно увидеть из выражений 2.11-2.13, подход на основе использования шестиугольника ПШР позволяет выполнять расчет степени дисбаланса по всем типам ВКР с использованием простых преобразований векторной алгебры в двумерной проекции.

Очевидными недостатками рассмотренного метода являются:

- возможность его использования на этапах проектирования и/или модернизации ЦОД (статическое размещение ВМ);

- возможность алгоритмической реализации только на основе «жадных» стратегий, которые в ряде случаев дают большие погрешности вычислений.

2.3.2 Метод векторной точки

В [92] предлагается вариант использования ресурсного представления в виде куба NRC, который именуется методом векторной точки (Vector Dot).

В основе метода векторной точки для выбора целевого элемента $F_k \in F$ при размещении на нем элемента $v_n \in V$ лежит вычисление скалярного произведения векторов ВВР и $ВЗР_{V_n}$. В методе векторной точки это значение является метрикой и именуется EDP (extended dot product).

При этом выбирается элемент $F_k \in F$, величина вектора ВВР которого дает наименьшее значение метрики EDP. Такой подход предполагает, что назначение элемента $v_n \in V$ выполняется на тот элемент $F_k \in F$, для которого величина вектора $ВЗР_{V_n}$ дополняет величину вектора ВВР. Это обеспечивает относительно сбалансированное использование всех трех видов ВКР. Так, например, элемент $v_n \in V$ с запросом небольшого значения ресурса типа C и, одновременно, с запросом большого значения ресурса типа M будет назначен на элемент $F_k \in F$, который имеет большую загрузку ресурса типа C, но небольшой используемый

объем ресурса типа M. В векторном представлении это означает, что для такого элемента $v_n \in V$ среди всех возможных элементов множества F (пространство решений) будет выбран элемент $F_k \in F$, вектор ВВР которого составляет наибольший угол с вектором $ВЗР_{V_n}$. При этом, как и для метода шестиугольника ПШР, алгоритмическая реализация метода векторной точки является вариантом «жадной» стратегии, не всегда обеспечивающей выбор оптимального решения.

2.3.3 Алгоритм XEN SandPiper

В [95] представлен алгоритм SandPiper – одно из наиболее известных решений в области автоматизации мониторинга дисбаланса ресурсов, реализованный на базе гипервизора первого типа XEN.

В основе алгоритма SandPiper лежит предположение о том, что общий выделенный объем всех трех типов ВКР (TUV – Total Utilization Volume) определяется, как произведение $(C \times M \times N)$, где C, M и N – соответствующие нормированные значения трех типов ВКР.

Исходя из этого, алгоритм принимает решение о назначении элемента $v_n \in V$ на целевой элемент $F_k \in F$ на основе метрики, именуемой sand_volume. Она задается следующим выражением:

$$\text{sand\_volume} = \frac{1}{1\text{-}C} \times \frac{1}{1\text{-}M} \times \frac{1}{1\text{-}N} \qquad (2.14)$$

С точки зрения рассмотрения подхода с использованием куба NRC, общее нормирование использование всех трех типов ВКР будет соответствовать занимаемой ими части объема куба NRC.

Тогда алгебраически оставшийся объем куба NRC можно вычислить на основе выражения:

$$\text{e\_volume} = (1\text{-}C) \times (1\text{-}M) \times (1\text{-}N) \qquad (2.15)$$

Таким образом метрика sand_volume, по сути, является обратная величиной значения e_volume.

При обнаружении дисбаланса ВКР алгоритм SandPiper сортирует все элементы множества F в порядке убывания их значений sand_volume.

При этом внутри каждого элемента $F_k \in F$ назначенные ему элементы множества V сортируются в порядке убывания значений так называемого соотношения VSR (sand_Volume to Size Ratio), где Size — объем ресурса типа M, занимаемого каждым элементом $F_k \in F$.

После выполнения процедур сортировки алгоритм SandPiper выбирает элемент $v_n \in V$ с самым высоким значением соотношения VSR в элементе $F_k \in F$ с самым высоким значением метрики sand_volume и принимает решение, возможно ли назначить его на элемент $F_k \in F$ с наименьшим значением метрики sand_volume.

Очевидно, что подобных подход означает, что вновь выбираемый элемент $F_k \in F$, фактически, обеспечивает выполнение запросов элемента $v_n \in V$ по каждому из трех типов ВКР. В случае, если значения значением метрики sand_volume текущего элемента $F_k \in F$ в отсортированном списке недостаточны, то выбирается нижележащий элемент списка.

В случае, если ни один из элементов $F_k \in F$ в отсортированном списке не удовлетворяет запросам на ВКР элемента $v_n \in V$, алгоритм SandPiper рассматривает следующий элемент $v_n \in V$ в отсортированном по величине VSR списке.

Как и в рассмотренных в п.п. 2.3.1 и 2.3.2 методах, алгоритм SandPiper реализует вариант «жадной» стратегии выбора оптимального решения. В [96] на примере двумерного ресурсного пространства (ресурсы C и M) демонстрируется его недостатки.

2.3.4 Метод вычисления нагрузки виртуализированного сервера

В [97] рассматривается всего один метод, широко применяемый для решения задачи балансировки нагрузки.

В основе этого метода лежит использование специализированной метрики «нагрузка виртуализированного сервера» (VSL – Virtualized Server Load).

Согласно [97] метрика VSL вычисляется с использованием выражения:

$$VSL=\sum_{R\in\{C,M,N\}} w_R \times \frac{\sum_{v\in V_{F_k}} v_{RU}}{F_k^{RC}}, \quad (2.16)$$

где R – один из типов ВКР $\{C,M,N\}$, w_R – вес каждого ресурса, V_{F_k} – подмножество элементов множества V, назначенных на элемент $F_k \in F$, v_{RU} – значение ВКР, использованных каждым элементом подмножества V_{F_k} (resource usage), а F_{RC} – ресурсная мощность (resource capacity) элемента $F_k \in F$.

При решении задачи балансировки нагрузки находится значение метрики VSL для всех элементов множества F и вычисляется их среднеквадратичное отклонение.

Так для подмножества $L \subset F$ среднее значение метрики VSL именуется, как μ_L, а его среднеквадратичное отклонение, как σ_L. В качестве метрики, отображающей баланс нагрузки используется значение:

$$C_L = \frac{\mu_L}{\sigma_L} \quad (2.17)$$

В случае перегрузки или дисбаланса нагрузки, элементы подмножества V_{F_k}, которые вызывают снижение C_L, определяются как требуемые для назначения на другие элементы множества F. То есть, фактически, метод вычисления нагрузки виртуализированного сервера стремится подобрать элементы $F_k \in F$ с наиболее близкими значениями VSL, используя обобщенное нормализованное представление всех видов ресурсов, что делает его схожим с точки зрения недостатков с рассмотренным в п. 2.3.3 алгоритмом XEN SandPiper.

2.4 Исследование подходов к решению задач выделения и перераспределения виртуализированных ресурсов центров обработки данных с использованием метаэвристических методов

2.4.1 Основы решения задач комбинаторной оптимизации с использованием метаэвристических методов

Наряду с использованием методов, основанных на «жадных» стратегиях выбора локально оптимального решения из существующего пространства решений (рассмотрены в п.п. 2.2.1-2.2.4), для решения NP-сложных задач дискретной оптимизации, к которым, как было определено в п. 1.4, относится задача векторной упаковки, широкое применение находят методы, которые не формируют гарантированно точного и/или оптимального решения, однако обеспечивают получение решения пригодного (достаточного) для конкретной постановки задачи. Последние нашли широкое применение для поиска решений NP-сложных задач, требующих построения дерева поиска решения экспоненциального размера. К подобным задачам, как было обосновано в п. 1.4, относятся задачи, связанные с перераспределением и/или объединением множества ВМ по множеству ФМ. В [98,99] такие методы относят к метаэвристическим – методам оптимизации, итерационно реализующие прямой случайный поиск для получения субоптимального результата без изначального знания пространства поиска. При этом подобный поиск реализуется на основе некоторых простых правил, именуемых эвристиками – стратегиями, найденными людьми на основе собственного опыта или опыта, заимствованного во внешних системах. Цель метаэвристических методов – эффективное, по какому-либо показателю (обычно – время) исследование пространства поиска для нахождения субоптимальных решений. Существующие метаэвристические методы реализуют алгоритмы от простейших процедур локального поиска до процессов, основанных на обучении.

В [98] представлено обобщение операций метаэвристических методов. К таким операциям относятся:

1. Инициализация – выбор метода нахождения некоторого начального решения $s_0 \in S$, где S – пространство решений.

2. Определение окрестности каждого решения $s \in S$ и множества N(s) связанных с ними переходов $\left\{N^s_1, N^s_2, ..., N^s_q\right\}$.

3. Определение подмножества переходов $C(s) \subseteq N(s)$ для каждой итерации поиска – кандидатов поиска. В зависимости от выбранного метода подмножество $C(s)$ может быть фиксированным для всех итераций, или обновляемым при каждой новой итерации поиска.

4. Определение критерия принятия наилучшего решения $\tilde{s} = \text{argopt}\left\{ f(s, N^s_j); N^s_j \in C(s)\right\}$.

5. Определение критерия останова итераций. Обычно показателями для выбора этого критерия являются: время выполнения алгоритма, максимальное количество итераций или темп улучшения решения $\tilde{s}$ на каждой итерации.

Как указывается в [99,100] метаэвристические методы объединяют некоторое подмножество эвристик в рамках более высокоуровневых алгоритмов. К наиболее известным метаэвристическим методам [99,100] относят:

- локальный поиск с итерациями;
- алгоритм табу-поиска (или вероятностного поиска с запретами);
- эволюционные вычисления, например, генетические алгоритмы;
- метод имитации отжига;
- алгоритмы оптимизации муравьиной колонии, и др.

Использование того или иного метаэвристического метода зависит от множества факторов, к основным из которых следует отнести:

- класс оптимизационных задач, для которых может быть адаптирован метаэвристический алгоритм;
- необходимая (достаточная) точность получаемого решения $\tilde{s}$;
- временные характеристики реализации алгоритма;
- вычислительная мощность, требуемая для реализации алгоритма.

В большинстве своем существующие метаэвристические методы развивались и совершенствовались для решения классов задач комбинаторной оптимизации на графовых структурах, таких, например, как задача поиска кратчайшего пути (дерева кратчайших путей), задача коммивояжера (TSP – travelling salesman problem), задача о назначениях, задача поиска максимального потока [100,101,102].

Кроме того, использование ряда метаэвристик хорошо зарекомендовали себя в решении класса задач упаковки [102,103]. В частности, ряд актуальных исследований [104,105,106] показывает эффективность использования алгоритмов оптимизации муравьиной колонии (далее ACO – ant colony optimization) для решения задач, как двумерной упаковки контейнеров, так и их d-мерных вариантов.

Поскольку в исследовании (см. п. 1.2-1.4) выдвигается гипотеза о возможности отнесения класса задач, связанных с перераспределением и/или объединением множества ВМ по множеству ФМ (см. п. 1.4) к классу задач многомерной векторной упаковки контейнеров (Multi-dimensional Vector Packing Problem) для достижения цели исследования предлагается использовать модифицированный вариант метаэвристического алгоритма ACO.

2.4.2 Обоснование применения метаэвристического алгоритма ACO для решения задачи перераспределения и/или объединения виртуальных машин в виртуализированных центрах обработки данных

2.4.2.1 Основы метаэвристического алгоритма ACO

Как указывается в [99,103,106] методы ACO обобщают эвристику взаимодействия особей некоторых видов муравьев при определении наиболее коротких путей от колонии к источникам пищи. Особенностью такой эвристики является отсутствие централизованных методов обмена информацией о маршрутах движения и организация децентрализованного взаимодействия муравьев между собой с использованием химического соединения – феромона, который каждый муравей оставляет по пути своего следования. Использование феромона позволяет

заменить случайное блуждание муравьев вероятностно упорядоченным вдоль путей, отмеченных феромоном. Особенностью феромона, как маркера путей получения решения является возможность его постепенного испарения на малоиспользуемых маршрутах и, наоборот, усиления на маршрутах, являющихся субоптимальным решением.

В общем виде метаэвристика ACO, применительно к решению задач комбинаторной оптимизации, представлена на рисунке 2.4.

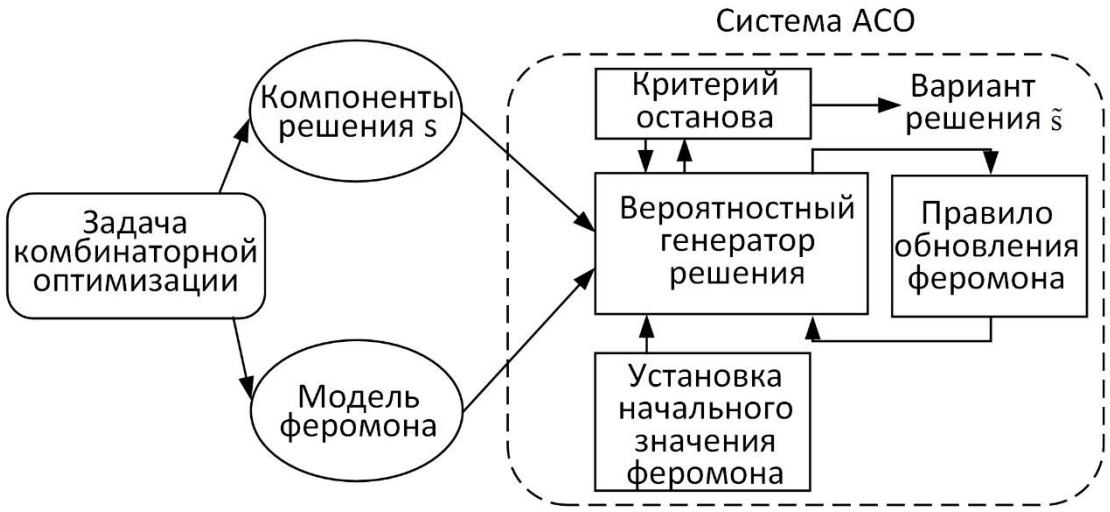


Рисунок 2.4 – Обобщенное представление метаэвристики Ant Colony Optimization

В системе ACO поиск субоптимального решения производится параллельно некоторым множеством $ИМ=\{ИМ_1,ИМ_2,...,ИМ_m\}$ искусственных муравьев (ИМ) – программных агентов, реализующих следующую эвристику поведения реального муравья:

1. k-й ИМ, находящийся в некотором i-м узле имеет память J_{ik} – список узлов по маршруту его следования, которые он может пройти. Выполнив переход в новый узел, ИМ удаляет его из J_{ik}. По завершении итерации поиска список J_{ik} очищается для новой итерации. В ряде случаев рассматривается инверсный список узлов, которые ИМ не должен выбирать на новом шаге итерации – taboo list.

2. Правило выбора k-м ИМ нового узла j для перехода из узла i на некотором шаге t является вероятностным и зависит от двух параметров:

- $\eta_{i,j}$ - величины значимости ребра (в ряде источников именуется «зрением» ИМ), являющейся обратной величине веса ребра $\omega_{i,j}$ (расстояния между узлами i и j) - $\eta_{ij} = \frac{1}{\omega_{ij}}$;

- τ_{ij} - уровня феромона ребра между узлами i и j.

В общем виде вероятность перехода k-го ИМ из узла i в узел j определяется выражением:

$$p_{ij,k}(t) = \begin{cases} \dfrac{\left[\tau_{ij}(t)\right]^{\alpha} \times \left[\eta_{ij}(t)\right]^{\beta}}{\sum\limits_{l \in J_{ik}} \left[\tau_{il}(t)\right]^{\alpha} \times \left[\eta_{il}(t)\right]^{\beta}} & \text{при } j \in J_{ik} \\ 0 & \text{в противном случае} \end{cases} \quad (2.18)$$

где α и β – эмпирические параметры, определяющие степень влияния уровня феромона.

Так при α=0 ИМ стремиться выбирать кратчайшее ребро, то есть алгоритм вырождается в «жадную» стратегию, а при β=0 – ИМ выбирает ребро, имеющее наибольший уровень феромона τ_{ij}. Многочисленные эксперименты показали, что наиболее подходящими для варьирования значения параметров α и β находятся в диапазоне значений от 1 до 3 [106,108].

В начале каждой итерации вероятности перехода ИМ из i-го узла в соседние узлы равны. Однако с течением времени, вероятность p_{ij} выбора наиболее короткого пути между узлами i и j возрастает, поскольку прирост уровня феромона $\Delta\tau_{ij,k}$ происходит обратно пропорционально длине L_k некоторого маршрута T_k и определяется выражением:

$$\Delta\tau_{ij,k}(t)=\begin{cases}\dfrac{Q}{L_k} & \text{при } (i,j)\in T_k\\ 0 & \text{в противном случае}\end{cases} \qquad (2.19)$$

где Q – эмпирический параметр, определяемый исследователем и имеющий смысл порядка длины оптимального пути.

С целью соблюдения правил эвристики, для путей, не являющихся кратчайшими и вероятность выбора которых стремится к 0, требуется определить правило уменьшения уровня феромона (функция испарения) на шаге t+1. Оно задается следующим выражением:

$$\tau_{ij}(t+1)=(1-\delta)\times\tau_{ij}(t)+\sum_{k=1}^{m}\Delta\tau_{ij,k}(t), \qquad (2.20)$$

где параметр $0\le\delta\le1$ является коэффициентом испарения, определяющим величину уровня феромона, оставшегося после каждой итерации. Этот параметр может варьироваться исследователем.

В общем виде алгоритм функционирования ACO состоящей из m ИМ, представлен на рисунке 2.5.

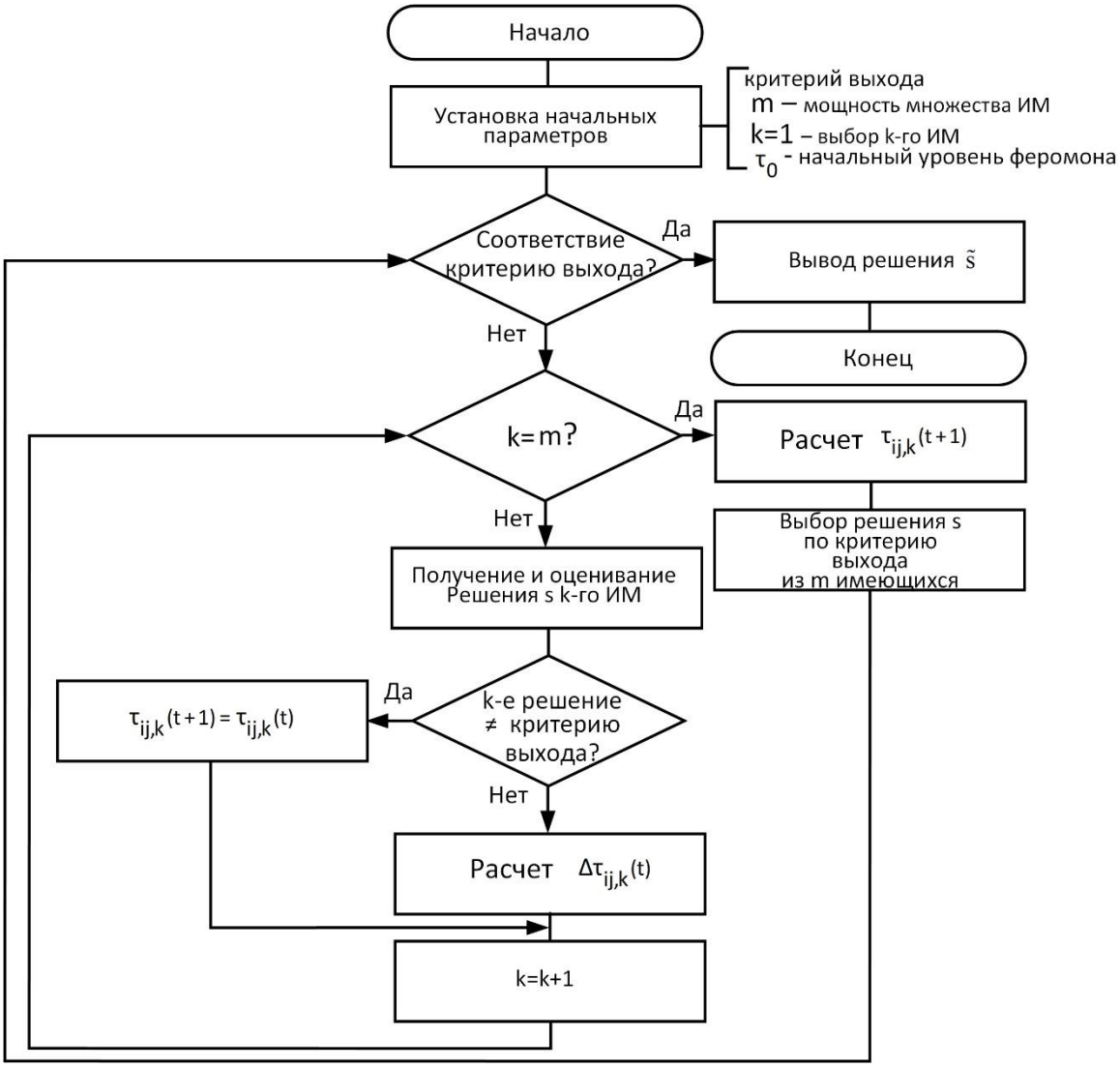


Рисунок 2.5 – Схема алгоритма обобщенной реализации метаэвристики ACO

Представленный на рисунке 2.5 алгоритм является классическим алгоритмом реализации метаэвристического метода «муравьиная система» (AS – Ant System), предложенного в [99]. Очевидно, что его применение для решения различного класса задач комбинаторной оптимизации, потребовало внесения ряда модификаций.

Наиболее известными из них являются [99,106,108]:

- EAS (Elitist Ant System) – система использования «элитных» ИМ – агентов, которые, проходя ребра, входящие в короткие пути, с более высокой вероятностью будут находить еще более короткие пути, что позволяет еще больше увеличивать уровень феромона на ребрах, входящих в их состав;

- Ant-Q (Ant Q-learning System) – система, использующая принцип обучения с подкреплением. В ее основе лежит специальная таблица ребер (Q-table), в которой каждому ребру сопоставляется уровень полезности перехода по нему. Значение уровня полезности итерационно пересчитывается в зависимости от значений уровней полезности перехода по ребрам, которые были получены на основе предшествующего поиска всех возможных новых состояний;

- MMAS (Max-min Ant System) – система, основанная на повышении уровня феромона только на путях, являющихся локально оптимальными. Это достигается введением верхней и нижней границ уровней феромона для каждого ребра $\tau_{min} \le \tau_{ru} \le \tau_{max} \forall \tau_{ru}$;

- AS-Rank (Ranked Ant System) – система, основанная на ранжировании множества ИМ по критерию длины пройденного ими пути. Для глобально оптимального пути уровень феромона увеличивается с весовым коэффициентом w, а для ребер путем, полученных наиболее высокоранговыми ИМ с весовым коэффициентом $w-1$, то есть k-й высокоранговый ИМ будет увеличивать уровень феромона с весовым коэффициентом $w-k$. Правило обновления феромона на шаге $t+1$ определяется выражением: $\tau_{ij}(t+1)=\delta\times\tau_{ij}(t)+w\times\Delta\tau_{ij}^{global}(t)+\sum_{k=1}^{w-1}(k-w)\times\Delta\tau_{ij}^{k}(t)$;

- ACS (Ant Colony System) – система, в которой обновление уровня феромона на каждом ребре выполняется не только в конце итерации алгоритма, но и при каждом переходе ИМ на новый узел. При этом повышение уровня феромона выполняется только на кратчайшем из найденных путей в соответствии с выражением $\tau_{ij}(t+1)=\delta\times\tau_{ij}(t)+(1-\delta)\times\Delta\tau_{ij}^{opt}(t)$. При этом правило перехода ИМ на новый узел усложняется за счет выбора в соответствии с однородным распределением и носит название псевдослучайного пропорционального правила, где выбор j-го узла происходит, либо с вероятностью, определенной в классическом алгоритме AS, либо ИМ безусловно выбирает лучшее ребро, с точки зрения его длины и уровня феромона.

Проведенный анализ исследований [106,108,109] показал, что для решения задач многомерной упаковки наиболее лучший результат показывает алгоритм ACS.

Очевидно, что, как и в случае применения любого иного метаэвристического метода, для выбранного алгоритма ACS требуется его предварительная интерпретация к условиям решаемой оптимизационной задачи. В частности, требуется:

- определить физический смысл величины уровня феромона;

- интерпретировать задачу нахождения оптимального пути, к условиям решаемой оптимизационной задачи. Применительно к научной задаче настоящего исследования оптимизационная задача относится к классу задач многомерной (исходя из предложенных в п. 2.1.2.1 решений – трехмерной) векторной упаковки элементов множества V на ресурсном кубе NRC элементов множества F.

2.4.2.2 Общий вид комбинаторной постановки задачи перераспределения ВМ на множестве ФМ ЦОД как задачи векторной упаковки

Исходя из предложенного в п. 2.1.2.1 векторного представления виртуализированных (C, M, N) ФМ ЦОД, как трехмерных векторов ВНР и ВВР, вектора ВОР, являющегося векторной разностью ВНР и ВВР, потребностей в них активных ВМ ЦОД, определяемых вектором ВЗР, а также вектора ВДР – ортогональной проекции векторов ВВР и ВОР на вектор ВНР и определяющего уровень дисбаланса виртуализированных ВКР, определим ВКР элемента $F_i \in F$ представим 3-мерным вектором ВВР:

$$P_i = \langle P_i^C, P_i^M, P_i^N \rangle, \tag{2.21}$$

где P_i^C – полнота ВКР типа C (процессорное время), P_i^M – полнота ВКР типа M (объем памяти), P_i^N – полнота ВКР типа N (сетевой ввод-вывод).

Соответственно, потребность в ВКР всех типов со стороны элемента $v_j \in V$ представим 3-мерным вектором ВЗР:

$$D_j = \langle D_j^C, D_j^M, D_j^N \rangle, \tag{2.22}$$

где D_j^m – потребность в m-м (C, M, N) ВКР, которым располагает элемент $F_i \in F$.

Определим 3-мерный вектор выделения (utilization) ВКР (ВВР) и булеву матрицу миграции (ММ) элементов множества V по элементам множества F. Вектор ВВР представим как:

$$U_i^k = \langle U_i^C, U_i^M, U_i^N \rangle, \tag{2.23}$$

где U_i^k вычисляется с использованием выражения:

$$U_i^k = \sum D_j^m \text{ для } \forall MM_{i,j} = 1 \tag{2.24}$$

где $MM_{i,j}$ – матрица миграции (размещения) элемента $v_i \in V$ на элемент $F_j \in F$, которая в операторном виде представляется выражением:

$$MM_{i,j} = \begin{cases} 1, \text{ если } v_i \to F_j \\ 0, \text{ в противном случае} \end{cases} \tag{2.25}$$

Выберем матрицу $MM_{i,j}$ в качестве элемента локального решения *s* обобщенного алгоритма ACS, а матрицу MM_G (от Global), как элемент результирующего (глобального) решения $\tilde{s}$ (рисунок 2.5).

Тогда, исходя из выражения 2.10 (п. 2.2) накладные расходы на ЖМ ВМ матрицы $MM_{i,j}$ определим как $HP_{MM_{i,j}}$, а целевую функцию f, обеспечивающую получение оптимального глобального решения s, определим, как максимизацию соотношения:

$$\max f(MM_{i,j}) = \frac{N_{F_{HA}}}{HP_{MM_{i,j}}}, \tag{2.26}$$

где $N_{F_{HA}}$ – количество неактивных ФМ ($F_{HA} \subset F$). Физический смысл целевой функции можно интерпретировать, как увеличение количества ФМ, на которые не распределена ни одна ВМ и которые можно исключить из схемы энергопотребления ЦОД и/или использовать их для нового цикла выделения ВМ, с учетом влияния накладных расходов на миграцию i-й ВМ на j-ю ФМ.

Тогда целевую функцию глобального решения $\tilde{s}$ определим как:

$$\max f(MM_G) = \frac{N_{F_{HA}}}{HP_{MM_G}}, \tag{2.27}$$

Интерпретируем понятие величины уровня феромона $\tau(t)$ на t-м шаге. Зададим $\tau(t)$ матрицей размерности $n \times m$, где n и m – соответственно мощности множеств V и F.

Поскольку для каждого компонента решения s (выбранной пары {v, F} элементов множеств V и F) величина $\tau(t)$ фактически определяет меру желательности выбора этого компонента в ходе получения локального решения s, то на нулевом шаге итерации начальное значение величины уровня феромона τ_0 для каждого из компонентов решения должно быть одинаковым (одинаковая желательность выбора любого компонента решения). Получение такого значения τ_0 возможно с использованием любого из вариантов «жадных» стратегий. Например, в [99] предлагается использование алгоритма «Первый подходящий с упорядочиванием» (FFD – First-Fit-Decreasing), широко используемого при решении задач упаковки [112].

Определим прирост уровня феромона $\Delta\tau_{vF,a}(t)$ для a-го ИМ, как:

$$\Delta\tau_{vF,a}(t) = \begin{cases} f(MM_G) \text{ при } (v,F) \in MM_G \\ 0 \quad \text{в противном случае} \end{cases}, \tag{2.28}$$

Соответственно, определим правило обновления уровня феромона на шаге (t+1) как:

$$\tau_{vF,a}(t+1) = (1-\delta) \times \tau_{vF,a}(t) + \delta \times \Delta\tau_{vF,a}, \tag{2.29}$$

Согласно п. 2.4.2.1 интерпретируем переменную $\eta_{v,F}$, как меру выгоды от решения по конкретной паре элементов множеств V и F.

Поскольку целью разрабатываемого алгоритма является увеличение величины подмножества $F_{НА} \subset F$ за счет сбалансированного по всем видам ВКР назначения элементов множества V, а также с учетом накладных расходов $ВКР^{нр}_{жм}$ на ЖМ элементов множества V, то представим $\eta_{v,F}$, как сбалансированную по всем измерениям величину вектора ВДР (выражение 2.30):

$$\eta_{v,F} = \omega \times \left| \lg ВДР' \right| + (1-\omega) \times ВВР^{v}_{F_k}, \qquad (2.30)$$

где ω - параметр важности полученного решения о сбалансированном использовании ресурсов R относительно их общего использования (вводится в [107]), а $ВВР^{v}_{F_k}$ - векторное представление выделенного m-го ресурса вектора R (выражение 2.31) для элемента $F_k \in F$, при назначении ему элемента $v_i \in V$:

$$ВВР^{v}_{F_k} = \sum_{R_m \in R} (U^{v}_{F_k} + D^{v}_{F_k}), \qquad (2.31)$$

где, в свою очередь, $U^{v}_{F_k}$ определяется выражением 2.24, а $D^{v}_{F_k}$ определяет потребности в ресурсах R элемента $v \in V$ при размещении его на элементе $F_k \in F$.

Использование десятичного логарифма величины ВДР′ связано с необходимостью получения более высоких эвристических значений параметров v_n и F_k.

Фактически $\eta_{v,F}$ является обратной величиной $\delta_{v,F}$ – длины вектора ВДР, то есть $\eta_{v,F} = \frac{1}{\delta_{v,F}}$.

Определим правило приятия локального решения s агентом ИМ. Интерпретируем шаг приятия решения ИМ, рассмотренный в п. 2.4.2.1, выбора j-го узла в качестве направления из i-го узла применительно к выбору ИМ элемента множества V, размещаемого на элементе множества F.

Тогда некоторый a-й ИМ для принятия решения о выборе элемента $v_n \in V$ для его размещения на элементе $F_k \in F$ использует следующую функцию, определенную в [107], как псевдослучайное правило пропорционального выбора (см. п. 2.4.2.1), которое фактически, благоприятствует выбору сбалансированной по всем измерениям величины вектора ВДР – $\eta_{v,F}$ с наибольшим значением уровня феромона $\tau_{v,F}$, ассоциированного с парой значений параметров v_n и F_k:

$$s = \begin{cases} \operatorname{argmax}_{v \in LV_a(F)} \left\{ \tau_{v_n,F_k} \times \left[\eta_{v_n,F_k} \right]^{\beta} \right\} & \text{при } q \le q_0 \\ S & \text{в противном случае} \end{cases} \qquad (2.32)$$

где:

- q – случайное число, равномерно распределенное в диапазоне $[0,...,1]$;

- q_0 – параметр, удовлетворяющий неравенству $0 \le q_0 \le 1$ и определяющий важность функции выбора пары значений параметров v_n и F_k в сравнении с функцией продолжения поиска подходящей пары.

- β – параметр, имеющий неотрицательное значение и определяющий относительную важность параметра τ_{v_n,F_k} относительно значения η_{v_n,F_k};

- S – (согласно [105]) случайная величина, выбранная в соответствии со следующим распределением вероятности $p_a(v_n, F_k)$:

$$p_a(v_n,F_k) = \begin{cases} \dfrac{\tau_{v_n,F_k} \times \left[\eta_{v_n,F_k} \right]^{\beta}}{\sum_{F_k \in F} \tau_{v_n,F_k} \times \left[\eta_{v_n,F_k} \right]^{\beta}} & \text{при } F_k \in F \\ 0 & \text{в противном случае} \end{cases} \qquad (2.33)$$

Множество $LV_a(F)$ является списком (List of Virtual Machines) элементов $v \in V$, принадлежащим a-му агенту и определяется как:

$$LV_a(F) = \left\{ v \left| \sum_{F=1}^{k} MM_{i,j} = 0 \wedge U_i^k + D_j^m < C_i^k, \forall R_m \in R \right. \right\} \quad (2.34)$$

Соответственно, когда, согласно выражению 2.32, значение $q \le q_0$, то выбирается подходящая пара значений параметров v_n и F_k, имеющая наибольшее значение $\tau_{v_n,F_k} \times \left[\eta_{v_n,F_k} \right]^{\beta}$. В противном случае значение параметра v выбирается с $p_a(v_n, F_k)$ (выражение 2.33).

Предложенная интерпретация метаэвристического метода ACS применительно к задаче размещения элементов множества V на элементах множества F с учетом накладных расходов на ЖМ ВМ позволила перейти к разработке алгоритма перераспределения ВМ, представленного в следующей главе.

2.5 Выводы по главе

В главе рассмотрен подход к разработке математического обеспечения процесса переразмещения ВМ в ЦОД. В рамках решения этой задачи было выполнено моделирование виртуализированных ресурсов ВЦОД. В качестве методологической основы моделирования была выбрана векторная алгебра, в частности, многомерное векторное представление наличия, выделения (утилизации) и запросов на использование виртуализированных ресурсов.

В процессе разработки модели была выдвинута гипотеза о возможности снижения размерности векторного пространства виртуализированных ВКР за счет его трехмерного представления, относительно базовых виртуализированных ресурсов. Был обоснованно выбран подход к представлению векторного пространства ВКР в виде нормализованного ресурсного куба.

В рамках поставленной задачи исследования предложенная векторная модель виртуализированных ВКР была дополнена учетом выделения ВКР алгоритмам ЖМ ВМ.

Анализ алгоритмов ЖМ, а также исследований в области моделирования процесса ЖМ ВМ позволил определить значение накладных расходов на ЖМ конкретной ВМ на конкретную ФМ.

Таким образом, решение задачи переразмещения ВМ с учетом накладных расходов, вносимых конкретными алгоритмами ЖМ, применяемых в гетерогенных ВЦОД можно определить, как задачу комбинаторной оптимизации, связанную с уменьшением величины вектора ВДР, задающего равновесное значение C, M и N измерений нормализованного ресурсного куба.

В рамках решения этой задачи были исследованы известные реализации механизмов переразмещения ВМ, на основе анализа недостатков которых делается обоснованный выбор метаэвристических методов решения задачи комбинаторной оптимизации.

Проведено обобщение операций, реализуемых в рамках метаэвристического подхода, рассмотрены основные метаэвристики. В их рамках для решения задачи переразмещения ВМ, как многомерной векторной задачи комбинаторной оптимизации обоснованно выбрана метаэвристика муравьиной ACO.

В результате этого анализа для разработки алгоритма переразмещения ВМ обоснованно был выбран вариант алгоритма ACS.

ГЛАВА 3. РАЗРАБОТКА АЛГОРИТМА ПЕРЕРАЗМЕЩЕНИЯ ВИРТУАЛЬНЫХ МАШИН ЦЕНТРА ОБРАБОТКИ ДАННЫХ НА ОСНОВЕ МЕТАЭВРИТИКИ МУРАВЬИНОЙ КОЛОНИИ

3.1 Модификация алгоритма муравьиной колонии для решения задачи переразмещения виртуальных машин центра обработки данных

Как и любой алгоритм, основанный на использовании метаэвристики, для решения прикладной задачи его необходимо модифицировать. В данном случае необходимо модифицировать алгоритм муравьиной колонии, основанный на метаэвристике ACS и описанный в п. 2.4, под решение задачи оптимизации размещения ВМ ЦОД, так как изначально алгоритмы муравьиной колонии использовались для поиска кратчайшего пути ввиду своего прямого предназначения – поиска кратчайшего пути к источнику пищи. Именно по этой причине их часто используют при решении задачи коммивояжера, так как они показали свою эффективность в сравнении с другими алгоритмами (например, на основе «жадного» поиска) [108,113].

Алгоритмы муравьиной колонии эффективны благодаря своей способности избегать локальных минимумов и находить глобальные оптимумы через параллельное исследование множества решений. Это достигается за счет комбинации детерминированных и стохастических элементов: детерминированные элементы включают в себя правила выбора на основе следов феромонов и эвристической информации, тогда как стохастические элементы вводят элемент случайности и изменчивости, обеспечивая исследование новых путей. Такая гибкая структура делает алгоритмы муравьиной колонии чрезвычайно востребованными в сложных и динамически изменяющихся задачах, как задача коммивояжера и задача переразмещения ВМ.

Поэтому за основу было взято решение задачи коммивояжера с помощью алгоритма муравьиной колонии для интерпретации данного алгоритма под решение задачи переразмещения ВМ ЦОД.

3.1.1 Алгоритм муравьиной колонии для решения задачи поиска кратчайшего пути

Впервые алгоритм муравьиной колонии был предложен для решения задачи поиска кратчайшего пути [108,110], по причине того, что сама задача поиска кратчайшего пути является задачей поиска кратчайшего гамильтонова пути в графе, что схоже с задачей поиска кратчайшего пути к источнику пищи муравьями, а также по тому, что задача коммивояжера является одной из наиболее изученных NP-трудных комбинаторных задач.

Задача поиска кратчайшего пути представляет собой задачу поиска минимального по стоимости замкнутого маршрута по всем вершинам без повторений на полном взвешенном графе с n вершинами. В контексте задачи вершины графа представляют собой города, которые необходимо посетить, а веса рёбер указывают на расстояния (длины) или затраты на проезд.

Формально можно представить следующим образом: дан ориентированный граф G = (S, U), где множество вершин S, |S| = n – представляет собой множество городов; множество ребер U, |U| = m – возможные пути между городами. Вес ребра $C_{ij} > 0$ эквивалентен времени проезда между городами (смежными вершинами графа). Через d_{ij} обозначается длина дуги $(i,j) \in U$, которая равна расстоянию между городами i и j, $(i,j) \in V$. Необходимо найти гамильтонов цикл минимальной длины.

Алгоритм муравьиной колонии на основе метаэвристике ACS работает поэтапно [99,106], выполняя следующие итерации:

1. **Инициализация.** Формируется виртуальная колония муравьев, при этом обычно число муравьев совпадает с количеством вершин графа. Задается начальное положение для каждого муравья. На всех ребрах устанавливается изначальное количество феромона τ_0.

2. **Поиск решения.** Каждый муравей прокладывает свой маршрут, неоднократно применяя правило перехода состояний. Для этого он поочередно посещает вершины, пока не столкнется с тупиком или не вернется к исходной

вершине. Выбор муравья определяется уровнем оставленного феромона и расстояниями между вершинами.

3. **Локальное обновление феромонов.** Будет ли муравей оставлять феромоны, зависит от успеха в выполнении задачи. В случае, если задача не выполнена, феромоны на пути отсутствуют. Если же задача выполнена, то муравей оставляет феромоны согласно длине маршрута. Таким образом усиливается путь для последующих муравьев и учитывается локальная информация о качестве маршрута.

4. **Определение лучшего решения.** В случае, если муравей справился с задачей, его маршрут и протяженность этого маршрута будут сохранены (при условии, что протяженность короче уже установленной).

5. **Глобальное обновление феромонов.** Когда все муравьи завершат свою задачу, осуществляется общее обновление феромонов на рёбрах графа, которое учитывает локальные изменения феромонов и их испарение.

6. **Повторение.** Процесс формирования решений и обновления феромонного следа повторяется заданное количество раз или до достижения условия остановки. С каждым новым циклом итераций муравьев качество решений, как правило, возрастает.

Учитывая особенности задачи поиска кратчайшего пути этапы алгоритма муравьиной колонии имеют следующий вид [99,103]:

1. **Инициализация.**

При решении задачи поиска кратчайшего пути работу настоящих муравьев имитируют ИМ, количество которых соответствует количеству вершин графа.

В качестве исходной информации задается множество вершин графа S, множество ребер графа U, множество ИМ A, начальный уровень феромона τ_0.

Выходной информацией является наилучший маршрут T_{opt} и его длина L_{opt} .

2. **Поиск локального решения.**

На каждой итерации t алгоритма муравьиной колонии, (при $t \le t_{max}$, где t_{max} – максимальное число итераций) ИМ строят k решений задачи коммивояжера, выполняя для этого две основные операции: переход ИМ в вершину j и выбор дуги (i, j) для включения ее в свой оптимальный путь.

Для обеспечения баланса между исследованием новых ребер и использования априорных знаний было разработано усовершенствованное правило [112] перехода ИМ из одной вершины в другую, на основании которого принимается локальное решение s о переходе ИМ (выражение 3.1):

$$s = \begin{cases} \operatorname{argmax}\left\{\tau_{ij}(t) \times \left[\eta_{ij}(t)\right]^{\beta}\right\} & \text{при } q \le q_0 \\ S & \text{в противном случае} \end{cases} \qquad (3.1)$$

где q – случайно сгенерированное число из промежутка [0…1];

q_0 – заданный параметр ($0 < q_0 < 1$);

S – это вершина, переход к которой осуществляется при использовании правила 2.18.

Переход между вершинами (i, j) ИМ осуществляет, двигаясь по графу задачи от одной вершины к другой согласно некоторому вероятностному закону (выражение 3.2).

$$p_{ij,k}(t) = \begin{cases} \dfrac{\left[\tau_{ij}(t)\right]^{\alpha} \times \left[\eta_{ij}(t)\right]^{\beta}}{\sum\limits_{l \in J_{ik}} \left[\tau_{il}(t)\right]^{\alpha} \times \left[\eta_{il}(t)\right]^{\beta}} & \text{при } j \in J_{ik} \\ 0 & \text{в противном случае} \end{cases} \qquad (3.2)$$

где $\left[\tau_{ij}(t)\right]^{\alpha} \times \left[\eta_{ij}(t)\right]^{\beta}$ – желание ИМ перейти из вершины i в вершину j;

$\sum\limits_{l \in J_{ik}} \left[\tau_{il}(t)\right]^{\alpha} \times \left[\eta_{il}(t)\right]^{\beta}$ – суммарное желание перехода ИМ;

α – коэффициент контроля влияния количества феромонов;

β – коэффициент контроля влияния расстояния между вершинами (i, j);

l – количество доступных вершин.

Вероятность перехода ИМ в вершину j возможна при условии, что вершина j еще не была посещена ИМ.

Выбор дуги (i,j) осуществляется на основании желания перехода ИМ в вершину j: $\left[\tau_{ij}(t)\right]^{\alpha} \times \left[\eta_{ij}(t)\right]^{\beta}$ и зависит от уровня феромона τ_{ij} на рассматриваемом ребре (i,j) и эвристической информации η_{ij}, определяющую степень предпочтительности выбора данного ребра.

Выбор вершины j происходит случайным образом, вне зависимости от желания ИМ. Тем не менее, логично предположить, что вероятность перехода из вершины i в вершину j прямо пропорциональна «желательности» этого перехода.

ИМ повторяет эти шаги до тех пор, пока не будет построен гамильтонов цикл. Начальная вершина выбирается случайным образом с равномерным распределением. Таким образом, построенное ИМ решение представляет собой список пройденных дуг.

На основе полученных локальных решений s формируется глобальное решение $\tilde{s}$.

3. **Локальное обновление феромонов.**

Количество откладываемого феромона обратно пропорционально расстоянию L_k, пройденному k-м ИМ на итерации t некоторого маршрута T_k (выражение 3.3).

$$\Delta\tau_{ij,k}(t)=\begin{cases}\dfrac{Q}{L_k} & \text{при } (i,j)\in T_k \\ 0 & \text{в противном случае}\end{cases} \qquad (3.3)$$

Чем короче выбранный маршрут, тем больше феромона будет отложено на его ребрах, следовательно, большее количество ИМ будет включать его в свои списки маршрута.

Все значения феромонов, добавленных по рёбрам, суммируются всеми муравьями и сохраняются в специальный буфер для того, чтобы учесть локальное обновление феромонов в процессе глобального обновления.

4. **Определение лучшего решения**

Лучшим решением $\tilde{s}$ является продолжительность оптимального расстояния L_{opt} пути T_{opt} после проверки всех $L_k(t)$ на лучшее решение (выражение 3.4).

$$\tilde{s} = \begin{cases} L_{opt} \text{ при } L_{opt} < L_k(t) \\ L_k(t) \quad \text{в противном случае} \end{cases} \tag{3.4}$$

где $L_k(t)$ продолжительность наилучшего тура в текущей итерации испытания.

5. **Глобальное обновления феромонов**

В каждый последующий момент времени происходит испарение феромона на ребрах графа $(i,j) \in T_k$ по правилу (выражение 3.5):

$$\tau_{ij}(t+1) = (1-\delta) \times \tau_{ij}(t) + \sum_{k=1}^{m} \Delta\tau_{ij,k}(t) \tag{3.5}$$

где $\tau_{ij}(t)$ – количество феромона на текущей итерации;

δ – скорость испарения феромона, определяемая исследователем в пределе [0…1];

$\Delta\tau_{ij,k}(t)$ – локальное обновление феромона на ребре между вершинами i и j.

Величина уровня феромона t_{ij} определяет вес ребра на графе пути T_k.

3.1.2 Алгоритм муравьиной колонии для решения задачи переразмещения виртуальных машин центра обработки данных

С учетом постановки задачи переразмещения ВМ ЦОД, рассмотренной во второй главе, и модификации алгоритма муравьиной колонии для решения задачи поиска кратчайшего пути, рассмотренной в п. 3.1.1, модификация алгоритма

муравьиной колонии для решения задачи переразмещения ВМ будет выглядеть следующим образом.

Задача переразмещения ВМ ЦОД также представляет собой задачу поиска оптимального решения. Но в отличии от задачи поиска кратчайшего пути, в которой рассматривается один параметр, задача переразмещения ВМ ЦОД относится к многомерным задачам упаковки [94,104,105,115], где оптимальным решением является такое размещение определённого количества элементов (ВМ) в минимальном количестве контейнеров (ФМ), которое не превышает их вместимость. То есть, требования ВМ по ресурсам (C;M;N) не должны превышать возможности ФМ F_j (выражения 3.6-3.8).

$$\forall F_j \in \{1,...,k\} \sum_{v=1}^{n} MM_{ij} \times C(V_v) \le C(F_F) \quad (3.6)$$

$$\forall F_j \in \{1,...,k\} \sum_{v=1}^{n} MM_{ij} \times M(V_v) \le M(F_F) \quad (3.7)$$

$$\forall F_j \in \{1,...,k\} \sum_{v=1}^{n} MM_{ij} \times N(V_v) \le N(F_F) \quad (3.8)$$

В задаче поиска кратчайшего пути выбирается город для посещения, при чем посетить его можно только один раз. В задаче переразмещения ВМ ЦОД будет выбираться ВМ для размещения на ФМ, при чем одна и та же ВМ может рассматриваться несколько раз в случае отсутствия возможности размещения ее на искомой ФМ.

В отличие от классического алгоритма муравьиной колонии, ориентированного на решение задачи поиска кратчайшего пути, в модифицированном алгоритме переразмещения ВМ предлагается использовать в качестве «кратчайшего пути» (расстояние между (v, F)) значение величины вектора ВДР, определяемую в общем случае как векторная разность проекций на векторы ВНР и ВВР (п. 2.2), так как ВМ представляет собой координату вектора ВВР, а ФМ является одной из координат вектора ВНР (выражение 3.9).

$$ВДР'=\sqrt{(C-\Delta)^2+(M-\Delta)^2+(N-\Delta)^2} \quad (3.9)$$

Соответственно, чем меньше величина (длина) вектора ВДР при выборе ВМ, тем более предпочтителен ее выбор.

Как и в решении задачи коммивояжера в задаче переразмещения ВМ ЦОД работу настоящих муравьев имитируют ИМ, только их количество ИМ будет выбираться исходя из оценки временной сложности разрабатываемого алгоритма.

1. Инициализация

Первоначальным этапом разработанного алгоритма является формирование на основе гетерогенного множества ФМ F гомогенных подмножеств F^g, где g – номер, идентифицирующий тип ГВ (например, Xen = 1, KVM = 2 и т.д.). По причине того, что не все гипервизоры совместимы для проведения процесса ЖМ ВМ.

Рассматриваемые далее процедуры разработанного алгоритма переразмещения ВМ применяются параллельно к элементам каждого из подмножеств F^g.

2. Поиск локального решения.

На каждой итерации t алгоритма муравьиной колонии ИМ строят решения задачи переразмещения ВМ в виде матрицы миграции $MM_{i,j}$, являющейся элементом локального решения s данной задачи в операторном виде представляемая выражением 3.10:

$$MM_{i,j}=\begin{cases}1, \text{ если } v_i \rightarrow F_j \\ 0, \text{ в противном случае}\end{cases} \quad (3.10)$$

Локальное решение ИМ s о включении ВМ v в матрицу миграции $MM_{i,j}$ также как и при решении задачи поиска кратчайшего пути выполняется на основе «псевдо-случайного правила» (выражение 3.11) [111].

$$s=\begin{cases}\underset{v\in LV_a(F^g)}{\operatorname{argmax}}\left\{\tau_{v,F^g}\times\left[\eta_{v,F^g}\right]^{\beta}\right\} & \text{при } q\le q_0\\ S & \text{в противном случае}\end{cases} \qquad (3.11)$$

где q – случайно сгенерированное число из промежутка [0…1];

q_0 – заданный параметр ($0<q_0<1$);

β – параметр, имеющий неотрицательное значение и определяющий относительную важность параметра $\tau_{v,F}{}^{g}$ относительно значения $\eta_{v,F}{}^{g}$;

$LV_a(F^g)$ – список возможных миграций ВМ, которые не нарушают ограничение емкости ресурсов целевой ФМ (выражение 1.13);

S – это ВМ, вероятность выбора которой осуществляется при использовании правила 3.12:

$$p_a(v,F_i^g)=\begin{cases}\dfrac{\tau_{v,F_i^g}\times\left[\eta_{v,F_i^g}\right]^{\beta}}{\sum\limits_{F_k^g\in F}\tau_{v,F_k^g}\times\left[\eta_{v,F_k^g}\right]^{\beta}} & \text{при } F_k^g\in F\\ 0 & \text{в противном случае}\end{cases} \qquad (3.12)$$

где F_k^g – количество доступных для размещения ФМ.

В выражении 3.11 при выборе пары (v,F_i^g) отдается предпочтение тем парам (v,F_i^g), которые улучшают использование ВКР целевой ФМ (как сбалансированное, так и общее) и снижают затраты на ЖМ, а также обладающие более высоким содержанием феромонов τ_{v,F_i^g}.

Данное правило s принятия решений о выборе ВМ и включении ее в список миграции работает следующим образом: когда $q\le q_0$, то выбирается пара (v,F_i^g), которая приводит к наибольшему значению $\tau_{v,F_i^g}\times\left[\eta_{v,F_i^g}\right]^{\beta}$ – желательности включения этой ВМ в список миграции и добавляется в матрицу миграции MM_{ij}.

В противном случае пара (v,F_i^g) выбирается с вероятностью $p_a(v,F_i^g)$ (выражение 3.12).

Эвристическая информация $\eta_{v,F}$ будет включать в себя параметры ВМ и ФМ, такие как количество доступных ресурсов, текущая загрузка и потребности ВМ. Учитывая эти параметры, ИМ оценивают возможность размещения и выбирают наиболее предпочтительные ФМ.

3. Определение глобального решения

В качестве глобального решения $\tilde{s}$ алгоритма муравьиной колонии будет рассматриваться матрица миграции MM_G с оптимальным переразмещением ВМ на ФМ, а ее элементами будут являться решения о переразмещении ВМ на ФМ.

Глобальное решение определяется по правилу 3.13:

$$\tilde{s} = \begin{cases} MM_G \text{ при } MM_G < MM_{ij} \\ MM_{ij} \text{ в противном случае} \end{cases} \quad (3.13)$$

В процессе разработки решения эвристические значения $\eta_{v,F}$ вычисляются динамически для каждой миграции ВМ на ФМ, которая определяет меру выгоды от решения по конкретной паре элементов множеств V и F (выражение 3.14):

$$\eta_{v,F_k^g} = \omega \times \left|\lg ВДР'\right| + (1-\omega) \times ВВР_{F_k^g}^v, \quad (3.14)$$

где ω – параметр важности полученного решения о сбалансированном использовании ресурсов R относительно их общего использования;

$ВДР'$ – величина вектора дисбаланса ресурсов $F_k^g \in F$ после присвоения ей ВМ;

$ВВР_{F_k}^v$ – векторное представление выделенного m-го ресурса вектора R для элемента $F_k^g \in F$, при назначении ему элемента $v \in V$ (выражение 3.15):

$$ВВР_{F_k^g}^v = \sum_{R_m \in R} (U_{F_k^g}^v + D_{F_k^g}^v) \quad (3.15)$$

Фактически η_{v,F^g} является обратной величиной δ_{v,F^g} – длины вектора ВДР:

то есть $\eta_{v,F^g}=\frac{1}{\delta_{v,F^g}}$ и представляет собой предпочтительный вариант ЖМ ВМ на целевую ФМ как с точки зрения использования ресурсов ФМ, так и с точки зрения накладных расходов на ЖМ ВМ.

4. Глобальное обновления феромонов

Также в отличие от классического алгоритма муравьиной колонии, где величина уровня феромона τ_{ij} определяет вес ребра на графе пути, в разработанном алгоритме переразмещения ВМ величина уровня феромона $\tau(t)$ на t-м шаге задается матрицей размерности $n\times m$, где n и m – соответственно мощности множеств V и F.

Величина обновления феромона $\Delta t_{vF^g,a}(t)$ для некоторого a-го ИМ изменяется в зависимости от качества решения с точки зрения целевой функции f и определяется как (выражение 3.16):

$$\Delta\tau_{vF^g,a}(t)=\begin{cases} f(MM_G) \text{ при } (v,F^g)\in MM_G \\ 0 \quad \text{в противном случае} \end{cases} \tag{3.16}$$

Чтобы облегчить ЖМ ВМ $(v)\in MM_G$ и чтобы ИМ могли лучше ориентироваться на следующих итерациях алгоритма, уровень феромонов $\tau_{vF^g,a}(t+1)$ каждой пары (v,F^g) обновляется с использованием правила, представленного в выражении 3.17.

$$\tau_{vF^g,a}(t+1)=(1-\delta)\times\tau_{vF^g,a}(t)+\delta\times\Delta\tau_{vF^g,a} \tag{3.17}$$

где $\Delta\tau_{vF^g,a}$ – усиление феромонами на каждой паре (v,F^g), составляющей решение глобальной матрицы миграции MM_G;

δ – глобальный параметр ослабления феромонов ($0 < \delta < 1$).

Значение феромона отражает «успешность» вариантов размещения ВМ на ФМ, увеличивая вероятность повторного использования удачных решений.

3.2 Обобщенное представление разрабатываемого алгоритма переразмещения виртуальных машин центра обработки данных

В данном разделе представлена обобщенная схема этапов алгоритма переразмещения ВМ ЦОД на основе метаэвристики муравьиной колонии и ее описание [116].

Используя предыдущий анализ, схема обобщенного алгоритма переразмещения ВМ (рисунок 3.1) может быть представлена последовательностью выполняемых этапов.

Этап 1. Инициализация параметров алгоритма

На первом этапе происходит здание входных данных и инициализация параметров алгоритма. В качестве входных данных используется кластер ФМ с размещенными ВМ, а схема переразмещения ВМ ЦОД использует модели выделения ресурсов на осуществление ЖМ ВМ, представленные в разделе 1.3.

Этап 2. Формирование гомогенных подмножеств ФМ

После задания входных данных и инициализации параметров алгоритма выполняется этап по формированию гомогенных подмножеств F^g на основе гетерогенного множества ФМ F.

Этап 3. Поиск локального решения

Следующим этапом является создание нескольких агентов – ИМ и предоставления каждому из них экземпляра входного кластера ФМ F^g с размещенными ВМ.

В процессе выполнения основной итерации алгоритма, созданные ИМ работают параллельно и каждый из них вычисляет в качестве решения алгоритма матрицу миграции ВМ – $MM_a = \langle v, F_i^g \rangle$, которая состоит из списка команд миграции ВМ на ФМ для всех ВМ в кластере.

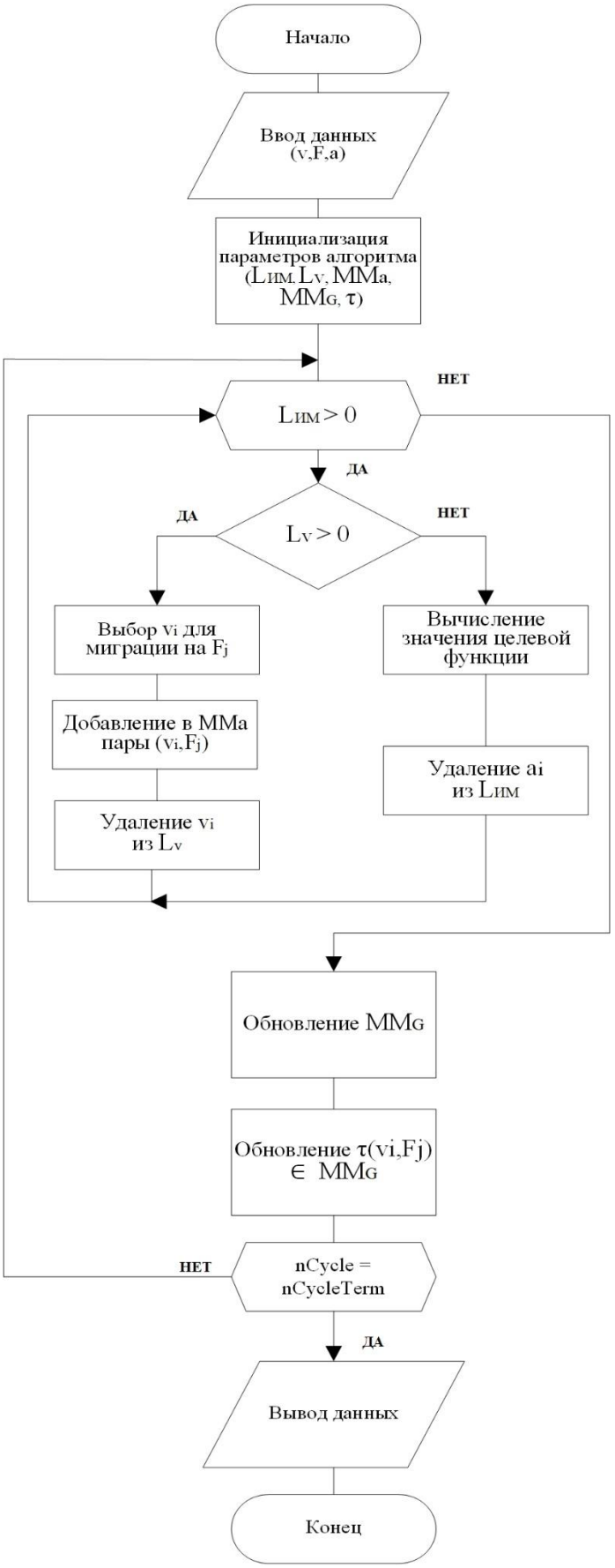


Рисунок 3.1 – Структурная схема алгоритма переразмещения виртуальных машин

В случае если исходная и целевая ФМ совпадают, все факторы миграции и накладные расходы для этих мигрируемых ВМ будут равны нулю и это не повлияет на общие затраты миграции.

Этап 4. Определение глобального решения

Затем алгоритм определяет наилучшую матрицу миграции MM_G на основе значения целевой функции f.

Этап 5. Глобальное обновления феромонов

На основе полученной наилучшей матрицы миграции MM_G алгоритм обновляет общую информацию о феромонах и запускает ИМ еще раз для следующего цикла.

Этап 6. Завершение работы алгоритма

Наконец, когда заданное условие завершения выполнено, алгоритм выводит найденную на данный момент наилучшую матрицу миграции MM_G.

На базе описанной обобщенной схемы интерпретированного алгоритма муравьиной колонии разработан подробный алгоритм переразмещения ВМ ЦОД, схема которого представлена в Приложении 2.

3.3 Алгоритм переразмещения виртуальных машин центра обработки данных

Применительно к ЦОД на основе проведенного анализа алгоритма муравьиной колонии определены следующие исходные данные для модифицированного алгоритма муравьиной колонии [116]:

- гетерогенное множество ФМ – F, представляющее собой количество серверов в кластере, использующие гипервизоры разных типов и которые необходимо консолидировать;

- множество ВМ – V, представляющее собой количество ВМ, размещенных на ФМ и которые необходимо перераспределить для оптимального размещения на консолидированных ФМ;

- множество ИМ – $A_{ИМ}$, представляющее количество муравьев-агентов, которые имитируют деятельность настоящих муравьев;

- показатель уровня феромонов – $\tau_{vF^g,a}$ для каждой пары ВМ-ФМ (v,F^g), на основе значения которого ИМ принимает решение о миграции ВМ.

Работу алгоритма переразмещения ВМ ЦОД можно описать последовательностью шагов его выполнения, при этом общую итерацию алгоритма можно разделить на два частных алгоритма:

1. **Алгоритм 1** – цикл, выполняющий итерацию по набору ИМ, и который входит в другой цикл, продолжающий свою работу пока список $L_{ИМ}$ не опустеет (рисунок 3.2). Включает в себя шаги 4–6.

2. **Алгоритм 2** – цикл формирования глобального решения $\tilde{s}$ (формирования глобальной матрицы миграции MM_G с оптимальным перераспределением ФМ в гомогенном подмножестве F^g) (рисунок 3.3). Включает в себя шаги 7–9.

Шаг 1. Ввод начальных данных

Процесс работы алгоритма начинается с установки начальных параметров, таких как:

- количество циклов завершения алгоритма – nCycleTerm;

- параметры алгоритма – δ, β, ω, q_0, q;

- множество ФМ (F), имеющих общую полноту ресурсов;

- множество ВМ (V), имеющих общие потребности в ресурсах;

- множество ИМ ($A_{ИМ}$) – общее количество ИМ.

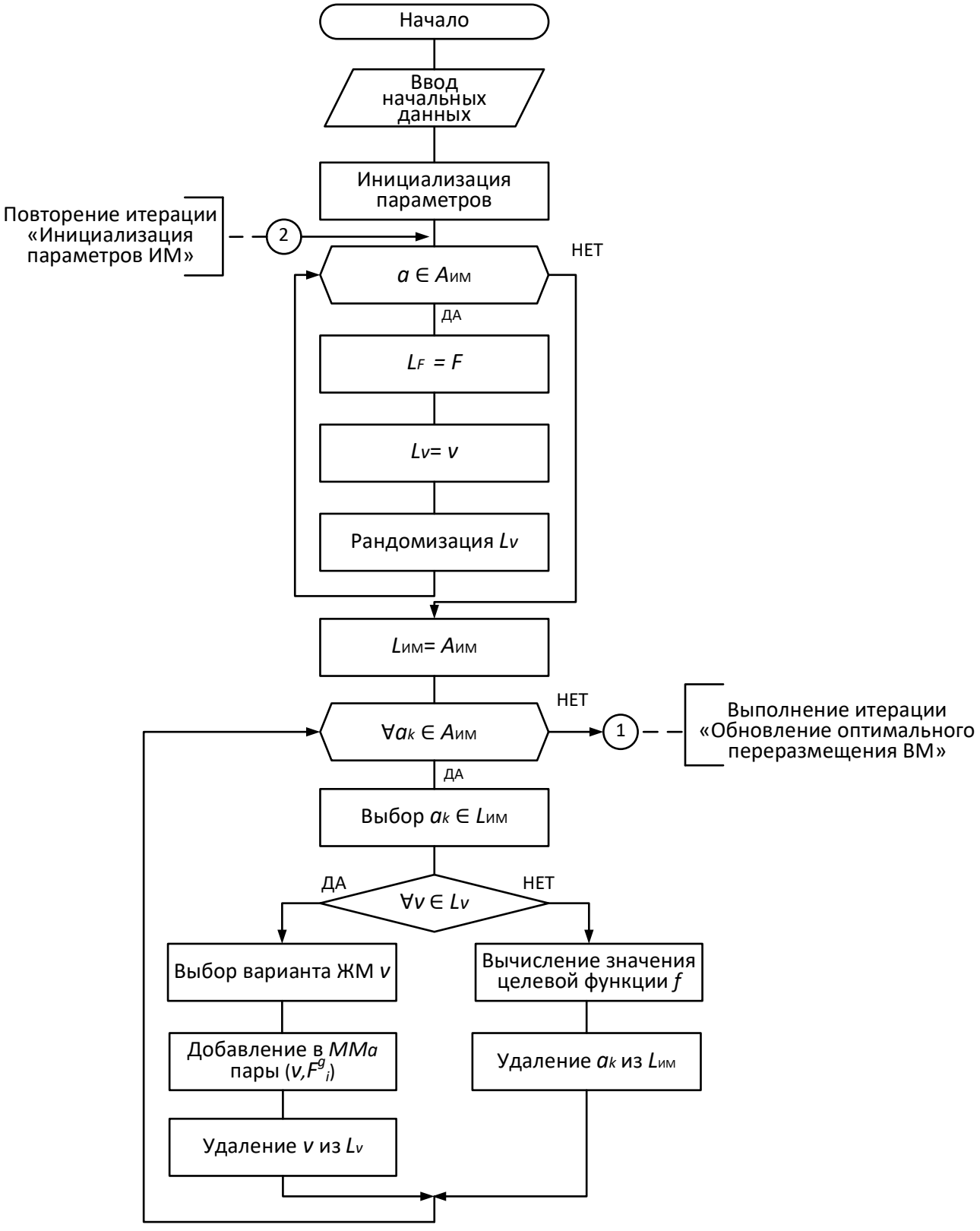


Рисунок 3.2 – Структурная схема алгоритма 1 - начальной инициализации множества ИМ и расчета локальной целевой функции s

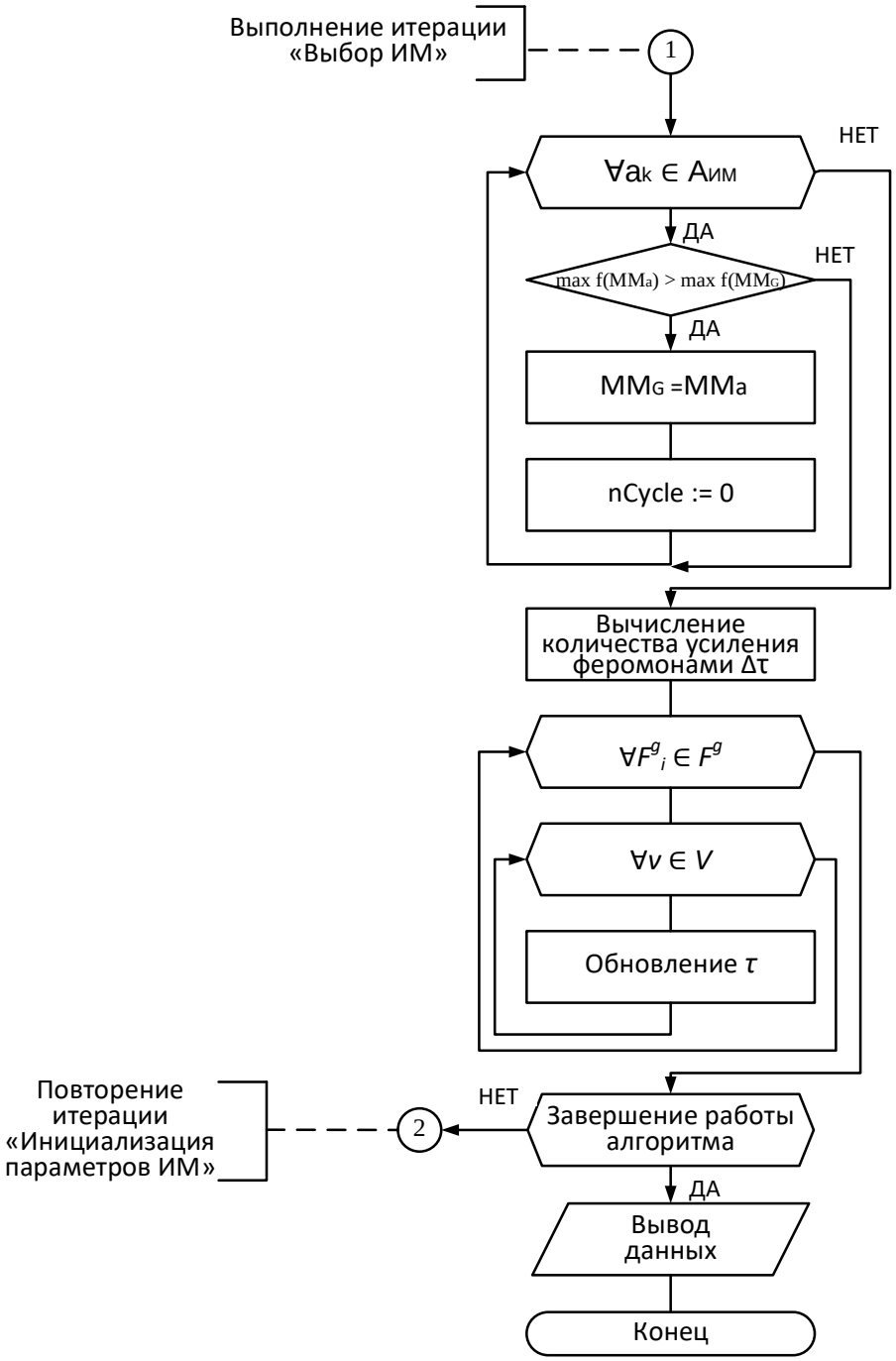


Рисунок 3.3 – Структурная схема алгоритма 2 - расчета глобальной целевой функции $\tilde{s}$

Шаг 2. Формирование гомогенных подмножеств ФМ

В рамках данного этапа происходит сортировка ФМ кластера с гипервизорами разного типа для последующего их объединения в подмножества ФМ с гипервизорами одного типа.

Схема алгоритма формирования гомогенных подмножеств F^g представлена на рисунке 3.4.

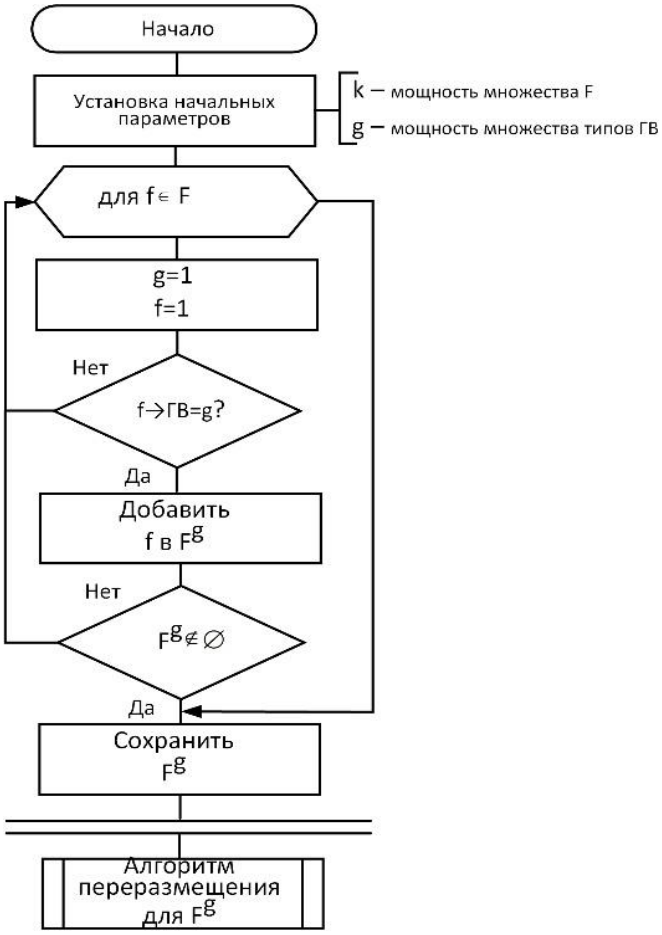


Рисунок 3.4 – Структурная схема алгоритма формирования гомогенных подмножеств F^g

Шаг 3. Инициализация параметров алгоритма

Следующим этапом является инициализация параметров алгоритма, таких как:

- список ФМ (LF_a^g) – в него входят выбранные ФМ, ресурсы которых выделяются распределяемым ВМ. Этот список назначается каждому ИМ, а количество ФМ выбирается случайным образом.

- список ВМ (L_V) – в него входят первоначально размещенные ВМ, ресурсы которых необходимо оптимально перераспределить на ФМ из списка LF_a. Очередность выбора ВМ происходит случайным образом.

- список ИМ ($L_{ИМ}$) – это «рабочий» список ИМ, в который входят ИМ из множества $A_{ИМ}$.

- показатель уровня феромона для каждой пары ВМ-ФМ (v,F^g) – $\tau_{vF^g,a}$.

В процессе инициализации параметров алгоритма устанавливается начальное значение уровня феромона $\tau_{vF^g,a} = \tau_0$, которое является фиксированным при выборе пары (v,F^g). На основе подхода, реализованного в классическом алгоритме муравьиной колонии [99,113], начальное значение величины уровня феромона τ_0 определяется как соотношение размещенных ВМ на активных ФМ до начала работы алгоритма:

$$\tau_0 = \frac{\sum ВМ}{\sum ФМ_A} \tag{3.18}$$

Матрица миграции с оптимальным переразмещением ФМ в кластере определяется как пустое множество ($MM_G = \varnothing$).

Шаг 4. Инициализация параметров ИМ

На этом этапе для каждого ИМ инициализируется матрица миграции MM_a, набор ФМ F^g, набор ВМ v, пустой список ФМ LF_a список ВМ L_V, которые еще не размещены на ФМ. После чего ВМ в списке L_V рандомизируются, чтобы внести случайность в процесс выбора ВМ для ИМ.

Шаг 5. Выбор ИМ

Алгоритм выполняет итерацию по набору a-го ИМ в «рабочий» список $L_{ИМ}$ из множества $A_{ИМ}$, для того чтобы проверить все ли ИМ были задействованы для

выполнения своих решений. И если в списке $L_{ИМ}$ остался хотя бы один не задействованный а-й ИМ, то он выбирается для выполнения последующих итераций. Если же список $L_{ИМ}$ пуст, то алгоритм переходит к выбору оптимального переразмещения ВМ (**шаг 7**). Выбор а-го ИМ из списка $L_{ИМ}$ происходит случайным образом.

Шаг 6. Выбор ВМ для ее миграции на заполняемую ФМ

Если у выбранного ИМ в списке L_V присутствуют ВМ, то он выбирает «ближайшую» ВМ из этого списка L_V на основе значений феромонов для миграции на текущую ФМ (выражение 3.5).

$$s = \begin{cases} \operatorname{argmax}_{v \in LV_a(F^g)} \left\{ \tau_{v,F^g} \times \left[\eta_{v,F^g} \right]^{\beta} \right\} & \text{при } q \le q_0 \\ S & \text{в противном случае} \end{cases} \qquad (3.19)$$

Затем ИМ добавляет выбранную пару (v,F^g) в свою матрицу миграции MM_a и удаляет выбранную ВМ из своего списка L_V.

При этом, если в списке L_V а-го ИМ есть хотя бы одна ВМ, но при этом запрашиваемые ресурсы для этой ВМ превышают полноту имеющихся ресурсов текущей ФМ ИМ, то для заполнения выбирается новая пустая ФМ из списка LF_a ИМ.

Если же список L_V выбранного ИМ пуст, то ИМ завершает принятие решений о миграции ВМ и вычисляет значение **целевой функции f** для последующего формирования своей матрицы миграции MM_a в соответствии с выражением 3.20, и ИМ удаляется из списка $L_{ИМ}$.

$$\max f(MM_G) = \frac{N_{F_{HA}}}{HP_{MM_G}}, \qquad (3.20)$$

Лучшая матрица миграции MM_G определяется по значению коэффициента использования ФМ в ней – K_{MM_G} :

$$K_{MM_G} = \frac{N_{F_{HA}}}{N_F}, \quad (3.21)$$

Физический смысл выражения 3.21 можно интерпретировать следующим образом: чем больше получается высвободить ФМ F_{HA} в результате переразмещения ВМ, тем выше значение коэффициента использования ФМ, и соответственно более оптимальна матрица миграции MM_G.

Алгоритм продолжает выполнять данную итерацию, пока список $L_{ИМ}$ не опустеет.

При назначении ВМ на ФМ а-й ИМ учитывает размещение и соответствующие издержки миграции этой ВМ, определяемые как накладные расходы на процесс ЖМ ВМ [78].

Шаг 7. Обновление оптимального переразмещения ВМ

На этом этапе алгоритм обновляет матрицу миграции MM_G с оптимальным переразмещением ВМ в кластере, только после того, когда все ИМ завершат построение своих матриц миграции MM_a и будет определена лучшая MM_G. Для этого алгоритм проверяет имеет ли какая-либо MM_a лучшее значение целевой функции f, чем текущая MM_G :

$$\max f(MM_a) > \max f(MM_G) \quad (3.22)$$

Если это так, то матрица миграции MM_G обновляется, счетчик текущей итерации алгоритма nCycle сбрасывается и процесс поиска повторяется.

Если не обнаруживается улучшение решения, то алгоритм завершает работу с текущим значением матрицы миграции MM_G в качестве выходных данных.

Шаг 8. Обновление феромонов

Количество усиления феромонами $\Delta\tau_{vF^g,a}$ зависит от оптимальности полученной матрицы миграции MM_G и определяется как:

$$\Delta\tau_{vF^g,a}(t)=\begin{cases} f(MM_G) \text{ при } (v,F^g)\in MM_G \\ 0 \quad \text{в противном случае} \end{cases} \tag{3.23}$$

Из выражения 3.23 видно, что алгоритм усиливает значение феромона только в парах (v,F^g), которые принадлежат глобальной матрице миграции MM_G.

Правило обновления уровня феромонов $\tau_{vF^g,a}$ каждой пары (v,F^g) на шаге $(t+1)$ определяется как:

$$\tau_{vF^g,a}(t+1)=(1-\delta)\times\tau_{vF^g,a}(t)+\delta\times\Delta\tau_{vF^g,a} \tag{3.24}$$

Шаг 9. Проверка условия завершения работы алгоритма

Основной процесс продолжает итерации до тех пор, пока не будет достигнуто максимальное количество циклов алгоритма nCycleTerm.

Результатом работы алгоритма является матрица миграции MM_G с квазиоптимальным размещением ВМ по ФМ, которая позволяет значительно увеличить число размещенных ВМ и повысить показатель утилизации ВКР ФМ. Также это способствует уменьшению количества процессов ЖМ ВМ и снижению связанных с этим накладных расходов, что в итоге позволяет сократить количество задействованных ФМ.

3.4 Оценка сложности алгоритма переразмещения виртуальных машин виртуализированного центра обработки данных

Для оценки сложности алгоритма переразмещения ВМ будет применяться метод «Big-O Notation» [119,120], который позволяет проводить анализ изменения работы алгоритма во времени и алгоритмическом пространстве с увеличением объема входных данных.

Так как с ростом количества входных данных повышается число операций и, следовательно, время выполнения алгоритма и объем памяти, необходимой для обработки данных алгоритмом.

Такой метод позволяет оценить ресурсы, требуемые для работы алгоритма при изменении объема данных.

Предложенный модифицированный алгоритм перемещения ВМ разработан на основе описаний из разделов 3.1-3.3 и выполняется в соответствии с последовательностью, представленной в Приложении 2. ВМ в ВЦОД организованы по кластерам ФМ, и переразмещение ВМ выполняется по модифицированному алгоритму отдельно в каждом кластере. Показатели эффективности ВЦОД (ресурсы, энергопотребление и накладные расходы на ЖМ) собираются из всех кластеров согласно заранее установленным формулам.

3.4.1 Временная сложность алгоритма переразмещения виртуальных машин

Оценка временной сложности алгоритма переразмещения ВМ будет производиться на основе оценки каждого шага этого алгоритма.

Шаг 1. Ввод начальных данных.

Временная сложность зависит от размера входных данных, таких как: количество итераций (nCycleTerm), количества ИМ ($A_{ИМ}$), количества ВМ и ФМ в кластере (|V| и |F|) и времени выполнения данной итерации t_1 .

Временная сложность определяется как:

$$T_1 = op_1(nCycleTerm; A_{ИМ}; V; F) * t_1 = O_1(1) \quad (3.25)$$

где $op_1(nCycleTerm; A_{ИМ}; V; F) = const$, $t_1 = const$.

Шаг 2. Формирование гомогенных подмножеств ФМ.

Временная сложность зависит от временной сложности этапов алгоритма формирования гомогенных подмножеств F^g (время выполнения каждого их этих

этапов постоянно) и времени выполнения итерации формирования гомогенных подмножеств F^g – t_2.

Временная сложность определяется как:

$$T_2=(op_2^1(k,g)*t_2^1+op_2^2(F)*t_2^2+op_2^3(\left|F^g\right|)*t_2^3)*t_2=O_2(n) \quad (3.26)$$

где k и g – входные параметры;

F –физические машины с ГВ разного типа;

F^g – физические машины с ГВ одного типа;

t_2^1, t_2^2, t_2^3 – время выполнения каждого этапа алгоритма формирования гомогенных подмножеств F^g;

$t_2^1=const, t_2^2=const, t_2^3=const, t_2=const$.

Шаг 3. Инициализация параметров алгоритма.

Временная сложность этого этапа равна:

$$T_3=op_3(\left|v,F^g\right|)*t_3=O_3(n) \quad (3.27)$$

где n – количество пар (v,F^g), по причине того, что алгоритм просто перебирает каждую пару (v,F^g) и устанавливает значение феромона τ_0 равным 0, что требует постоянного количества времени t_3 на выполнение итерации для каждой пары (v,F^g).

Шаг 4. Инициализация параметров ИМ.

Временная сложность этапа инициализации структур данных, связанных с ИМ равна:

$$T_4=op_4(\left|A_{ИМ}\right|)*t_4=O_4(n) \quad (3.28)$$

где n – количество ИМ в $A_{ИМ}$. Это определяется тем, что алгоритм выполняет итерации и постоянный объем работы для каждого ИМ в $A_{ИМ}$ за время $t_4=const$.

Шаг 5. Выбор ИМ.

Временная сложность этого этапа равна:

$$T_5=op_5(|L_{ИМ}|)*t_5=O_5(n) \quad (3.29)$$

где n – количество ИМ в списке $L_{ИМ}$, потому, что алгоритм выполняет итерацию по набору ИМ в список $L_{ИМ}$, выбирая ИМ случайным образом и выполняя операции над ним, пока $L_{ИМ}$ не опустеет.

Шаг 6. Выбор ВМ для ее миграции на заполняемую ФМ.

Временная сложность данного этапа равна:

$$T_6=op_6(|L_V|)*t_6=O_6(n) \quad (3.30)$$

где n – количество ВМ в L_V. Это связано с тем, что алгоритм выполняет итерацию по каждой ВМ в списке один раз, чтобы выбрать пару (v,F^g) и добавить ее в матрицу миграции MM_a, выполняя операции с постоянным временем t_6 для каждой ВМ.

Шаг 7. Обновление оптимального переразмещения ВМ.

Временная сложность этого этапа равна:

$$T_7=op_7(|A_{ИМ}|)*t_7=O_7(n) \quad (3.31)$$

где n – количество ИМ в $A_{ИМ}$, так как алгоритм выполняет итерацию по каждому ИМ в $A_{ИМ}$ ровно один раз за время $t_7=const$.

Шаг 8. Обновление феромонов.

Временная сложность этапа обновления феромонов определяется как:

$$T_8 = op_8(\left|F^g\right| * |V|) * t_8 = O_8(n*m) \tag{3.32}$$

где n – количество ФМ F_a^g в наборе F^g, а *m* – количество ВМ в наборе V.

Это связано с тем, что существуют два вложенных цикла, которые перебирают все элементы в наборах F^g и V, выполняя операции с постоянным временем t_8 внутри каждой итерации.

Шаг 9. Проверка условия завершения работы алгоритма.

Временная сложность этапа проверки условия завершения работы общей итерации алгоритма зависит от временной сложности этапов общей итерации алгоритма переразмещения ВМ и количества выполнения циклов итерации $nCycleTerm = const$.

Временная сложность данного этапа определяется как :

$$\begin{aligned} &T_9 \le (T_1+T_2+T_3+T_4+T_5+T_6+T_7+T_8) * nCycleTerm \Rightarrow \\ &\Rightarrow T_9 \le ((O_1(1)+O_2(n)+O_3(n)+O_4(n)+O_5(n)+O_6(n)+ \\ &+O_7(n)+O_3(n+m))+nCycleTerm) \Rightarrow \\ &\Rightarrow T_9 \le O_9(n+m+nCycleTerm) \end{aligned} \tag{3.33}$$

Из этого следует, что временная сложность алгоритма переразмещения ВМ в ЦОД в наихудшем случае может быть определена как:

$$T_{АЛГ} = T_9 \le O_9(n+m+nCycleTerm) \tag{3.34}$$

Из выражения (3.34), что временная сложность алгоритма переразмещения ВМ в ЦОД существенно зависит от числа ВМ и ФМ в ЦОД. При увеличении количества ВМ возрастает размер наилучшей матрицы миграции MM_G, что приводит к росту вычислительной нагрузки. На каждом шаге алгоритма необходимо оценивать текущее состояние ресурсов ФМ, что в свою очередь требует дополнительных затрат времени.

3.4.2 Пространственная сложность алгоритма переразмещения виртуальных машин

Пространственная сложность алгоритма измеряет объем памяти, которую он использует в зависимости от размера входных данных.

Оценка пространственной сложности алгоритма переразмещения ВМ будет производиться на основе оценки каждого шага этого алгоритма.

Шаг 1. Ввод начальных данных.

Пространственная сложность этого этапа равна:

$$M_1=O_1(|A_{ИМ}|;|V|;|F|) \quad (3.35)$$

поскольку она зависит от мощности множеств ИМ $A_{ИМ}$, ВМ V и ФМ F.

Шаг 2. Формирование гомогенных подмножеств ФМ.

Пространственная сложность алгоритма формирования гомогенных подмножеств ФМ зависит от пространственной сложности этапов этого алгоритма F^g, а конкретно – от мощности множеств ФМ F, типов ГВ g и ФМ с ГВ одного типа F^g. Пространственная сложность определяется как:

$$M_2=(O_2^1(|F|)+O_2^2(|g|)+O_2^3(|F^g|)) \quad (3.36)$$

Шаг 3. Инициализация параметров алгоритма.

Так как алгоритму требуется место для хранения только n пар (v,F^g), и при этом каждой такой паре (v,F^g) требуется постоянный объем пространства для хранения значения феромона t_{v,F^g}, то общая сложность пространства этапа инициализации параметров алгоритма линейна по количеству пар и равна:

$$M_3=O_3(|v,F^g|) \quad (3.37)$$

Шаг 4. Инициализация параметров ИМ.

Алгоритм инициализирует структуры данных для каждого ИМ в $A_{ИМ}$, что приводит к линейному увеличению использования памяти с увеличением количества ИМ. Их чего следует, что пространственная сложность этапа инициализации параметров ИМ равна

$$M_4=O_4(|A_{ИМ}|) \quad (3.38)$$

где n – количество ИМ в $A_{ИМ}$.

Шаг 5. Выбор ИМ.

Пространственная сложность этого этапа алгоритма равна:

$$M_5=O_5(|L_{ИМ}|) \quad (3.39)$$

где n – количество ИМ в списке $L_{ИМ}$.

Это связано с тем, что алгоритму требуется место только для списка $L_{ИМ}$ и отдельных ИМ в списке. Здесь нет дополнительных структур данных или переменных, которые значительно увеличивают пространственную сложность. В целом пространственная сложность этапа выбора ИМ линейна по отношению к количеству ИМ в списке $L_{ИМ}$.

Шаг 6. Выбор ВМ для ее миграции на заполняемую ФМ.

Пространственная сложность данного этапа равна:

$$M_6=O_6(|MM_a|;|v,F^g|) \quad (3.40)$$

потому что алгоритму требуется только постоянный объем дополнительного пространства для таких переменных, как пары (v,F^g) и матрица миграции ИМ MM_a. Размер входных данных (LF_a^g и L_V) не влияет пространственную сложность.

Шаг 7. Обновление оптимального переразмещения ВМ.

Пространственная сложность этого этапа алгоритма равна:

$$M_7=O_7(1) \quad (3.41)$$

потому что алгоритм использует здесь только постоянный объем пространства независимо от размера входных данных.

Шаг 8. Обновление феромонов.

Пространственная сложность этапа обновления феромонов равна:

$$M_8=O_8(1) \quad (3.42)$$

потому что алгоритм использует только постоянное количество дополнительного пространства независимо от мощности множеств F^g и V.

Шаг 9. Проверка условия завершения работы алгоритма.

Поскольку алгоритм требует хранения ВКР всех ФМ и ВМ, то пространственная сложность алгоритма определяется как:

$$M_{АЛГ}=O_{АЛГ}(M_F*N_F+M_V*N_V) \quad (3.43)$$

где M_{F^g} – это общая память, выделяемая на все ФМ F;

N_{F^g} – количество ФМ F;

M_V – общая память выделяемая на все ВМ V;

N_V –количество ВМ V.

Из выражения (3.43), что пространственная сложность алгоритма переразмещения ВМ в ЦОД существенно зависит от числа ВМ и ФМ в ЦОД. При увеличении количества ВМ возрастает размер наилучшей матрицы миграции MM_G, что приводит к росту объема памяти для хранения значений феромонов для каждой пары (v,F^g).

3.5 Выводы по главе

В главе представлен разработанный алгоритм переразмещения ВМ, основанный на метаэвристике муравьиной колонии, обеспечивающий получение матрицы миграции ВМ, отличающийся от известных алгоритмов учетом при ее расчете гетерогенной структуры ЦОД и дополнительных ресурсов, представляющих собой накладные расходы, необходимые алгоритмам ЖМ ВМ.

Алгоритм переразмещения ВМ представлен как трехпроцедурный алгоритм, рассчитывающий параллельно матриц миграции ВМ для гомогенных подмножеств физических машин ЦОД.

ГЛАВА 4. РАЗРАБОТКА АРХИТЕКТУРЫ ПРОГРАММНО-РЕАЛИЗОВАННОЙ РАСПРЕДЕЛЕННОЙ СИСТЕМЫ ПОДДЕРЖКИ ПЕРЕРАЗМЕЩЕНИЯ ВИРТУАЛЬНЫХ МАШИН ВИРТУАЛИЗИРОВАННОГО ЦЕНТРА ОБРАБОТКИ ДАННЫХ

4.1 Архитектура программного комплекса поддержки процесса переразмещения виртуальных машин виртуализированного центра обработки данных

В [120] под распределенной системой понимается группа автономных компьютерных систем, которые физически разделены, но подключены к единой компьютерной сети и управляются программным обеспечением распределенной системы. Эти независимые компьютеры взаимодействуют между собой, делятся ресурсами и файлами, а также выполняют назначенные им задачи.

ВЦОД предлагается рассматривать как распределенную систему, поскольку он соответствует следующим ее характеристикам:

1) **Совместное использование ресурсов**: возможность задействовать любые аппаратные средства, программное обеспечение или данные в любом месте системы.

2) **Открытость**: это касается того, насколько легко система может быть дополнена и улучшена (т.е. насколько открыто разрабатывается и передается программное обеспечение).

3) **Параллелизм:** естественным образом существует в распределенных системах, где аналогичные задачи могут выполняться разными пользователями, расположенными в различных местах. Каждая локальная система включает свои собственные операционные системы и ресурсы.

4) **Масштабируемость:** позволяет увеличивать размер системы, так как несколько процессоров взаимодействуют с большим количеством пользователей, увеличивая скорость реакции системы.

5) **Отказоустойчивость:** обеспечивает надежное функционирование системы даже в случае сбоев в аппаратном обеспечении или программном обеспечении, не снижая общей производительности.

6) **Прозрачность:** скрывает сложность распределенных систем от пользователей и приложений, обеспечивая конфиденциальность в каждой системе.

7) **Гетерогенность:** различные компоненты распределенной системы могут отличаться по типу сети, компьютерному оборудованию, операционным системам, языкам программирования и реализации.

ВЦОД учитывает эти аспекты, предлагая решения, которые обеспечивают высокую степень гибкости и эффективности. Во-первых, совместное использование ресурсов позволяет компаниям оптимизировать использование вычислительной мощности, распределяя рабочую нагрузку между различными серверами и узлами. Это уменьшает издержки на оборудование и повышает производительность, используя все доступные ресурсы на максимум. В таких системах можно динамически добавлять или удалять ресурсы в зависимости от текущих потребностей, что делает их крайне эффективными и устойчивыми к изменениям.

Открытость ВЦОД играет ключевую роль в их внедрении и развитии. Благодаря поддержке открытых стандартов, эти системы легко интегрируются с различными платформами и технологиями. Возможность модификации и улучшения программного обеспечения позволяет операторам ЦОД внедрять новейшие технологии и поддерживать высокие стандарты безопасности и производительности. Более того, пользователи и разработчики могут вносить свои предложения по улучшению системы, что способствует созданию экосистемы инноваций.

Параллелизм и отказоустойчивость включают в себя важные механизмы, которые обеспечивают постоянную доступность услуг и ресурсов. ВЦОД могут обрабатывать множество параллельных задач от различных пользователей, позволяя выполнять комплексные вычислительные процессы значительно быстрее.

Это достигается благодаря распределению нагрузки между различными серверами и использованием резервных входов.

Отказоустойчивость достигается за счет создания резервных копий данных и использования дублирующих систем, которые автоматически включаются в работу при обнаружении сбоев. Это обеспечивает высокую степень надежности и минимизирует вероятность простоев.

Прозрачность и гетерогенность ВЦОД дают пользователям ощущение работы с единым целым, несмотря на сложную внутреннюю структуру системы. Прозрачность скрывает все сложности распределенных компонентов, предоставляя пользователям простой и интуитивно понятный интерфейс доступа к ресурсам и данным.

В то же время, поддержка гетерогенных компонентов позволяет интегрировать различные устройства и программы в единую систему, что значительно расширяет возможности и адаптивность таких ЦОД. Это позволяет компаниям гибко реагировать на изменяющиеся потребности бизнеса и быстро внедрять новые технологии.

Система поддержки переразмещения ВМ в ВЦОД также относится к распределенным системам в связи с необходимостью учета локальной и/или территориальной распределенности серверных платформ ВЦОД, с которых собираются/поступают параметры ВМ и ФМ ВЦОД.

В связи с этим при разработке архитектуры системы рекомендуется использовать парадигму «менеджер-агенты». Для этой цели предлагается использовать специализированную двухуровневую архитектуру, изображённую на рисунке 4.1.

Уровнями архитектуры разработанной распределенной системы поддержки переразмещения ВМ на основе парадигмы «менеджер-агент» являются:

1. Уровень агентов – программных модулей, взаимодействующих с ГВ ВЦОД и вышележащим уровнем. Функциями модуля-агента являются: получение данных от ГВ о его типе, значениях наличия и остатка ВКР, количестве активных ВМ, потребности активных ВМ в ВКР.

2. Уровень менеджмента – программных модулей:

- формирования гомогенных подмножеств ФМ по типу используемых ГВ (в п. 1.1.1 на рисунке 1.2 подмножества g1 и g2);

- формирования значений векторов ВНР, ВОР, ВВР для подмножеств g1 и g2;

- выполнения расчета значений множества векторов ВДР для подмножеств g1 и g2;

- запуска копий модуля расчета матриц миграции MMG для подмножеств g1 и g2;

- формирования на основе рассчитанных матриц миграции планов миграции подмножеств g1 и g2 и передачи их уровню агентов.

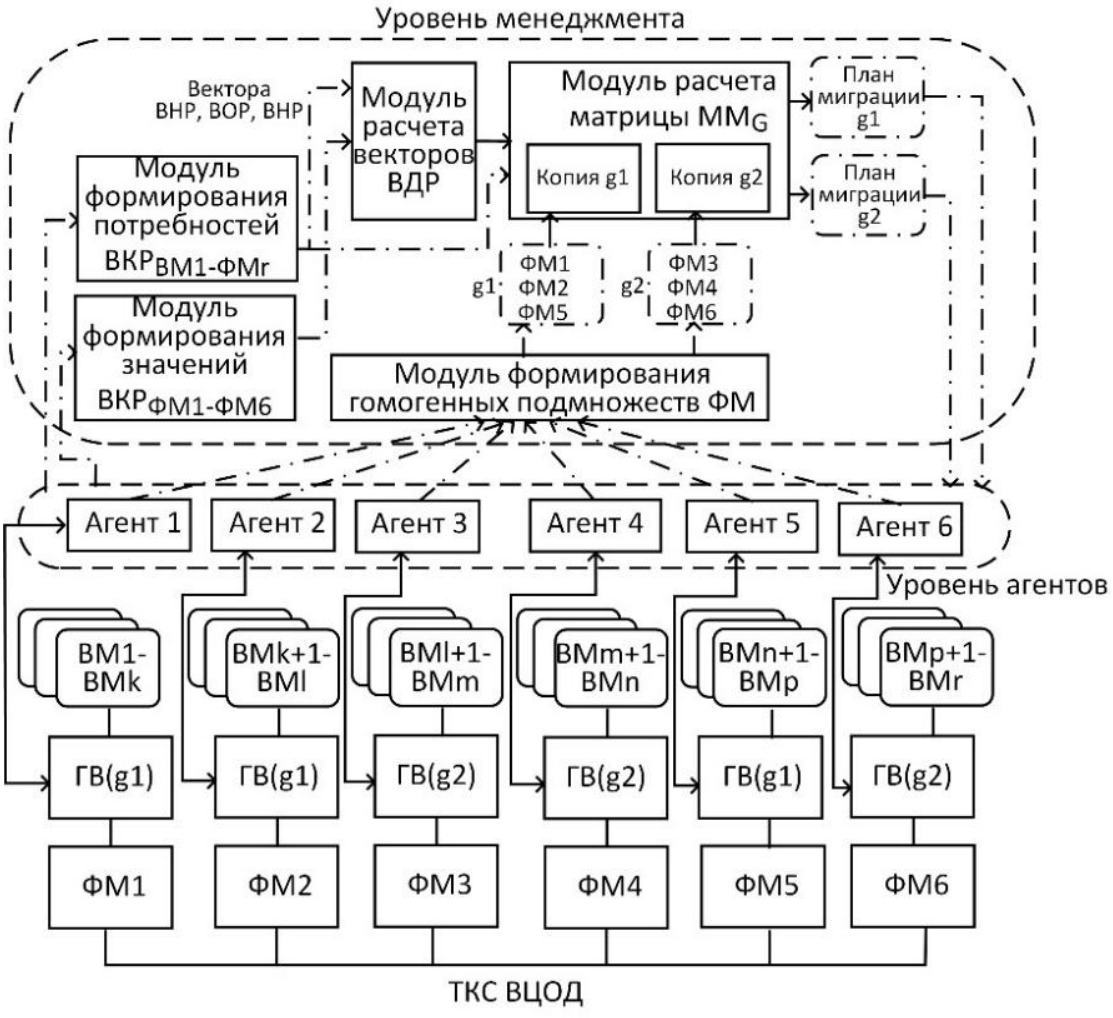


Рисунок 4.1 – Архитектура программного комплекса поддержки процесса переразмещения ВМ на основе парадигмы «менеджер-агент»

Парадигма «менеджер-агент» позволяет эффективно распределять задачи по сбору и обработке данных между различными узлами системы, что, в свою очередь, повышает гибкость и масштабируемость всей архитектуры.

В рамках данной парадигмы центральный менеджер отвечает за координацию работы агентов, которые размещаются на различных серверных платформах. Каждый агент собирает параметры ВМ и ФМ в своем локальном узле и передает эти данные менеджеру. Менеджер, в свою очередь, агрегирует и анализирует полученную информацию, обеспечивая целостное представление о состоянии всей системы. Это позволяет оперативно обнаруживать потенциальные проблемы и принимать необходимые меры для их устранения.

Использование агентов также способствует повышению устойчивости системы. В случае выхода из строя одного из агентов, менеджер перенаправляет его задачи другим агентам, обеспечивая непрерывность сбора и анализа данных. Таким образом, парадигма «менеджер-агенты» обеспечивает не только высокую производительность и точность сбора информации, но и высокую отказоустойчивость системы в целом.

Кроме того, данная архитектура позволяет легко масштабировать систему в зависимости от растущих потребностей организации. Добавление новых серверов и агентов не требует значительных изменений в основной логике работы менеджера, что делает процесс расширения системы быстрым и малозатратным.

Таким образом, использование парадигмы «менеджер-агенты» является эффективным и перспективным решением для построения распределенной системы поддержки ВЦОД.

Важно отметить, что агенты, выполняющие функции сбора данных на местах, включая мониторинг состояния оборудования, параметров производительности и использования ресурсов, были разработаны с учетом возможности их установки на различные операционные системы и аппаратные платформы, благодаря чему поддерживается высокая степень кроссплатформенности сервисного ПО ВЦОД. Это оказалось возможным за счет

использования универсальных интерфейсов прикладного программирования (API), что позволяет интегрировать агентов с существующими системами мониторинга.

Совместимость с такими системами, как Zabbix, NetCrunch, OpenNMS и другими, была достигнута благодаря открытым интерфейсам и протоколам, что значительно упрощает процесс интеграции и снизило затраты на внедрение системы поддержки [122]. Используя эти интерфейсы, менеджер получает доступ к данным, которые уже собираются и обрабатываются существующими системами мониторинга, что позволяет избежать дублирования функций и значительно увеличивает операционную эффективность.

Представленные программные модули архитектуры распределенной системы поддержки переразмещения ВМ предлагается реализовать в фреймворке PySyncObj, предназначенном для разработки распределенных систем и имеющим соответствующие API к существующим системам мониторинга [123,124]. Важным аспектом функционирования этих модулей является их способность взаимодействовать не только с данными, предоставляемыми ГВ ВЦОД, но и с внешними системами для обеспечения всестороннего анализа и принятия решений на основе актуальных данных.

На уровне агентов программные модули осуществляют мониторинг и сбор данных в режиме реального времени, что позволяет своевременно реагировать на изменения в активности ВМ и потребления ВКР. Эти модули работают автономно, но в тесной интеграции с уровнем менеджмента, что гарантирует непрерывность и надежность функционирования всей системы.

Уровень менеджмента, в свою очередь, отвечает за стратегическое планирование миграции ВМ. Благодаря использованию векторов ВНР, ВОР и ВВР для подмножеств g1 и g2, система может легко адаптироваться к любым изменениям в конфигурации ВКР. Результаты расчетов матриц миграции MM_G служат основой для принятия решений, которые затем передаются на уровень агентов для непосредственного выполнения. Эта структура обеспечивает высокую гибкость и эффективность переразмещения ВМ, оптимизируя использование ресурсов и минимизируя простои.

4.2 Проектирование структуры имитационной модели гетерогенного виртуализированного центра обработки данных

4.2.1 Обоснование выбора среды имитационного моделирования

Чтобы оценить работоспособность и эффективность разработанного в третьей главе алгоритма, необходимо решить задачу синтеза такой сложной системы используя средства компьютерного моделирования. Из всех доступных методов наилучшим выбором является применение инструментов имитационного моделирования, так как это наиболее распространённый подход для решения таких задач [125,126].

Главная цель моделирования заключается в получении как количественных, так и качественных результатов на основе имитационной модели. Под качественными характеристиками подразумеваются обнаруженные свойства моделируемой системы, а под количественными – значения переменных и их характеристики.

Создание компьютерной модели представляет собой сложную задачу, поскольку необходимо учитывать все внутренние связи объекта моделирования, и модель должна точно отражать все его факторы. Поэтому требуется проанализировать существующие инструменты имитационного моделирования на рынке, которые способны решить поставленные исследовательские задачи.

В настоящее время на рынке программного обеспечения для имитационного моделирования представлено множество различных продуктов, разработчики которых предлагают широкий спектр систем для различных методов имитационного моделирования. (Таблица 4.1) [126].

Таблица 4.1 Программное обеспечение для имитационного моделирования

Методы имитационного моделирования	Программные инструменты имитационного моделирования
Дискретно-событийное моделирование	GPSS, AnyLogic, Arena, Extend, PowerSim Studio, ProModel, Pilgrim, SimScript, FlexSim и др.

Системная динамика	AnyLogic, Arena, SimBioSys, eMPlant, Plant Simulation, SimuLab, VenSim, Pilgrim, Dynamo, Stella и др.
Агентное моделирование	AnyLogic, SimAgent, SimBioSys, AgentSpeak, TeleScript, RePast, NetLogo, Mason и др.

Так как в качестве структуры архитектуры программного комплекса поддержки процесса переразмещения ВМ выбрана парадигма «менеджер-агент», то выбор среды моделирования был сделан в пользу программного продукта российского разработчика – AnyLogic, разработанного российской компанией «The AnyLogic Company». Эта платформа позволяет моделировать системы различных типов, включая дискретную событийную, системную динамику и агентное моделирование, что делает её универсальным инструментом для широкого спектра задач [128,129].

Эта среда имитационного моделирования обладает рядом достоинств: необходимые для решения исследовательских задач встроенные библиотеки, возможность программирования на Java для создания нестандартных свойств объектов и классов, удобный и интуитивно понятный графический интерфейс, множество примеров моделей и исчерпывающая документация. Платформа также предлагает гибкость и мощные аналитические инструменты.

В состав структуры модели в среде имитационного моделирования AnyLogic включены следующие объекты основной библиотеки:

- «Source» – формирует задачи. Задачи создаются с учетом определенной интенсивности. Источником заявок может стать любой из вычислительных узлов.
- «Delay» – откладывает заявки на заданное время. Временной интервал задержки определяется динамически исходя из параметров задачи.
- «Sink» – удаляет задачи. Задачи, попавшие в систему, должны быть уничтожены.

- «ResourcePool» – определяет набор ресурсов, которые могут быть захвачены и освобождены при выполнении задач. В качестве ресурсов указываются процент использования процессора и объем оперативной памяти.
- «DataSet» – класс на языке Java, предназначенный для хранения данных типа "double" и отслеживания минимальных и максимальных значений данных для каждого измерения.
- «Event» – класс на языке Java, используемый для планирования действий в модели. События активируются при выполнении определённых условий, таких как наличие достаточных вычислительных ресурсов.
- «Connections» – хранит связи с контактами текущего агента и задает параметры их взаимодействия.

4.2.2 Формирование структуры имитационной модели гетерогенного виртуализированного центра обработки данных

Структура имитационной модели гетерогенного ВЦОД представлена следующими модулями (генераторами структурных элементов ЦОД):

1) Модуль генерации параметров ЦОД.
2) Модуль генерации параметров кластера.
3) Модуль генерации параметров ВМ.
4) Модуль генерации параметров ЖМ.

В модуле генерации «Параметры ЦОД» генерируется количество кластеров ФМ, представленных как кластеры серверов так как ЖМ ВМ осуществляется в пределах одного кластера.

В модуле генерации «Параметры кластера» ФМ задается количество серверов (ФМ) и пропускная способность кластера, при этом в каждом сервере генерируются ресурсы: C (CPU), M (RAM).

Тип ГВ (g1, g2) представлен типом алгоритма ЖМ:

- g1 – это Pre-copy;
- g2 – это Post-copy.

В модуле генерации «Параметры ВМ» задается количество ВМ с запрашиваемыми/потребляемыми ресурсами: C (CPU), M (RAM). Для каждой ВМ генерируются разные по объему ресурсы.

В модуле генерации «Параметры ЖМ» реализованы два алгоритма ЖМ ВМ Pre-copy и Post-copy, на основе результатов анализа в п. 1.3, и три алгоритма перераспределения ВКР ЦОД:

1) на основе метода XEN SandPiper – XEN SandPiper;

2) на основе метода вычисления нагрузки виртуализированного сервера – МНВС;

3) разработанный алгоритм переразмещения ВМ ЦОД на основе метаэвристики ACS – МП ВМ ЦОД.

Принципы перераспределения ВКР, реализованные в алгоритмах XEN SandPiper и МНВС рассмотрены в п. 2.3.3 и п. 2.3.4.

Вид интерфейса настройки структуры ВЦОД в разработанной имитационной модели представлен на рисунке 4.2.

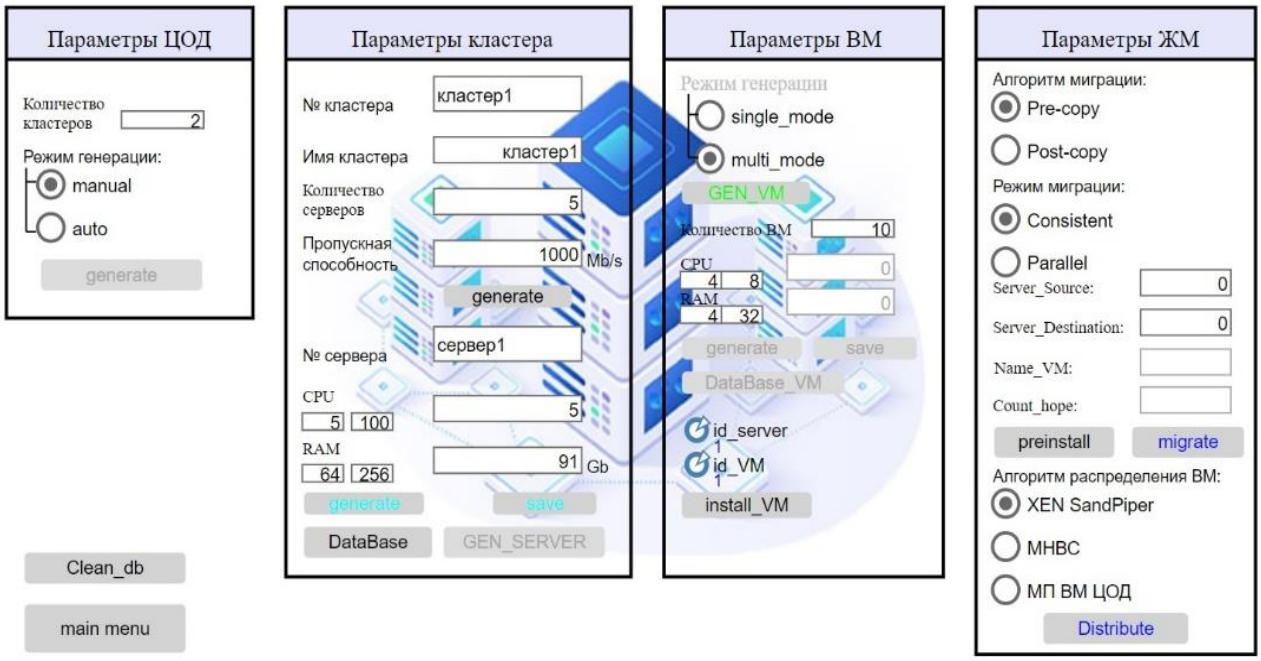


Рисунок 4.2 – Интерфейс настройки структуры ВЦОД

4.3 Экспериментальная оценка сходимости и вычислительной сложности алгоритма переразмещения виртуальных машин виртуализированного центра обработки данных

Для оценивания степени достижения цели исследования было принято решение о проведении имитационного эксперимента на разработанном симуляторе гетерогенного ВЦОД. Средой имитационного моделирования был выбран программный продукт AnyLogic (п. 4.2.1).

4.3.1 Проверка сходимости алгоритма переразмещения виртуальных машин виртуализированного центра обработки данных

Для проверки разработанного алгоритма переразмещения ВМ на свойство сходимости необходимо запустить алгоритм на разработанной имитационной модели ВЦОД и проверить остановится ли он после выполнения конечного числа шагов с выводом результата.

Конфигурация используемого в эксперименте компьютера включается в себя:

- процессор Intel Core i5-2500 CPU;
- ОЗУ – 16 Гб;
- видеоадаптер Intel HD Graphics;
- операционная система MS Windows 10 PRO.

	name_cluster	id_server	cpu	ram	bandwidth
1	Кластер	1	26	246	1,000
2	Кластер	2	29	143	1,000
3	Кластер	3	21	234	1,000
4	Кластер	4	28	249	1,000
5	Кластер	5	25	152	1,000
6	Кластер	6	18	216	1,000
7	Кластер	7	32	245	1,000
8	Кластер	8	32	192	1,000
9	Кластер	9	21	103	1,000
10	Кластер	10	16	192	1,000
*					

Рисунок 4.3 – Первоначальная конфигурация серверов в кластере до использования алгоритма МП ВМ ЦОД

	id_server	id_vm	cpu	ram
1	6	VM_1	4	160
2	5	VM_2	5	185
3	8	VM_3	7	139
4	7	VM_4	4	216
5	7	VM_5	4	238
6	6	VM_6	8	234
7	4	VM_7	5	131
8	1	VM_8	5	138
9	7	VM_9	7	189
10	5	VM_10	4	204
11	2	VM_11	6	225
12	8	VM_12	6	167
13	4	VM_13	8	240
14	2	VM_14	5	191
15	5	VM_15	4	165
16	1	VM_16	4	252
17	8	VM_17	8	215
18	9	VM_18	5	172
19	8	VM_19	7	217
20	8	VM_20	8	143
*				

Рисунок 4.4 – Первоначальное распределение ВМ в кластере ЦОД до использования алгоритма МП ВМ ЦОД

	name_cluster	id_server	cpu	ram	bandwidth
1	Кластер	1	26	246	1,000
2	Кластер	2	29	143	1,000
3	Кластер	3	21	234	1,000
4	Кластер	4	23	77	1,000
5	Кластер	5	25	152	1,000
6	Кластер	6	18	216	1,000
7	Кластер	7	32	245	1,000
8	Кластер	8	32	192	1,000
9	Кластер	9	26	275	1,000
10	Кластер	10	16	192	1,000
*					

Рисунок 4.5 – Конфигурация ЦОД после использования алгоритма МП ВМ ЦОД

	id_server	id_vm	cpu	ram
1	6	VM_1	4	160
2	5	VM_2	5	185
3	8	VM_3	7	139
4	7	VM_4	4	216
5	7	VM_5	4	238
6	6	VM_6	8	234
7	4	VM_7	5	131
8	1	VM_8	5	138
9	7	VM_9	7	189
10	5	VM_10	4	204
11	2	VM_11	6	225
12	8	VM_12	6	167
13	4	VM_13	8	240
14	2	VM_14	5	191
15	5	VM_15	4	165
16	1	VM_16	4	252
17	8	VM_17	8	215
18	4	VM_18	5	172
19	8	VM_19	7	217
20	8	VM_20	8	143
*				

Рисунок 4.6 – Распределение ВМ в кластере ЦОД после использования алгоритма МП ВМ ЦОД

Результаты проведения серии прогонов алгоритма переразмещения ВМ позволяют сделать вывод о том, что алгоритм обладает сходимостью.

4.3.1 Экспериментальная оценка вычислительной сложности алгоритма переразмещения виртуальных машин виртуализированного центра обработки данных

Теоретическая оценка вычислительной сложности разработанного алгоритма переразмещения ВМ была осуществлена в п. 3.4, согласно которой временная сложность алгоритма существенно зависит от числа ВМ и ФМ в ВЦОД.

Для проверки результата теоретической оценки вычислительной сложности разработанного алгоритма он был запущен на построенной имитационной модели ВЦОД.

Конфигурация используемого в эксперименте компьютера включается в себя:

- процессор Intel Core i5-2500 CPU;
- ОЗУ – 16 Гб;
- видеоадаптер Intel HD Graphics;
- операционная система MS Windows 10 PRO.

В ходе проведения эксперимента результаты сравнительного анализа времени работы алгоритма были сведены в Таблицу 4.2 и представлены на графике рисунка 4.7.

Таблица 4.2 Оценка временной сложности алгоритма переразмещения виртуальных машин

№ прогона	Размер входных данных, ФМ	Время работы алгоритма *t*, с
1	10	0,55
2	20	2,56
3	30	5,55
4	40	9,51
5	50	13,8
6	60	17,7
7	70	21,02
8	80	24,59
9	90	27,63
10	100	31,08

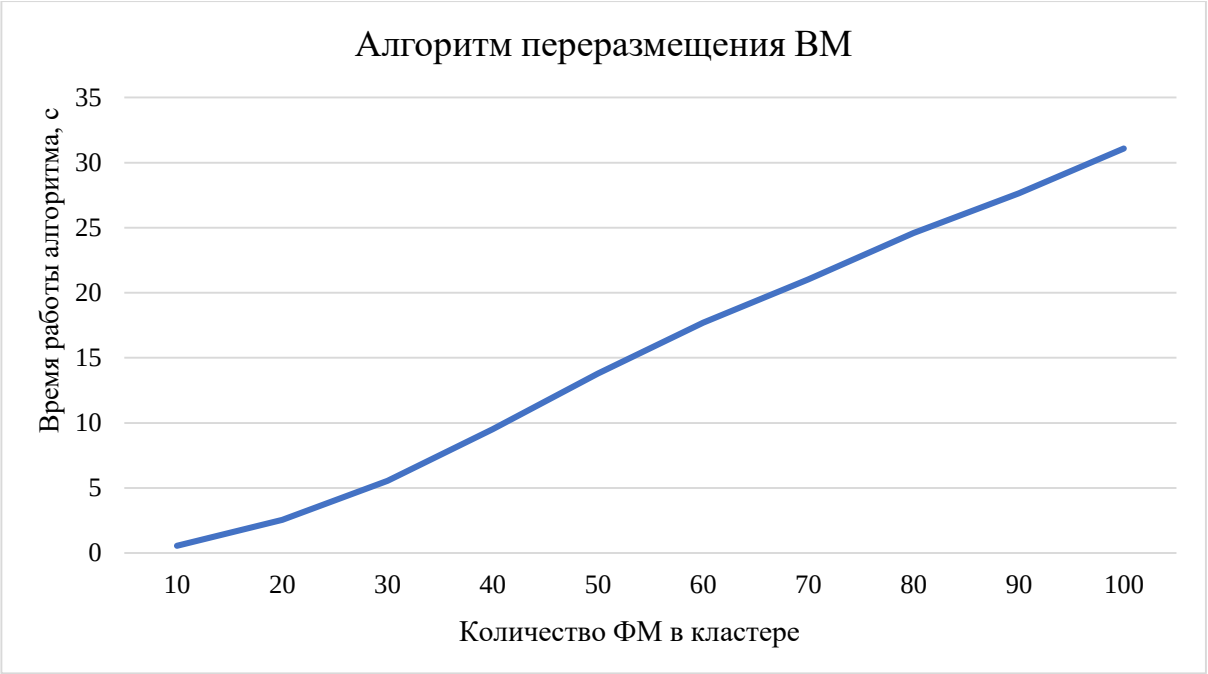


Рисунок 4.7 – Зависимость длительности работы алгоритма переразмещения ВМ от увеличения объема кластера ФМ

Результаты эксперимента позволяют сделать вывод, что время работы алгоритма переразмещения ВМ увеличивается линейно с увеличением размера кластера. Для кластера размером 100 ФМ время принятия решения составляет около 31,08 секунды, что является вполне приемлемым временем выполнения для автономного алгоритма.

Таким образом можно сделать вывод, что предложенный алгоритм переразмещения ВМ является быстрым и осуществимым методом в контексте крупномасштабных ЦОД.

4.4. Результаты экспериментального исследования переразмещения виртуальных машин в виртуализированном центре обработки данных

В рамках разработанной имитационной модели был спланирован сравнительный имитационный эксперимент для оценивания результата переразмещения ВМ в ВЦОД с различной структурой и степенью гетерогенности для:

- разработанного алгоритма переразмещения ВМ;

- алгоритма XSP – Xen SandPiper (функционирует только на гомогенных ЦОД с ГВ типа Xen);

- алгоритма VSL – Virtualized Server Load (функционирует только на гомогенных ЦОД с ГВ типа KVM).

Входные параметры для каждого прогона имитационного эксперимента случайным образом формировались соответствующими генераторами нагрузки имитационной модели: количество ФМ, тип ГВ каждой ФМ, начальные значения ВКР каждой ФМ, количество ВМ, запущенных на каждой ФМ, начальное значение ВКР, используемой каждой ВМ. Вид интерфейса сгенерированных параметров прогона имитационной модели представлен на рисунке 4.8. Исследуемые алгоритмы переразмещения ВМ подключались к модели, через AnyLogic Python API (библиотека AnyLogic Cloud).

Рисунок 4.8 – Интерфейс генерации параметров и переменных имитационного прогона

Для оценивания результата переразмещения ВМ в ВЦОД различной структурой и степенью гетерогенности будет проводиться имитационный эксперимент в рамках разработанной имитационной модели ВЦОД для:

- разработанного алгоритма переразмещения ВМ;

- алгоритма XSP – Xen SandPiper (функционирует только на гомогенных ЦОД с ГВ типа Xen);

- алгоритма VSL – Virtualized Server Load (функционирует только на гомогенных ЦОД с ГВ типа KVM).

Степень гетерогенности ВЦОД определяется фиксированным коэффициентом гетерогенности $K_{гет} = \{0;0,1;0,2;0,3;0,4;0,5\}$ – соотношения ФМ с ГВ Xen – g_1 и ГВ KVM – g_2. Если $K_{гет} = 0$, то ВЦОД полностью гомогенный с ГВ Xen или ГВ KVM, а если $K_{гет} = 0,5$, то ВЦОД максимально гетерогенный (50 ГВ Xen, 50 ГВ KVM).

Входные параметры для каждого прогона имитационного эксперимента формируются случайным образом по нормальному закону распределения [125] соответствующими генераторами нагрузки имитационной модели: количество ФМ – $N_{ФМ}$, тип ГВ каждой ФМ, начальные значения ВКР каждой ФМ, количество ВМ – $N_{ВМ}$, запущенных на каждой ФМ, начальное значение ВКР, используемой каждой ВМ.

В ходе эксперимента число ВМ имитирует нагрузку, создаваемую пользователями в реальном центре обработки данных. При возникновении задачи на переразмещение ВМ предсказать количество запущенных ВМ и потребляемые ими ресурсы достаточно сложно. Поэтому количество ВМ и их параметры будут определены здесь случайным образом.

Целью такого имитационного эксперимента является понимание предельных возможностей и слабых мест разработанного алгоритма переразмещения ВМ в условиях масштабируемости ЦОД. С увеличением числа ФМ на входе алгоритма, нагрузка на систему переразмещения возрастает, что может привести к ухудшению производительности и эффективности. Важно оценить, насколько алгоритм способен справляться с возрастающей сложностью задач при росте инфраструктуры.

Различные стратегии переразмещения и настройки алгоритмов могут показывать разную степень эффективности в зависимости от нагрузки и специфики задач, выполняемых в ЦОД. Использование имитационного подхода помогает моделировать реальные сценарии и выявлять потенциал оптимизации алгоритма на ранних этапах.

Показателем эффективности работы алгоритма переразмещения ВМ в ВЦОД является коэффициент использования ФМ $K_{исп}$, показывающего соотношение неактивных $ФМ_{НА}$ с общим количеством ФМ в кластере. Для каждого значения коэффициента гетерогенности будет проводиться 10 прогонов «до» использования алгоритмов перераспределения ресурсов и «после».

1) При $K_{гет} = 0$ усредненные результаты имитационного эксперимента по оцениванию значения коэффициента использования ФМ $K_{исп}$ после реализации исследуемых алгоритмов переразмещения ВМ для ВЦОД представлены на рисунке 4.9.

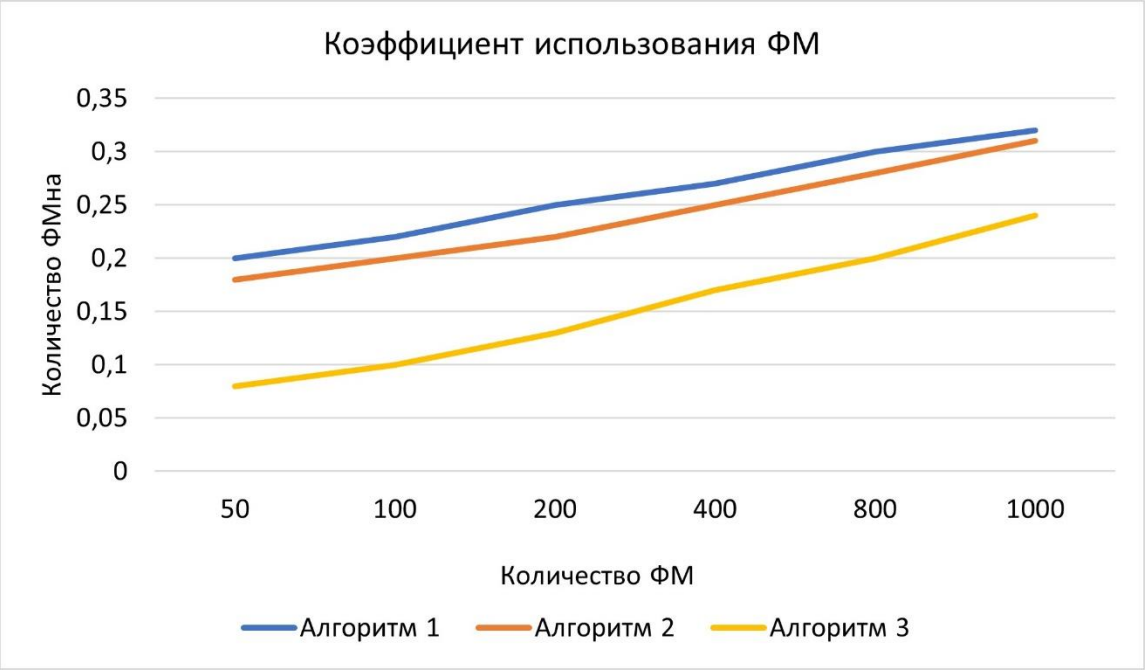


Рисунок 4.9 Коэффициент использования ФМ при $K_{гет} = 0$

Исходя из рисунка 4.9, можно заметить, что алгоритм переразмещения ВМ (алгоритм 3) уступает представленным альтернативным алгоритмам. Это

объясняется тем, что альтернативные алгоритмы проявляют наибольшую эффективность в гомогенных структурах ЦОД, поскольку им не нужно учитывать гетерогенность структуры ЦОД. Кроме того, работа этих алгоритмов, основанная на принципе «жадного алгоритма», приводит к минимальному количеству отказов при переразмещении ВМ, что также обеспечивает им преимущество в скорости выполнения по сравнению с алгоритмом 3.

2) При $K_{гет} = 0{,}1$ усредненные результаты имитационного эксперимента по оцениванию значения коэффициента использования ФМ $K_{исп}$ после реализации исследуемых алгоритмов переразмещения ВМ для ВЦОД представлены на рисунке 4.10.

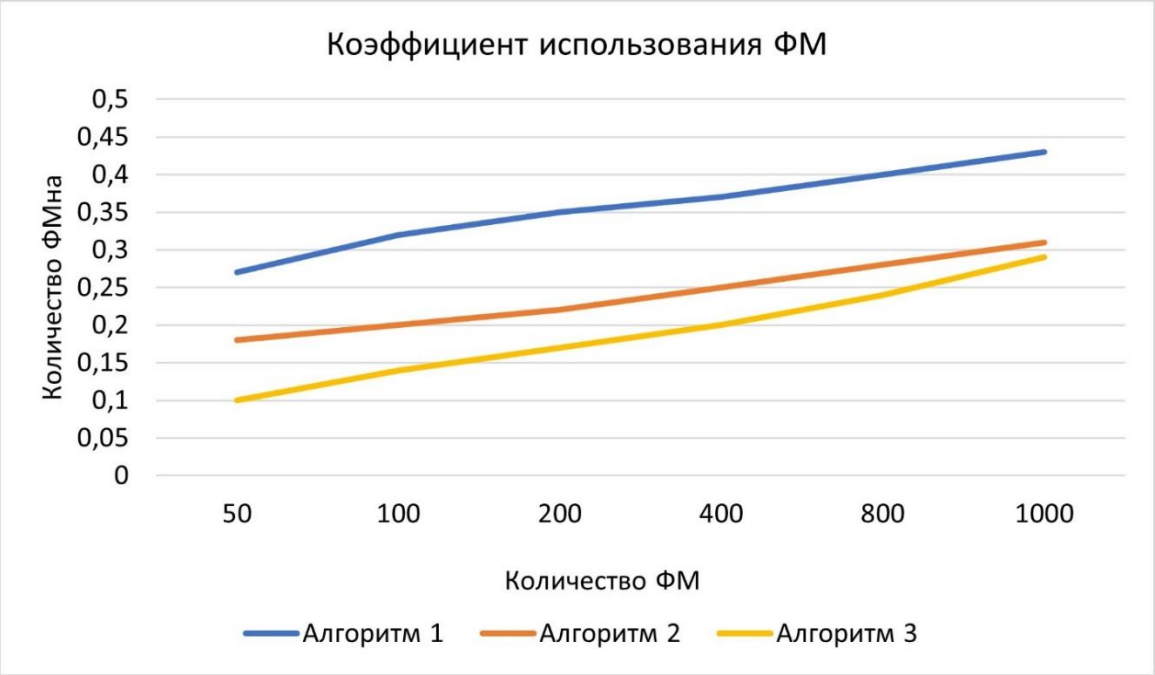


Рисунок 4.10 Коэффициент использования ФМ при $K_{гет} = 0{,}1$

3) При $K_{гет} = 0{,}2$ усредненные результаты имитационного эксперимента по оцениванию значения коэффициента использования ФМ $K_{исп}$ после реализации исследуемых алгоритмов переразмещения ВМ для ВЦОД представлены на рисунке 4.11.

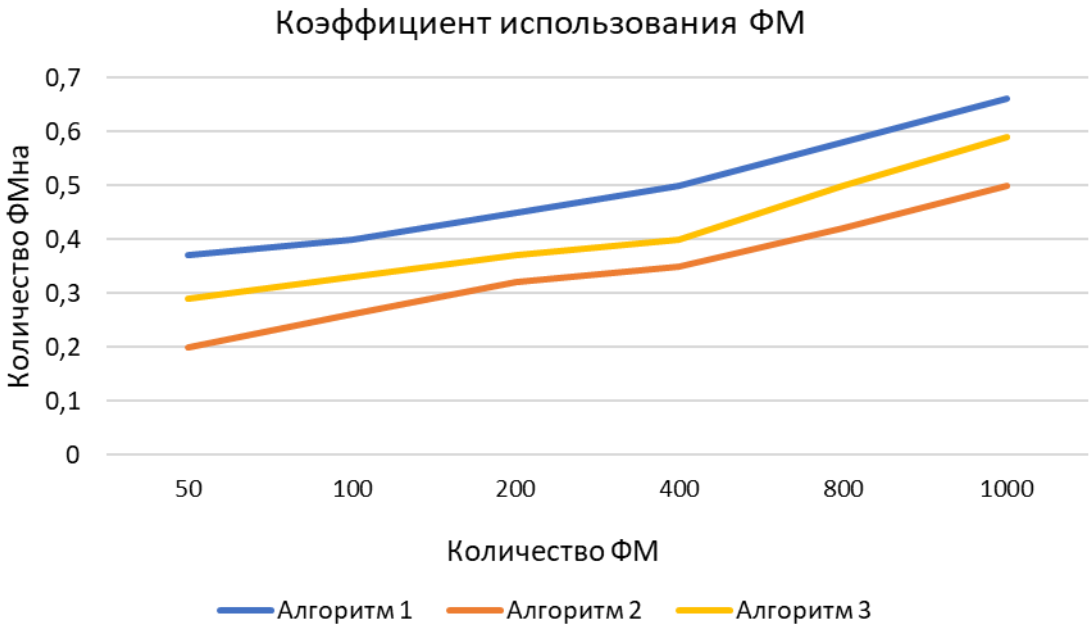


Рисунок 4.11 Коэффициент использования ФМ при $K_{гет} = 0,2$

Из рисунков 4.10 и 4.11 видно, что полученные значения коэффициента использования ФМ алгоритма переразмещения ВМ (алгоритм 3) приближаются к значениям коэффициентов использования ФМ альтернативных алгоритмов (алгоритм 1 и алгоритм 2), но все еще незначительно уступает им, несмотря на увеличение значения коэффициента гетерогенности структуры ЦОД.

При более тщательном анализе результатов работы алгоритмов 1 и 2 выявляется что, количество высвобожденных ФМ значительно меньше полученного значения, так как альтернативные алгоритмы не учитывают совместимость типов ГВ ФМ при принятии решения о ЖМ ВМ. Это приводит к увеличению числа отказов при осуществлении ЖМ ВМ через альтернативные алгоритмы.

3) При $K_{гет} = 0,3$ усредненные результаты имитационного эксперимента по оцениванию значения коэффициента использования ФМ $K_{исп}$ после реализации исследуемых алгоритмов переразмещения ВМ для ВЦОД представлены на рисунке 4.12.

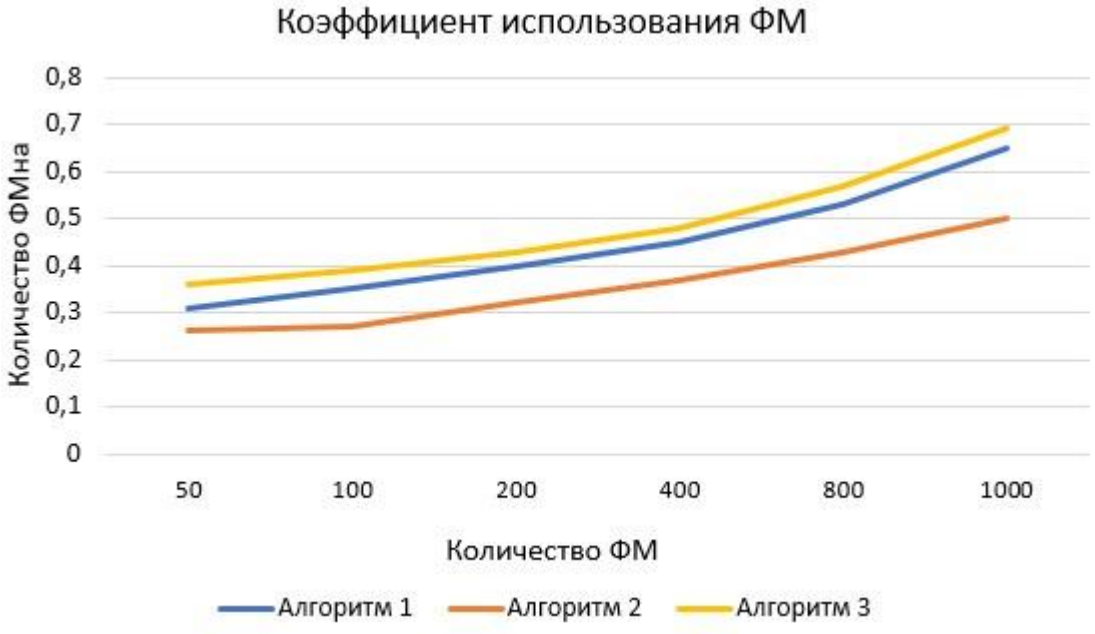


Рисунок 4.12 Коэффициент использования ФМ при $K_{гет} = 0,3$

4) При $K_{гет} = 0,4$ усредненные результаты имитационного эксперимента по оцениванию значения коэффициента использования ФМ $K_{исп}$ после реализации исследуемых алгоритмов переразмещения ВМ для ВЦОД представлены на рисунке 4.13.

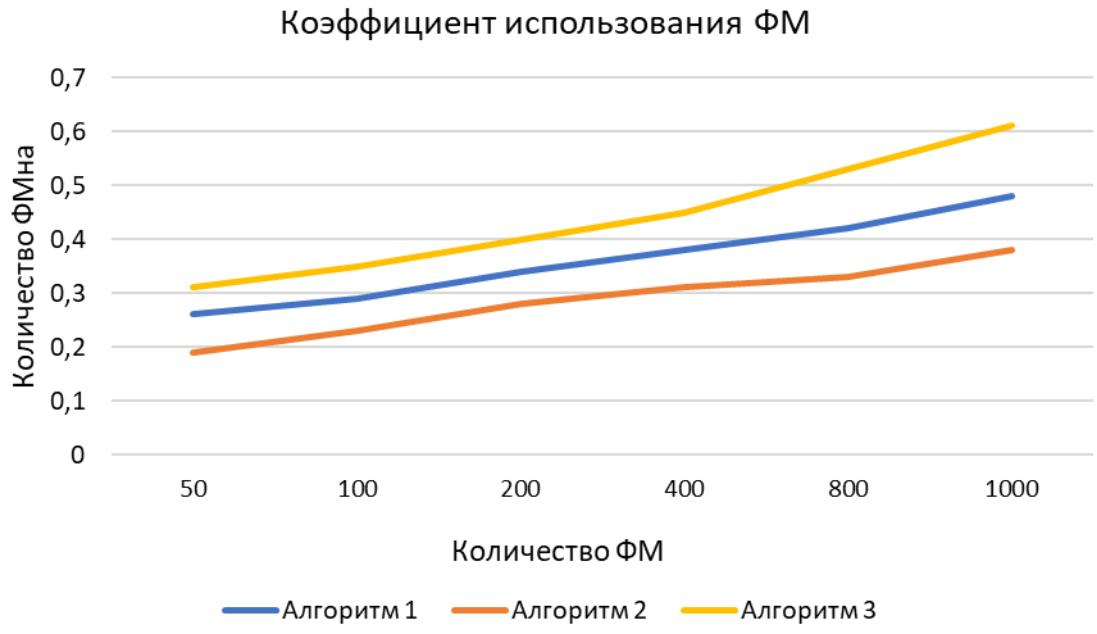


Рисунок 4.13 Коэффициент использования ФМ при $K_{гет} = 0,4$

5) При $K_{гет} = 0,5$ усредненные результаты имитационного эксперимента по оцениванию значения коэффициента использования ФМ $K_{исп}$ после реализации исследуемых алгоритмов переразмещения ВМ для ВЦОД представлены на рисунке 4.14.

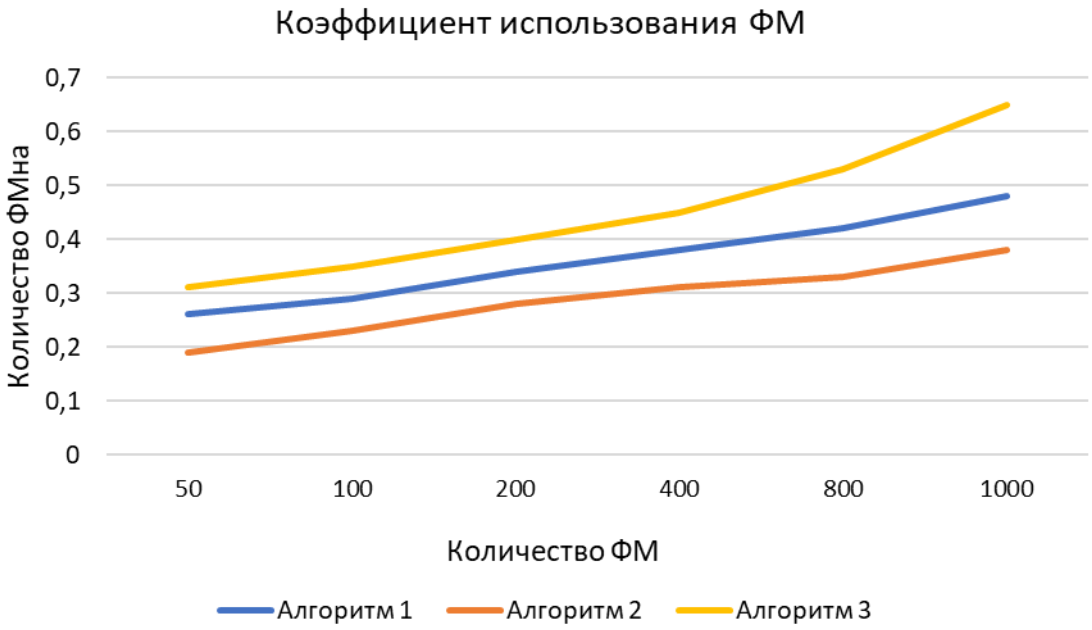


Рисунок 4.14 Коэффициент использования ФМ при $K_{гет} = 0,5$

Из рисунков 4.12, 4.13 и 4.14 видно, что алгоритм переразмещения ВМ превосходит рассматриваемые алгоритмы.

Усредненные результаты проведенных экспериментов имитационной модели представлены на рисунке 4.15.

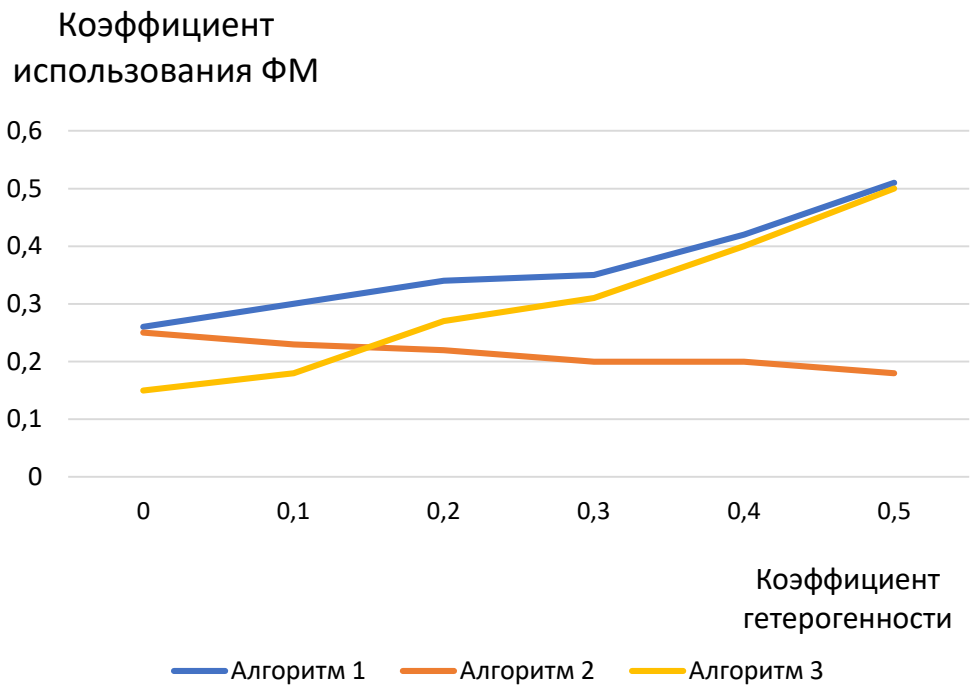


Рисунок 4.15 – Усредненные результаты имитационного эксперимента по оцениванию значения коэффициента использования ФМ после реализации исследуемых алгоритмов переразмещения ВМ для ВЦОД с коэффициентом гетерогенности

Из рисунка видно, что при увеличении коэффициента гетерогенности $K_{гет}$ (при начальной гомогенности ФМ ВЦОД для ГВ Xen) разработанный алгоритм переразмещения ВМ превосходит алгоритмов XSP и VSL, начиная со значений коэффициента гетерогенности от 0,3 до 0,5.

В ходе эксперимента особенное внимание уделяется метрикам, которые отражают качество и скорость переразмещения ВМ. К таким метрикам относятся метрики алгоритмов ЖМ ВМ: время миграции ВМ, степень использования ресурсов и временные интервалы простоев. Степень использования ресурсов оценивается коэффициентом утилизации K_{util}, который показывает соотношение используемых размещенными ВМ ресурсов на каждой активной $ФМ_A$ до и после переразмещения ВМ.

4.5 Выводы по главе

Разработанная система поддержки обеспечивает гибкость, масштабируемость и высокую степень адаптации к условиям современного ВЦОД. Она предоставляет возможность централизованного управления и мониторинга, поддерживая при этом требования кроссплатформенности и интеграции с существующими решениями в области мониторинга, что делает ее важным инструментом для обеспечения надежности и эффективности работы центр обработки данных.

Инновационное использование фреймворка PySyncObj и грамотное разделение функциональных зон на два архитектурных уровня позволяют достичь высоких показателей производительности и надежности в управлении ресурсами ВМ. Система поддержки переразмещения ВМ не только упрощает задачи администраторов, но и способствует более рациональному использованию вычислительных мощностей, что особенно важно в условиях современного динамичного развития IT-инфраструктуры.

Представлена разработанная архитектура программного комплекса поддержки процесса переразмещения ВМ в гетерогенных ВЦОД, которая обеспечивает динамическое переразмещение ВМ, основанное на дисбалансе виртуализированных ВКР предварительно сформированных подмножеств ФМ, имеющих гомогенную структуру. Результаты имитационного эксперимента подтверждают достижение цели исследования. На реализованные элементы разработанного ПО системы поддержки переразмещения ВМ получено свидетельство о регистрации программы для ЭВМ в реестре ФИПС.

ЗАКЛЮЧЕНИЕ

Диссертационная работа направлена на разработку средств математического и программного обеспечения переразмещения виртуальных машин виртуализированных ЦОД с гетерогенной структурой, обеспечивающего повышение эффективности использования ресурсов виртуализированных ЦОД с гетерогенной структурой.

Научная задача, решенная в диссертации, может быть классифицирована как задача применения известных научных методов в новой предметной области.

Достоверность и обоснованность полученных результатов подтверждается научно организованными экспериментами, корректным применением известных методов исследования, адекватных природе изучаемых процессов и явлений, непротиворечивостью и воспроизводимостью результатов, полученных в процессе сравнительного анализа вычислений и натурных экспериментов.

В процессе выполнения диссертационного исследования получены следующие основные результаты:

1. Проведен анализ проблематики процесса размещения и переразмещения виртуальных машин в виртуализированных ЦОД, включая вопросы особенности реализации алгоритмов живой миграции современных гипервизоров.

2. Разработана модель многомерного представления виртуализированных ресурсов, учитывающая нормированное значение их использования алгоритмами живой миграции и обеспечивающая возможность расчета их дисбаланса, определяющего необходимость выполнения процесса переразмещения виртуальных машин.

3. Разработан алгоритм переразмещения виртуальных машин, основанный на метаэвристике муравьиной колонии, отличающийся учетом при расчете матрицы миграции виртуальных машин гетерогенной структуры ЦОД и дополнительных ресурсов для алгоритмов живой миграции, и обеспечивающий получение квазиоптимальной матрицы миграции виртуальных машин для существующей структуры гетерогенных ЦОД.

4. Разработана архитектура программного комплекса поддержки процесса переразмещения виртуальных машин в гетерогенных виртуализированных ЦОД, отличающаяся наличием модуля формирования гомогенных подмножеств физических машин, и обеспечивающая формирование плана миграции виртуальных машин, основанном на дисбалансе ресурсов физических машин.

5. Проведено экспериментальное исследование по оцениванию эффективности использования ресурсов виртуализированного ЦОД на основе предложенного алгоритма решения задачи переразмещения ВМ для различных вариантов структуры виртуализированного ЦОД и выполняемых на ней виртуальных машин.

Рекомендации, перспективы дальнейшей разработки темы

Дальнейшие исследования стоит вести в направлении разработки нейросетевых методов оптимизации и их сравнительной оценки с предложенным эвристическим подходом по соотношению: время формирования матрицы миграции/коэффициент использования ФМ.

СПИСОК ТЕРМИНОВ, СОКРАЩЕНИЙ И УСЛОВНЫХ ОБОЗНАЧЕНИЙ

В настоящей работе применяются следующие сокращения:

ЦОД – центр обработки данных

SDDC – Software Defined Data Center, Программно-определяемый Центр Обработки Данных

ВЦОД – виртуализированный центр обработки данных

ИИ – искусственный интеллект

SaaS – Software as a Service, программное обеспечение как услуга

PaaS – Platform as a Service, платформа как услуга

IaaS – Infrastructure as a Service, инфраструктура как услуга

ВКР – вычислительные и коммуникационные ресурсы

СХД – системы хранения данных

SDN – software-defined Networking, программно-определяемая сеть

SDS – Software-Defined Storage, программно-определяемая сеть хранения

ITaaS – Information Technology as a service, Информационные Технологии как Услуга

DCaaS – Data Center as a Service, центр обработки данных как услуга

FaaS – Facilities as a Service, объект как услуга

ВМ – виртуальная машина

ФМ – физическая машина

ОС – операционная система

ПО – программное обеспечение

CPU – Central Processing Unit, центральный процессор

VCPU – Virtual Central Processor. Unit, виртуальный центральный процессор

NAS – Network-Attached Storage, сетевое хранилище данных

SAN – Storage Area Network, сеть хранения данных

VMM – Virtual Machine Monitor, диспетчер виртуальных машин

IDPS – Intrusion Detection Prevention System, система обнаружения и предотвращения вторжений

AWS – Amazon Web Services, веб-сервисы Amazon

VDI – Virtual Desktop Infrastructure, удаленный рабочий стол

TCO – Total Cost of Ownership, совокупная стоимость владения

ВТКС – виртуализированная телекоммуникационная структура

SLA – Service Level Agreement, соглашение об уровне сервиса

QoS – Quality of Service, качество обслуживания

NRC – Normalized Resource Cube, нормализованный ресурсный куб

ВНР – вектор наличия ресурсов

ВВР – вектор выделения ресурсов

ВОР – вектор оставшихся ресурсов

ВДР – вектор дисбаланса ресурсов

ВЗР – вектор запроса ресурсов

ПШР – планарный шестиугольник ресурсов

EDP – Extended Dot Product, расширенный точечный продукт

TUV – Total Utilization Volume, общий объем использования

VSR – Volume to Size Ratio, коэффициент компактности

VSL – Virtualized Server Load, нагрузка виртуализированного сервера

TSP – Travelling Salesman Problem, задача коммивояжера

ACO – Ant Colony Optimization, оптимизация муравьиной колонии

ИМ – искусственный муравей

AS – Ant System, муравьиная система

EAS – Elitist Ant System, элитарная муравьиная система

Ant-Q – Ant Q-learning System, муравьиная система с машинным обучением

MMAS – Max-min Ant System, Max-Min муравьиная система

AS-Rank – Ranked Ant System, система рейтингов муравьиной колонии

ACS – Ant Colony System, система муравьиной колонии

FFD – First-Fit-Decreasing, первый подходящий с упорядочиванием

СПИСОК ЛИТЕРАТУРЫ

Список литературы

1. Жукабаева Т.К., Кусаинова А.Т. Технология Больших данных (Big Data). Основные характеристики и перспективы применения [Электронный ресурс], ResearchGate, URL: https://www.researchgate.net.

2. Hemn B. A. A brief survey on big data: technologies, terminologies and data-intensive applications // Journal of Big Data. – November 2022. – Volume: 9(1). – P. 36. DOI: https://doi.org/10.1186/s40537-022-00659-3.

3. Maryoosh A.A., Hussein E.M. A Review: Data Mining Techniques and Its Applications // International Journal of Computer Science and Mobile Applications. – March 2022. – Volume: 10(3). – Pp. 1-14. DOI: https://doi.org/10.47760/ijcsma.2022.v10i03.001.

4. Li Z., Lida X., Zhongzhi S., Maoguang W., Wenjuan W. Distributed Computing Environment: Approaches and Applications // IEEE International Conference on Systems, Man and Cybernetics. – January 2007. – Pp. 3240-3244. DOI: https://doi.org/10.1109/ICSMC.2007.4413876.

5. Матюшок В.М., Красавина В.А., Матюшок С.В. Мировой рынок систем и технологий искусственного интеллекта: становление и тенденции развития // RUDN Journal of Economics. – August 2020. – Том 28(3). – С. 505-521. DOI: https://doi.org/10.22363/2313-2329-2020-28-3-505-521.

6. Miller, R. Inside Amazon’s Cloud Computing Infrastructure. [Электронный ресурс] URL: http://datacenterfrontier.com/inside-amazoncloud-computing-infrastructure/ September 2015. Режим доступа: 10.06.2024.

7. Bakhtiyar S., Oleg S. Building the Software Defined Data Center // Proceedings of the Institute for System Programming of RAS. – January 2018. – Volume: 30(6). – Pp. 7-24. DOI: https://doi.org/0.15514/ISPRAS-2018-30(6)-1.

8. Yaseein S.H., Maen A., Ahmed S.A., Saleh A. Data Centre Infrastructure: Design and Performance // Latest Advances and New Visions of Ontology in Information Science. – February 2023. – DOI: https://doi.org/10.5772/intechopen.109998.

9. Google is bringing AI to the classroom – in a big way [Электронный ресурс]. – URL: https://translated.turbopages.org/.

10. All About New Gemini AI Tool: Google’s New AI, December 2023 [Электронный ресурс] URL: https://translated.turbopages.org/proxy_u/en-ru.ru.d7f280e6-66f512fe-f3094fe7-74722d776562/https/www.geeksforgeeks.org/gemini-ai-tool-google-new-ai/ (дата обращения: 02.09.2024).

11. Google Cloud Platform, October 2016 [Электронный ресурс] URL: https://gcloud.readthedocs.io/_/downloads/en/stable/pdf/ (дата обращения: 02.09.2024).

12. Diego S., Vera P. Exploring TPUS for AI applications // Robotics & AI Club at IE University. – November 2023. – DOI: https://doi.org/10.48550/arXiv.2309.08918.

13. Leon P., Omid M., Joon O., Arjun S. Jupiter evolving: transforming google's datacenter network via optical circuit switches and software-defined networking // SIGCOMM '22: ACM SIGCOMM 2022 Conference. – August 2022. – DOI: https:// doi.org/10.1145/3544216.3544265.

14. Kieran M. Cloud adoption: enterprise AZURE OPENAI for financial services, January 2024 [Электронный ресурс] URL: https://www.capgemini.com/ (дата обращения: 02.09.2024).

15. Глобальная инфраструктура Azure [Электронный ресурс] URL: https://datacenters.microsoft.com/globe/explore?info=region_southcentralus (дата обращения: 02.09.2024).

16. Облачные сервисы для разработки и бизнес-задач [Электронный ресурс] URL: https://yandex.cloud/ru/ (дата обращения: 02.09.2024).

17. Батура Т.В., Мурзин Ф.А., Семич Д.Ф. Облачные технологии: основные модели, приложения, концепции и тенденции развития // Программные продукты и системы. – №3(107). – 2014. – стр. 64-72.

18. Shuo Z., Dong Y., Li P., Shijun L., Selling Reserved Instances through Pay-as-You-Go Model in Cloud Computing // IEEE International Conference on Web Services (ICWS). – June 2017. – DOI: https://doi.org/10.1109/ICWS.2017.25.

19. Lee B., Robert B., Shilong C. The NIST Definition of Cloud Computing, November 2011 [Электронный ресурс] URL: https://www.nist.gov/ (дата обращения: 02.09.2024).

20. Бумажкина Н.Ю. К вопросу о виртуализации вычислительных и телекоммуникационных ресурсов центра обработки данных / Н.Ю. Бумажкина // XXIX Международная научно-техническая конференция студентов, аспирантов и молодых ученых «Научная сессия ТУСУР – 2024». – Томск. – 2024. – С. 107-110.

21. Shabanov B. M., Samovarov O. I. Building the Software Defined Data Center // Proceedings of the Institute for System Programming of RAS. – January 2018. – DOI: https:// doi.org/ 10.15514/ISPRAS-2018-30(6)-1.

22. Аббасова Т.С., Артюшенко В.М. Виртуализация вычислительных мощностей в центрах обработки данных // Электротехнические комплексы и системы. – №4(5). – 2009, URL: https://cyberleninka.ru/article/n/virtualizatsiya-vychislitelnyh-moschnostey-v-tsentrah-obrabotki-dannyh/viewer.

23. Терешкевич А.А., Зубалов А.Н. Обзор технологии «Программно-конфигурируемые сети ПКС/SDN» // Газовая промышленность. – № 12(746). – 2016, URL: https://cyberleninka.ru/article/n/obzor-tehnologii-programmno-konfiguriruemye-seti-pks-sdn/viewer.

24. Солуковцев А. IBM Software Defined Storage: новые технологии, новое мышление // JETINFO. – №5(262). – 2015, URL: https://www.jetinfo.ru/interviews/ibm-software-defined-storage-novye-tehnologii-novoe-myshlenie/?ysclid=m1j4hknvk1105550840 (дата обращения: 02.09.2024).

25. Amir M.T. Rusli A. A Model of Information Technology as a Service (ITaaS) in Cloud Computing: A Case of Collaborative Knowledge Management System // Advanced Computer Science Applications and Technologies (ACSAT). – November 2012. – DOI: https://doi.org/10.1109/ACSAT.2012.24.

26. Комплексные инженерные решения [Электронный ресурс] URL: https://softline.ru/solutions/infrastructure-solutions/inzhenernyie-sistemyi (дата обращения: 02.09.2024).

27. Российская Федерация. Федеральный закон. О персональных данных. Федеральный закон № 152-ФЗ : текст с изменениями и дополнениями на 8 августа 2024 года : принят Государственной Думой 8 июля 2006 года : одобрен Советом Федерации 14 июля 2006 года. [Электронный ресурс] URL: https://www.consultant.ru/document/ cons_doc_LAW_61801/ (дата обращения: 02.09.2024).

28. Луковников М. Чем служит дата-центр [Электронный ресурс] URL: https://www.i-teco.ru/press-center/ayteko-v-smi/chem-usluzhit-data-tsentr/ (дата обращения: 02.09.2024).

29. Беляев Д. FaaS: ЦОД как услуга [Электронный ресурс] URL: https://dcforum.ru/sites/default/files/12.55-13.15_faas_ot_hpe_cod_ 2016_iks_ final_ belyaev.pdf (дата обращения: 02.09.2024).

30. Снытко А.С., Терников А.А. Особенности модернизации центра обработки данных и космоцентра // Программные продукты и системы. – 2015. – Том: №4. – Стр. 22-27. DOI: https://doi.org/10.15827/0236-235X.112.022-027.

31. Raghava S.S.D. IAAS cloud architecture distributed cloud infra structures and virtualized data centers // International Research Journal of Modernization in Engineering Technology and Science. – July 2023. – DOI: https://doi.org/10.56726/IRJMETS43486.

32. Congfeng J. Xianghua X. Jilin Z. Resource Allocation in Contending Virtualized Environments through VM Performance Modeling and Feedback // Journal of Information Science and Engineering. – September 2011. – Volume: 29(2). – Pp. 196 - 203. DOI: https://doi.org/10.1109/ChinaGrid.2011.44.

33. Amol J., Syed A.R.S., Seo-Y.N., Sangwook B. Performance Analysis of NAS and SAN Storage for Scientific Workflow // Conference: 2016 International Conference on Platform Technology and Service (PlatCon). – February 2016. – DOI: https://doi.org/10.1109/PlatCon.2016.7456814.

34. Dan C.M. Chapter 5 – Cloud Resource Virtualization // Cloud Computing. – 2013. – Pp. 131-161. DOI: https://doi.org/10.1016/B978-0-12-404627-6.00005-1.

35. Nagarajan A., Mueller F., Engelmann C., Scott S. Proactive fault tolerance for HPC with XEN virtualization // Proceedings of the 21st Annual International Conference on Supercomputing, ACM. – June 2007. – Pp. 23–32. DOI: https://doi.org/10.1145/1274971.1274978.

36. Kernel Virtual Machine – Features [Электронный ресурс] URL: https://linux-kvm.org/page/KVM_Features/ (дата обращения: 02.09.2024).

37. Обзор технологии Hyper-V. [Электронный ресурс] URL: https://learn.microsoft.com/ru-ru/windows-server/virtualization/hyper-v/hyper-v-technology-overview/ (дата обращения: 02.09.2024).

38. VMware ESXi [Электронный ресурс] URL: https://www.vmware.com/products/cloud-infrastructure/esxi-and-esx (дата обращения: 02.09.2024).

39. Обзор продуктов VMware: Виртуализация и вычисления для конечных пользователей [Электронный ресурс] URL: https://www.itc.by/obzor-produktov-vmware-virtualizacziya-i-vychisleniya-dlya-konechnyh-polzovatelej/ (дата обращения: 02.09.2024).

40. Данилин Д.Д., Мартышкин А.И., Данилов Е.А. Виртуализация вычислительных машин и центры обработки данных в вычислительных сетях [Электронный ресурс] URL: https://doicode.ru/doifile/lj/97/trnio-05-2023-658.pdf (дата обращения: 02.09.2024).

41. Преимущества виртуальных дата-центров (VDC) 41 [Электронный ресурс] URL: https://www.xelent.ru/blog/preimushchestva-virtualnykh-data-tsentrov-vdc/?ysclid=m1jcujk1wp446444825 (дата обращения: 02.09.2024).

42. DATA CENTERS [Электронный ресурс] URL: https://www.corporate.carrier.com/our-company/innovation/data-centers/ (дата обращения: 02.09.2024).

43. Data center virtualization market revenue worldwide in 2022 versus 2032 [Электронный ресурс] URL: https://www.statista.com/statistics/1254835/application-virtualization-market-size/ (дата обращения: 02.09.2024).

44. Тенденции на российском рынке дата-центров [Электронный ресурс] URL: https://www.tadviser.ru/index.php/Статья:Тенденции_на_российском_рынке_дата-центров?ysclid=m1jd7vdc74854063603 (дата обращения: 02.09.2024).

45. Выручка «Группы Астра» выросла на 58% в первом полугодии 2024 года [Электронный ресурс] URL: https://astragroup.ru/about/press-center/news/vyruchka-gruppy-astra-vyrosla-na-58-v-pervom-polugodii-2024-goda/ (дата обращения: 02.09.2024).

46. «Базис» подвел финансовые и операционные итоги 2023 года [Электронный ресурс] URL: https://basistech.ru/news/bazis-podvel-finansovye-i-operaczionnye-itogi-2023-goda/ (дата обращения: 02.09.2024).

47. Bliedy D., Mazen S. and Ezzat E. Datacenter Total Cost of Ownership (TCO) Models: a survey // International Journal of Computer Science, Engineering and Applications. – August 2018. – Volume: 8. DOI: https://doi.org/10.5121/ijcsea.2018.8404.

48. Handlin H. Using a Total Cost of Ownership (TCO) Model for Your Data Center [Электронный ресурс] URL: https://www.datacenterknowledge.com/business/using-a-total-cost-of-ownership-tco-model-for-your-data-center (дата обращения: 02.09.2024).

49. Zhang Q., Cheng L., Boutaba R. Cloud computing: state-of-the-art and research challenges // Journal of Internet Services and Applications. – May 2010. – Volume: 1(1). – Pp. 7–18. DOI: https://doi.org/10.1007/s13174-010-0007-6.

50. Yan W., Yao J., Zhang Y. LT-TCO: A TCO Calculation Model of Data Centers for Long-Term Data Preservation // IEEE International Conference on Networking, Architecture and Storage (NAS). – August 2019. – Pp. 112-134. DOI: https://doi.org/10.1109/NAS.2019.8834714.

51. Yujian Zhang, Y., Li, M., Tong, F. Energy-efficient load balancing for divisible tasks on heterogeneous clusters // Transactions on Emerging

Telecommunications Technologies. – July 2023. – Volume: 34(5). DOI: https://doi.org/10.1002/ett.4829.

52. Grot B., Hardy D., Lofti-Kamran P., Falsaf, B., Nicopoulos C., Sazeindes Y. Optimizing data-center TCO with scale-out processors // IEEE Micro. – September 2012. – Volume: 32(5). – Pp. 52-63. DOI: https://doi.org/10.1109/MM.2012.71.

53. Cui Y., Ingalz C., Gao T., Heydari A. Total cost of ownership model for data center technology evaluation // 16th IEEE Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems (ITherm). – May 2017. – Pp.112-134. DOI: https://doi.org/ 10.1109/ITHERM.2017.7992587.

54. Bumazhkina N.Yu., Zaharova I.N., Kochkurov A.E. On the use of live migration technologies for virtual machines in the task of optimizing data center resources // Материалы of the XXIX-the International Open Science Conference «Modern informatization problems». – 2024. –Volume: 2. – Pp. 133-137. (In Russ.).

55. Колчин И.В., Филлипов С.Н. Архитектура автономного микрогипервизора реального времени и измерение его временных характеристик // Информационно-управляющие системы. – Том 3(70). – 2014. – С. 57 –67.

56. Mollamotalebi M. Shahnaz H. Multi-objective dynamic management of virtual machines in cloud environments // Journal of Cloud Computing. – July 2017. – Volume: 6(1). DOI: https://doi.org/10.1186/s13677-017-0086-z.

57. Ghasemi A., Haghighat A. T., Keshavarzi A. Enhanced multi-objective virtual machine replacement in cloud data centers: combinations of fuzzy logic with reinforcement learning and biogeography-based optimization algorithms // Cluster Computing. – October 2022. – Volume: 26(4). – Pp. 1-14. DOI: https://doi.org/10.1007/s10586-022-03794-x.

58. Tang S., He B., Liu H., Lee B.-S. Chapter 7 – Resource Management in Big Data Processing Systems // Big Data Principles and Paradigms. – 2016. – Pp. 161-188. DOI: https://doi.org/10.1016/B978-0-12-805394-2.00007-6.

59. Verma A, Kumar G, Koller R, Sen A. Cosmig: Modeling the impact of reconfiguration in a cloud // IEEE Annual International Symposium on Modelling, Analysis, and Simulation of Computer and Telecommunication Systems. – July 2011. – Pp. 3–11. DOI: https://doi.org/10.1109/MASCOTS.2011.37.

60. Beloglazov A, Buyya R. Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of Virtual Machines in Cloud data centers. Concurrency and Computation: Practice and Experience 2012; 24(13):1397–1420.

61. Tai J., Sheng B., Yao Y., Mi N. Live Data Migration for Reducing SLA Violations in Multi-tiered Storage Systems // IEEE International Conference on Cloud Engineering (IC2E). – March 2014. DOI: https://doi.org/10.1109/IC2E.2014.8.

62. Clark C., Fraser K., Hand S., Hansen J., Jul E. Live migration of virtual machines // 2nd Symposium on Networked Systems Design and Implementation. – May 2005. – Volume: 2. – Pp. 273–286.

63. Sturm R., Pollard C., Craig J. Chapter 5 – Cloud Resource Virtualization // Managing Applications for Cloud, Mobile, IoT and eBusiness. – 2017. – Pp. 53-69. DOI: https://doi.org/10.1016/B978-0-12-804018-8.00005-X.

64. Алексанков С.М. Модель динамической миграции виртуальных машин с гибридным подходом // Научно-методический вестник информационных технологий, механики и оптики – Июль 2017. –Том 17(4).– Стр. 725-732. DOI: https://doi.org/10.17586/2226-1494-2017-17-4-725-732.

65. Choudhary A., Govil M., Singh G., Awasthi L., Pilli E., Kapil D. A critical survey of live virtual machine migration techniques // Journal of Cloud Computing: Advances, Systems and Applications. – November 2017. – Volume: 6(23). – Pp. 2-41. DOI: https://doi.org/10.1186/s13677-017-0092-1.

66. Руссинович М. и др. Внутреннее устройство Windows. 7-е издание / М. Руссинович, А. Ионеску, Д. Соломон, П. Йосифович – СПб: Питер, 2021. – 944 с. ISBN 978-5-4461-0663-9.

67. Уорд Б. Внутреннее устройство Linux. 3-е изд. / Б. Уорд – СПб.: Питер, 2022. – 480 с.: ил. – (серия для профессионалов).

68. Shrivastava V., Zerfos P., Lee Kw., Jamjoom H., Liu YH., Banerjee S. Application-aware Virtual Machine migration in data centers // 30th IEEE International Conference on Computer Communications. – May 2011. – Pp. 66–70. DOI: https://doi.org/10.1109/INFCOM.2011.5935247.

69. Shakya A., Garg D., Nayak P. Hybrid Live VM Migration: An Efficient Live VM Migration Approach in Cloud Computing // Advanced Informatics for Computing Research. – December 2018. – Pp. 600–611. DOI: https://doi.org/10.1007/978-981-13-3140-4_54.

70. Mishra M., Sahoo A. On theory of VM placement: Anomalies in existing methodologies and their mitigation using a novel vector based approach // IEEE International Conference on Cloud Computing (CLOUD). – July 2011. – Pp. 275–282. IEEE. DOI: https://doi.org/10.1109/CLOUD.2011.38.

71. Cerroni W. Network performance of multiple virtual machine live migration in cloud federations // Journal of Internet Services and Applications. – December 2015. – Volume: 6(1). DOI: https://doi.org/10.18411/spc-22-01-2018-02.

72. Устранение неполадок в развертывании при перезагрузке или изменении размера существующей виртуальной машины Windows в Azure [Электронный ресурс] URL: https://learn.microsoft.com/ru-ru/troubleshoot/azure/virtual-machines/windows/restart-resize-error-troubleshooting (дата обращения: 02.09.2024).

73. Wang J., Gu H., Yu J., Song Y. Research on virtual machine consolidation strategy based on combined prediction and energy-aware in cloud computing platform // Journal of Cloud Computing. – May 2022. DOI: https://doi.org/10.21203/rs.3.rs-1645107/v1.

74. Christensena H.I., Khanb A., Pokutta S., Tetali P. Approximation and online algorithms for multidimensional bin-packing: A survey // Computer Science Review. – 2017. – Volume:24. – Pp. 63–79. DOI: https://doi.org/10.1016/j.cosrev.2016.12.

75. Coffman E.G. Jr., Csirik J., Galambos G., Martello S., Vigo D. Bin Packing Approximation Algorithms: Survey and Classification // Handbook of Combinatorial Optimization. – January 2013. – Pp. 455-531. DOI: https://doi.org/10.1007/978-1-4419-7997-1_35.

76. Захаров В.Н., Мунерман В.И. Модели и методы параллельной обработки структурированных больших данных // Современные информационные технологии и ИТ-образование. – 2014. – Том: 10. Стр. 534-547.

77. Alperin J.L. Local Representation Theory: Modular Representations as an Introduction to the Local Representation Theory of Finite Groups // Cambridge University Press. – 1986. ISBN978-0-521-44926-7. P. 178.

78. Белов А.С., Бумажкина Н.Ю. Подход к моделированию виртуализированных вычислительных и коммуникационных ресурсов центра обработки данных с учетом накладных расходов на процесс живой миграции виртуальных машин // Системы управления и информационные технологии. – Воронеж. – 2024. – Том: 2(96). – Стр. 8-14.

79. Покинтелица А.Е. Проблемы и специфика редукции данных в автономных робототехнических системах // Problems of Artificial Intelligence. – 2023. –Том: 1(28). – Стр. 31-41.

80. What is Amazon Elastic Block Store? [Электронный ресурс] URL: https://docs.amazonaws.cn/en_us/ebs/latest/userguide/what-is-ebs.html (дата обращения: 02.09.2024).

81. Sun X., Su S., Xu P. Optimizing multi-dimensional resource utilization in virtual data center // 4th IEEE International Conference on Broadband Network and Multimedia Technology. – October 2011. DOI: https://doi.org/doi:10.1109/icbnmt.2011.615.

82. Xiong A., Xu Ch. EnergyEfficient Multiresource Allocation of Virtual Machine Based on PSO in Cloud Data Center // Theory and Applications of Complex Networks. – June 2014. DOI: https://doi.org/10.1155/2014/816518.

83. Pagare J.D., Koli N. A technical review on comparison of XEN and KVM hypervisors an analysis of virtualization technologies // International Journal of Advanced Research in Computer and Communication Engineering. – December 2014. – Volume: 3(12). – DOI: https://doi.org/10.17148/IJARCCE.2014.31234.

84. Zu Y., Huang T., Zhu Y. An Efficient Power-Aware Resource Scheduling Strategy in Virtualized Datacenters // International Conference on Parallel and Distributed Systems. – December 2013. DOI: https://doi.org/10.1109/ICPADS.2013.27.

85. Sun X. Ansari N., Wang R. Optimizing Resource Utilization of a Data Center // IEEE Communications Surveys & Tutorials. – April 2016. – Volume: 18(4). DOI: https://doi.org/10.1109/COMST.2016.2558203.

86. Panwar S.S., Rauthan M.M.S., Barthwal V. A systematic review on effective energy utilization management strategies in cloud data centers // Journal of Cloud Computing. – December 2022. – Volume: 11(1). DOI: https://doi.org/10.1186/s13677-022-00368-5.

87. Sharkh M.A., Shami A., Ouda A. Optimal and suboptimal resource allocation techniques in cloud computing data centers // Journal of Cloud Computing. – March 2017. – Volume: 6(1). DOI: https://doi.org/ 10.1186/s13677-017-0075-2.

88. Ivanisenko I., Kirichenko L., Radivilova T. Методы балансировки с учетом мультифрактальных свойств нагрузки // International Journal "Information Content and Processing". – Number 2015. – Volume: 2. – Pp. 345-368.

89. Beloglazov A. Buyya R. Managing Overloaded Hosts for Dynamic Consolidation of Virtual Machines in Cloud Data Centers under Quality of Service Constraints // IEEE Transactions on Parallel and Distributed Systems. – July 2013. – Volume: 24(7). Pp. 1366-1379. DOI: https://doi.org/10.1109/TPDS.2012.240.

90. Niles S., Donovan P. Virtualization and Cloud Computing: Optimized Power, Cooling, and Management Maximizes Benefits [Электронный ресурс] URL: https://www.sanitysolutions.com/wp-content/uploads/2018/02/snis-7aulcp-r4-en.pdf.

91. Mergenci C., Korpeoglu I. Generic resource allocation metrics and methods for heterogeneous cloud infrastructures // Journal of Network and Computer Applications. – July 2019. – Volume: 146. DOI: https://doi.org/10.1016/j.jnca.2019.102413.

92. Mishra M. Sahoo A. On Theory of VM Placement: Anomalies in Existing Methodologies and Their Mitigation Using a Novel Vector Based Approach // IEEE International Conference on Cloud Computing. – July 2011. DOI: https://doi.org/10.1109/CLOUD.2011.38.

93. Rathor V.S., Pateriya R.K., Gupta R. Survey on Load Balancing through Virtual Machine Scheduling in cloud computing Environment // International Journal of Cloud Computing and Services Science. – April 2014. – Volume: 3(1). DOI: https://doi.org/10.11591/closer.v3i1.5864.

94. Li Y., Liu Y. Wang W. Planar Hexagonal Meshing for Architecture // IEEE Transactions on Visualization and Computer Graphics. – September 2015. – Volume: 21(1). – Pp. 95-106. DOI: https://doi.org/10.1109/TVCG.2014.2322367.

95. Almohimeed I., Sikkandar Y.M., Mohanarathinam A., Parvathy V.S. Sandpiper Optimization Algorithm With Region Growing Based Robust Retinal Blood Vessel Segmentation Approach // IEEE Access PP. – January 2024. DOI: https://doi.org/10.1109/ACCESS.2024.3365273.

96. Caprara A., Toth P. Lower bounds and algorithms for the 2-dimensional vector packing problem // Discrete Applied Mathematics. – August 2001. – Volume: 111(3). – Pp. :231–262. DOI: https://doi.org/10.1016/S0166-218X(00)00267-5.

97. Arzuaga E. Kaeli D. Quantifying load imbalance on virtualized enterprise servers // The first joint WOSP/SIPEW International Conference on Performance Engineering. – January 2010. – DOI: https://doi.org/10.1145/1712605.1712641.

98. Blum C. Roli A. Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison // ACM Computing Surveys. – January 2001. – Volume: 35. – Pp. 268-308. DOI: https://doi.org/10.1145/937503.937505.

99. Dorigo M., St¨utzle T. The Ant Colony Optimization Metaheuristic: Algorithms, Applications, and Advances // Computer Science. – 2003. – 250–285 p. DOI: https://doi.org/10.1007/0-306-48056-5_9.

100. Пантелеев А.В. Метаэвристические алгоритмы поиска глобального экстремума : [монография] / Пантелеев А.В. – М. : Изд-во МАИ-ПРИНТ, 2009. – 159 с. ISBN 978-5-7035-2115-1.

101. Glover F.W., Kochenberger G.A. Handbook of Metaheuristics // International Series in Operations Research & Management Science. – 2003. – 557 p. DOI: https://doi.org/10.1007/b101874.

102. Nesmachnow S. An overview of metaheuristics: accurate and efficient methods for optimization // International Journal of Metaheuristics. – 2014. Volume: 3(4). – Pp. 320–347. DOI: https://doi.org/10.1504/ijmheur.2014.068914.

103. Pintea C-M., Pop P.C., Chira C. The generalized traveling salesman problem solved with ant algorithms // Complex Adaptive Systems Modeling. – August 2017. – Volume: 13(7). DOI: https://doi.org/10.1186/s40294-017-0048-9.

104. Reeves C. Hybrid genetic algorithms for bin-packing and related problems // Annals of Operations Research. – January 1998. – Volume: 63(3). – Pp. 371–396. DOI: https://doi.org/10.1007/BF02125404.

105. Kaaouache M.A., Bouamama S. Solving bin Packing Problem with a Hybrid Genetic Algorithm for VM Placement in Cloud // Procedia Computer Science. – December 2015. – Volume: 60(1). – Pp. 1061–1069. DOI: https://doi.org/10.1016/j.procs.2015.08.151.

106. Dorigo M., Birattari M., Stutzle T. Ant colony optimization // IEEE Computational Intelligence Magazine. – 2006. – Volume: 1(4). – Pp. 28–39. DOI: https://doi.org/10.1109/mci.2006.329691.

107. Dorigo M. Maniezzo V. Colorni A. Ant System: An Autocatalytic Optimizing Process // Technical Report 91-016. – February 1999.

108. Gambardella, L.M. Ant Colony System: A cooperative learning approach to the Traveling Salesman Problem // IEEE Transactions on Evolutionary Computation. – May 1997. – Volume: 1(1). Pp. 53 – 66. DOI: https://doi.org/10.1109/4235.585892.

109. Saramud M.V. Kovalev I.V. Kovalev D.I. Voroshilova A.A. Modification of the ant colony algorithm for multiversion software application development // IOP Conference Series Materials Science and Engineering. – February 2021. – Volume: 1047(1):012155. DOI: https://doi.org/10.1088/1757-899X/1047/1/012155.

110. Семенов С.С., Педан А.В., Воловников В.С., Климов И.С. Анализ трудоемкости различных алгоритмических подходов для решения задачи коммивояжера // Системы управления, связи и безопасности. – 2017. – Том №1. – Стр. 116-131. URL: http://sccs.intelgr.com/archive/2017-01/08-Semenov.pdf.

111. Colorni A., Dorigo M., Maniezzo V. Distributed Optimization by Ant Colonies // European Conference on Artificial Life. – January 1991. – Pp. 134—142.

112. Dósa G. The tight bound of first fit decreasing bin-packing algorithm is FFD(I) ≤ 11/9 OPT(I) + 6/9 // Combinatorics, Algorithms, Probabilistic and Experimental Methodologies, First International Symposium. – January 2007. DOI: https://doi.org/10.1007/978-3-540-74450-4_1.

113. Dorigo M. Ottimizzazione, apprendimento automatico, ed algoritmi basati su metafora naturale (Optimization, Learning, and Natural Algorithms) // PhD “Doctorate in Systems and Information Electronic Engineering”, Politecnico di Milano. – 1992.

114. Gambardella L.M., Dorigo M. An Ant Colony System Hybridized with a New Local Search for the Sequential Ordering Problem // INFORMS Journal on Computing. – August 2000. – Volume: 12(3). – Pp. 237-255. DOI: https://doi.org/10.1287/ijoc.12.3.237.12636.

115. Chekuri C. Khanna S. On Multidimensional Packing Problems // SIAM Journal on Computing. – May 2004. Volume: 33(4). – Pp. 837-851. DOI: https://doi.org/10.1137/S0097539799356265.

116. Бумажкина Н.Ю. Алгоритм перераспределения виртуализированных вычислительных и коммуникационных ресурсов центра обработки данных на основе метаэвристики ACS / Н.Ю. Бумажкина // Моделирование, оптимизация и информационные технологии. – 2024. – Том: 11(3). Режим доступа: https://moitvivt.ru/ru/journal/article?id=1367.

117. Bumazhkina N.Yu., Rubtsov A. A., Mikhalev P. V. An algorithm for obtaining an ensemble machine learning model for node federation in decentralized machine learning systems // AIP Conference Proceedings. Manuscript № AIPCP21-AR-MIP 2024-00155.

118. Борзова Н.Ю., Белов А.С., Трахинин Е.Л., Добрышин М.М., Слесарева К.М. Программа оценки эффективности комплексного использования информационных ресурсов в системе распределенных ситуационных центров. – Свидетельство о государственной регистрации программы для ЭВМ № 2022617593. - М: ФИПС, 2022.

119. Базеева Н.А., Овтин Р.А. Нотация BIG O и сложность алгоритмов // Информационные технологии. URL: https://cyberleninka.ru/article/n/notatsiya-big-o-i-slozhnost-algoritmov/viewer.

120. Bae S. JavaScript Data Structures and Algorithms: An Introduction to Understanding and Implementing Core Data Structure and Algorithm // Library of Congress Control Number: 2019930417. – 2019. DOI: https://doi.org/10.1007/978-1-4842-3988-9.

121. What is a Distributed System? [Электронный ресурс] URL: https://www.geeksforgeeks.org/what-is-a-distributed-system/ (дата обращения: 02.09.2024).

122. Федорова Л.М. Системы мониторинга. Обзор и сравнение // Вестник науки и образования № 10(88). – 2020. – Том:4. Стр. 16-18.

123. Ramasubramani V., Adorf C.S., Dice B.D. signac: A Python framework for data and workflow management // 17th Python in Science Conference At: Austin, TX. – July 2018. – DOI: https://doi.org/10.25080/Majora-4af1f417-016.

124. Ильин Д.Ю., Никульчев Е.В., Колясников П.В. Выбор технологических решений для разработки программного обеспечения распределенных информационных систем // Современные информационные технологии и ИТ-образование. – Июль 2018. – Том 14(2). DOI: https://doi.org/10.25559/SITITO.14.201802.344-354.

125. Майер Р.В. Компьютерное моделирование: учебно-методическое пособие для студентов педагогических вузов [Электронное учебное издание на компакт диске]. – Глазов: Глазов. гос. пед. ин-т. – 2015. – 24,3 Мб (620 с.)

126. Аверина Т.А. Построение алгоритмов статистического моделирования систем со случайной структурой: учеб. пособие / Т.А. Аверина– Новосибирск : РИЦ НГУ. – 2015. – 155 с.

127. Маликов Р.Ф. Практикум по имитационному моделированию сложных систем в среде AnyLogic 6 / Р.Ф. Маликов учебное пособие. Уфа: Изд-во БГПУ. – 2013. – 298 с.

128. Стенников В.А., Барахтенко Е.А., Майоров Г.С. Разработка мультиагентной модели интегрированной энергоснабжающей системы в программной среде AnyLogic // Вестник иркутского государственного технического университета. – 2020. Том: 24(5):1080–1092.

129. Egorov S. Выбор языка имитационного моделирования, или не заколачивайте гвозди микроскопом [Электронный ресурс] URL: https://www.anylogic.ru/blog/vybor-yazyka-imitatsionnogo-modelirovaniya-ili-ne-zakolachivayte-gvozdi-mikroskopom/. (дата обращения: 02.09.2024).

ПРИЛОЖЕНИЕ А

Сравнение алгоритмов живой миграции

Таблица А.1 Сравнение гипервизоров Xen и KVM

Базовый метод миграции	Методика	Используемый гипервизор	Показатели	Достижения/выгода	Ссылка на источник
Pre-copy algorithm	Общее хранение	Xen	- время миграции	Сокращение на 32% общего времени миграции;	DOI:10.1109/CUBE.2013.14 S. Sharma Meenu Chawla 2013; DOI:10.1145/2451512.2451524 Changyeon Jo Erik Gustafsson Jeongseok Son Bernhard Egger 2013
			- время простоя	Сокращение на 37% времени простоя	
			- сетевой трафик		
			- количество переданных страниц памяти	Количество переданных страниц практически одинаковое;	
	Адаптивное сжатие памяти	Xen	- время миграции	Сокращение на 32% общего времени миграции;	DOI:10.1109/CLUSTR.2009.5289170 Hai Jin Li Deng Song Wu 2009
			- время простоя	Сокращение на 27.1% времени простоя.	
			- сетевой трафик		
			- количество переданных страниц памяти	Сокращение количества переданных страниц на 68.8%	
Post-copy algorithm	Адаптивная предварительная подкачка	Xen	количество переданных страниц памяти	Устранены все дублирующиеся передачи страниц	DOI:10.1145/1508293.1508301 Michael Hines

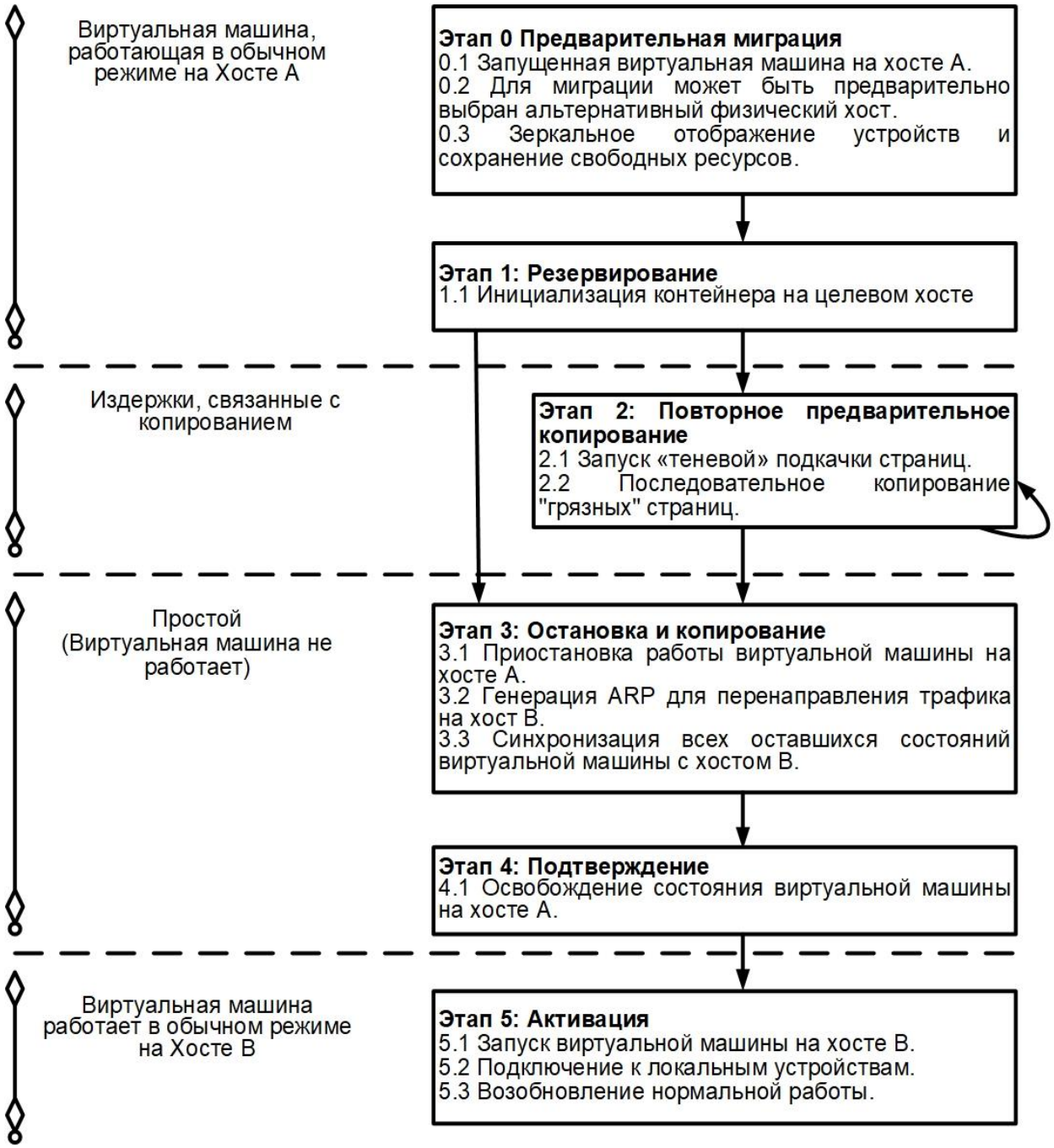
Базовый метод миграции	Методика	Используемый гипервизор	Показатели	Достижения/выгода	Ссылка на источник
					Kartik Gopalan 2009
				Сокращение количества сбоев передачи страниц, связанных с сетью, до 21%	Corpus ID: 15175482 Diego Perez-Botero 2011; DOI:10.1145/1618525.1618528 Michael Hines Umesh Deshpande Kartik Gopalan 2009
Pre-copy algorithm	Устранение дублирования данных	Xen	- время миграции	Сокращение на 56,60% общего объема данных, передаваемых во время миграции	DOI:10.1109/CLUSTER.2010.17 Xiang Zhang Zhigang Huo Jie Ma Dan Meng 2010
			- время простоя	Сокращение на 34,93% общего времени миграции	
Другие методы живой миграции виртуальных машин					
Live migration Pre-copy	Энергосберегающий алгоритм виртуальной машины	Специально созданный симулятор	Потребляемая мощность	Уменьшение энергопотребления на 28% при статической нагрузке и на 22% при динамической	DOI:10.1109/ICPADS.2009.20 Wenchao Cui Dian-fu Ma Tianyu Wo Qin Li 2009

Базовый метод миграции	Методика	Используемый гипервизор	Показатели	Достижения/выгода	Ссылка на источник
	Постоянная миграция	KVM	Отказоустойчивость	Аппаратный сбой (восстановление менее чем за одну секунду)	DOI:10.1109/TPDS.2011.86 Haikun Liu Hai Jin Xiaofei Liao Chen Yu 2012
				Сокращение времени простоя, общая производительность составляет 30%	
	Асинхронная репликация и синхронизация состояний	UMLinux/ Инструмент для записи в журнал и воспроизведения	- время простоя	Сокращение времени простоя на 72,4%	
			- время миграции	Сокращение времени миграции на 31,4%	
			-сетевой трафик	Пропускная способность сети составляет 95,9%	
			- отказоустойчивость		
	Групповая миграция	QEMU/KVM	- время миграции	Сокращение времени миграции на 42%	
			-сетевой трафик	Сокращение времени простоя на 65%	

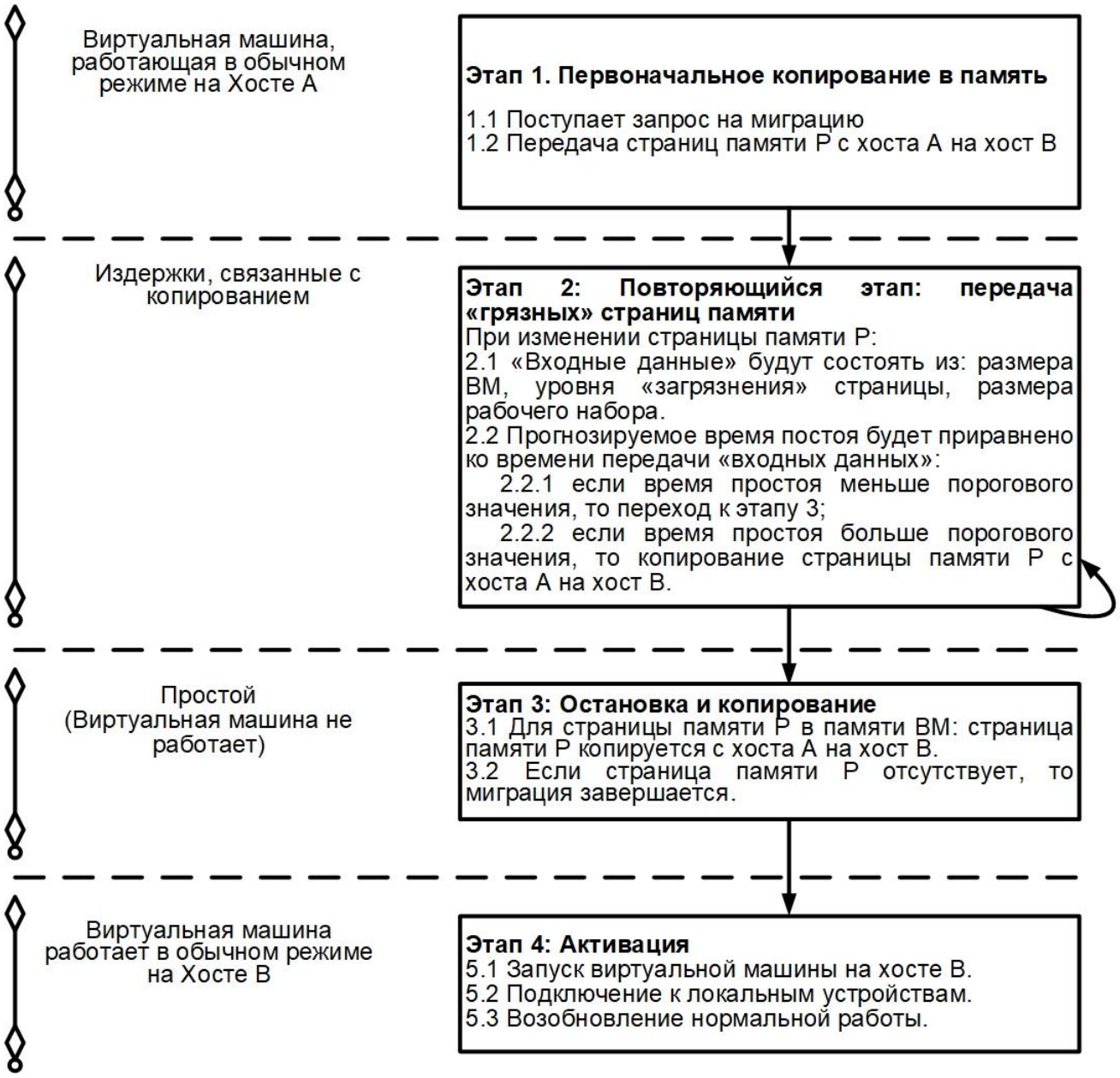
Таблица А.2 Сравнение гипервизоров Xen, KVM и VMware

Тип гипервизора	Особенность миграции	Метод миграции	Системные требования	Конфигурация процесса	Тип хранения	Время простоя
Xen	XenMotion	Pre-copy	Две физические системы в пуле ресурсов	Две физические системы в пуле ресурсов	Общее хранение	1 или более секунд
KVM	KVM Migration	Pre-copy	QEMU-KVM	Возможна миграция с хоста AMD на хост Intel и обратно	Общее хранение	Стремится к нулю
VMware	VMotion	Pre-copy	ESXi 5.1 или более поздняя версия	Совместимость с процессором	Общее хранение	Равно нулю

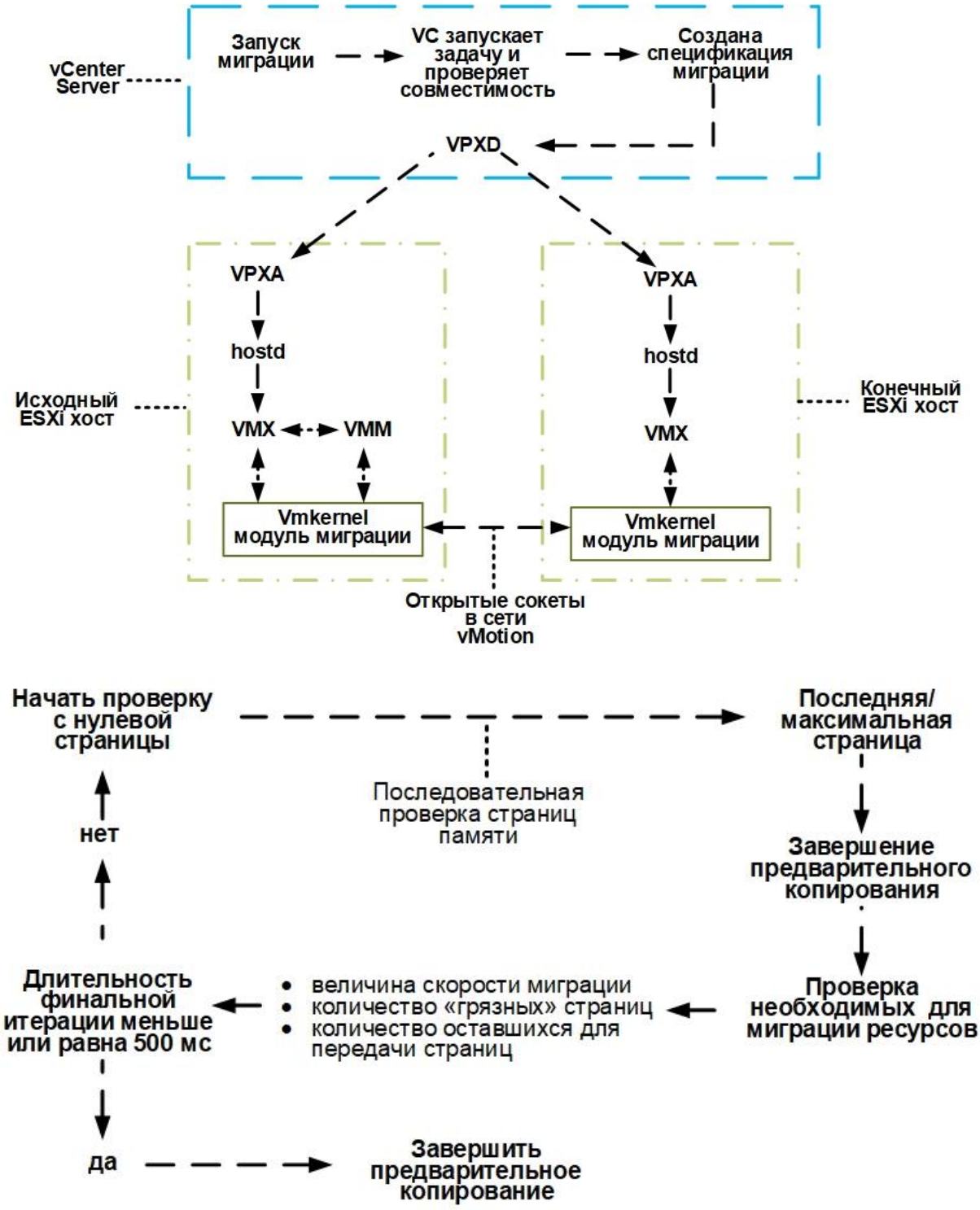
а) алгоритм живой миграции гипервизора Xen (XenMotion)



б) Алгоритм живой миграции гипервизора KVM (KVM Migration)

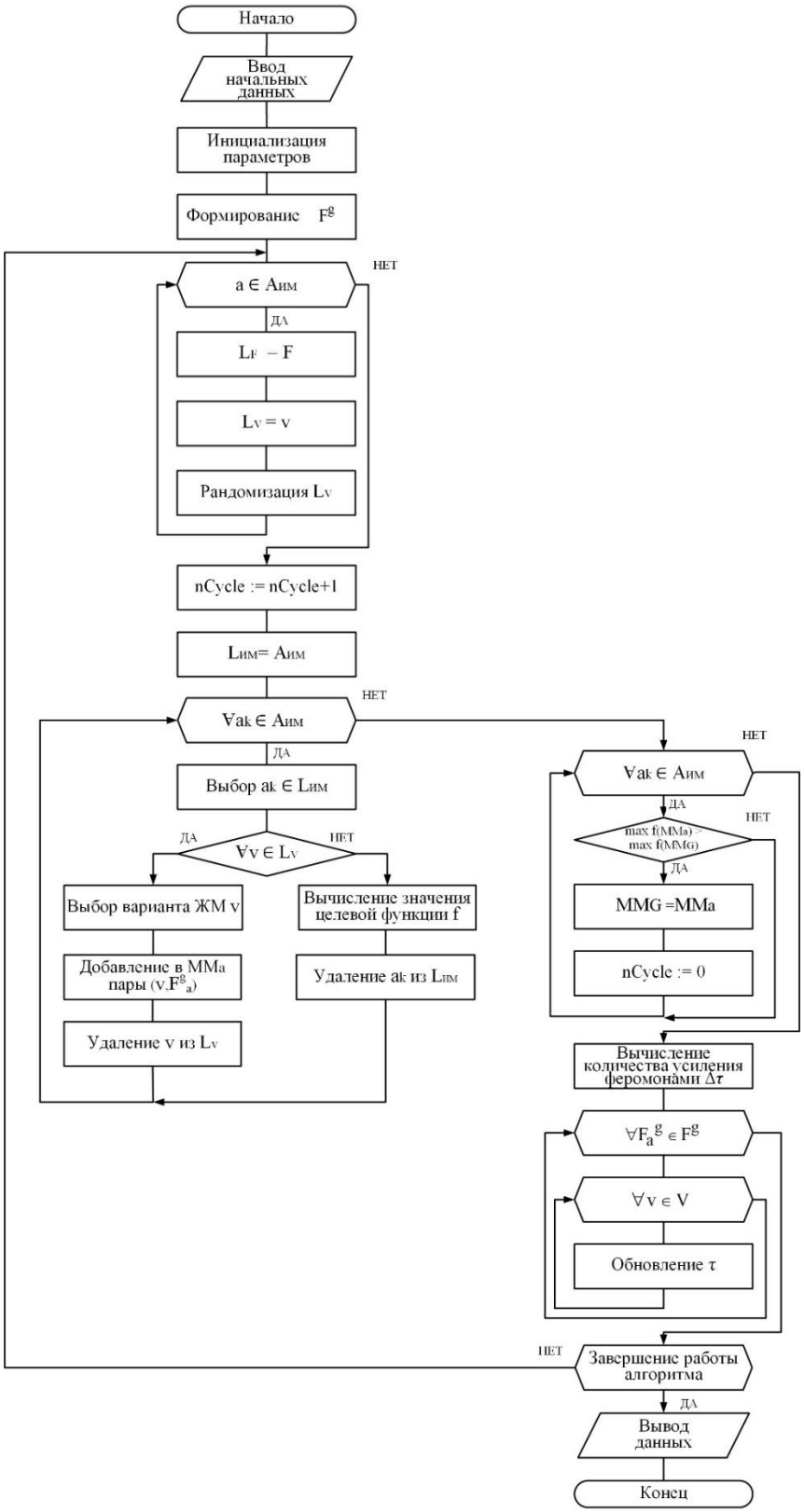


в) Алгоритм живой миграции гипервизора VMware ESXi (VMotion): схема и последовательность действий



ПРИЛОЖЕНИЕ Б

Структурная схема алгоритма переразмещения виртуальных машин гетерогенного центра обработки данных



ПРИЛОЖЕНИЕ В

Свидетельство о регистрации программы для ЭВМ

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2022617593

ПРОГРАММА ОЦЕНКИ ЭФФЕКТИВНОСТИ КОМПЛЕКСНОГО ИСПОЛЬЗОВАНИЯ ИНФОРМАЦИОННЫХ РЕСУРСОВ В СИСТЕМЕ РАСПРЕДЕЛЕННЫХ СИТУАЦИОННЫХ ЦЕНТРОВ

Правообладатель: *Федеральное государственное казенное военное образовательное учреждение высшего образования "Академия Федеральной службы охраны Российской Федерации" (RU)*

Авторы: *Трахинин Егор Леонидович (RU), Белов Андрей Сергеевич (RU), Добрышин Михаил Михайлович (RU), Борзова Наталья Юрьевна (RU), Слесарева Ксения Максимовна (RU)*

Заявка № 2022616258

Дата поступления 12 апреля 2022 г.

Дата государственной регистрации в Реестре программ для ЭВМ *22 апреля 2022 г.*

Руководитель Федеральной службы по интеллектуальной собственности

Ю.С. Зубов

ПРИЛОЖЕНИЕ Г

Акты внедрения

SOVIT SECURITY

ООО «СОВИТ»
Современные информационные технологии

http://www.sovit.net
e-mail: info@sovit.net
Тел.: +7 (495) 120-25-30

ООО «СОВИТ», ИНН 7724780809, КПП 772401001, ОГРН 1117746158342, юридический адрес 115230, г. Москва, Хлебозаводский проезд, д. 7, стр. 9., пом XII ком 6 оф 56Г, +7 (495) 120-25-30, info@sovit.net, р/с 40702810200030002617 в ПАО АКБ «АВАНГАРД», БИК 044525201, к/с 30101810000000000201

Москва **23.09.2024**

АКТ
внедрения результатов диссертационной работы на соискание ученой степени кандидата технических наук Бумажкиной Натальи Юрьевны

Комиссия в составе технического директора ООО «СОВИТ» Кривоноса Алексея Павловича и инженера по сетевой безопасности ООО «СОВИТ» Мичкина Андрея Алексеевича составила настоящий акт о внедрении результатов диссертационного исследования Бумажкиной Н.Ю. по тематике «Математическое и программное обеспечение переразмещения виртуальных машин в виртуализированных центрах обработки данных с гетерогенной структурой» в опытный модуль административного управления IT-инфраструктурой, а именно:

1. Алгоритм переразмещения виртуальных машин виртуализированных ЦОД;
2. Архитектура программно-реализованной системы поддержки переразмещения виртуальных машин в виртуализированных ЦОД.

Результаты диссертационного исследования Бумажкиной Н.Ю. внедрены в состав программного обеспечения административного управления IT-инфраструктурой в виде специального программного модуля.

Члены комиссии:

Кривонос А. П. _____

Мичкин А. А. _____

генеральный директор ООО «СОВИТ» Ноздрина М.В.

«СОВИТ» • МОСКВА •

УТВЕРЖДАЮ

Начальник
Академии ФСО России
кандидат технических наук

С.П. Горелик
«__» октября 2024 г.

АКТ
о внедрении результатов диссертационной работы
Бумажкиной Натальи Юрьевны

Настоящим актом подтверждается, что результаты диссертационного исследования Бумажкиной Н.Ю. по тематике переразмещения виртуальных машин в виртуализированном центре обработки данных с гетерогенной структурой, а именно:

– алгоритм переразмещения виртуальных машин в виртуализированном центре обработки данных с гетерогенной структурой;

– архитектура программно-реализованной системы поддержки переразмещения виртуальных машин в виртуализированных ЦОД с гетерогенной структурой

внедрены в учебный процесс ФГКВОУ ВО «Академия Федеральной службы охраны Российской Федерации» и включены в тематический план учебной дисциплины «Администрирование операционных систем», а именно в практическое занятие 3.9 «Гибридная виртуализация в серверной ОС».

Сотрудник Академии ФСО России
доктор технических наук, доцент
«4» октября 2024 г.

Белов Андрей Сергеевич