Федеральное государственное автономное образовательное учреждение
высшего профессионального образования
«Московский физико-технический институт
(государственный университет)»
Факультет управления и прикладной математики
Кафедра теоретической и прикладной информатики

ДИССЕРТАЦИЯ

на соискание учёной степени

МАГИСТРА

Тема: **Динамическое управление питанием в задаче динамического управления ресурсами: вариации планировщика и эксперименты.**

Направление: 03.04.01 — Прикладные математика и физика

Магистерская программа: 010956 — Математические и информационные технологии

Выполнил студент гр. 073(а) _____Ефанов Н.Н.

Научный руководитель

к. т. н. _____Мелехова А.Л.

Москва — 2016

Содержание:

1.1. Терминология.

DPM (Dynamic Power Management) — динамическое управление

питанием.

ФМ (host) — Физическая машина.

ВМ (Virtual Machine) — виртуальная машина.

ACPI (Advanced Configuration and Power Interface) — усовершенствованный интерфейс управления конфигурацией и питанием.

QoS (Quality of service, качество обслуживания)— политика соответствия заданному соглашению о трафике, гарантия соответствия сети или вычислительной системы определённым параметрам надёжности.

SLA(Service Level Agreement, соглашение об уровне услуг)— формализация QoS в форме соглашения между поставщиком и потребителем услуги, регламентирующая качество услуги, обычно в терминах эффективной метрики предмета соглашения: скорости доступа, максимального времени задержки или недоступности сервиса, уровня потери пакетов и др.

SLAV(SLA Violation, нарушение SLA) — состояние системы, в котором условия SLA не выполняются.

1.2. Введение.

Вопросы повышения производительности вычислительных систем в настоящее время очень актуальны. В связи с активной миграцией ресурсоёмких приложений в облачные системы, к эффективности последних предъявляются повышенные требования [1]. Наряду с выделением и управлением ресурсами, энергосбережение облачной системы также является важной задачей, ввиду возрастания вычислительных мощностей, приводящего к увеличению расходов на эксплуатацию и техническое обслуживание: современные дата-центры

представляют собой сложные системы из тысяч физических машин (ФМ), выполняющих функции вычисления и хранения информации, маршрутизации и сетевого обмена, а также средств поддержки работы центра: систем питания, охлаждения, управления и контроля. По данным Американского Общества Инженеров Тепловых Систем [2], в 2014 году затраты на инфраструктуру и энергопотребление составляют порядка 75% общей стоимости поддержки дата-центра[3]. При этом, данные, собранные с порядка 5000 ФМ в различных дата-центрах за 6-месячный период говорят о достаточно редких ситуациях активной загруженности всех ФМ и загруженности на уровне 100 % [4], что говорит о возможности выключения незагруженных ФМ в зависимости от суммарных нагрузок, а также о возможности консолидации нагрузок с целью высвобождения некоторого числа ФМ, в сумме с балансировкой нагрузки на активных ФМ с целью поддержки производительности (минимизация времени отклика, качество обслуживания). Следовательно , возникает проблема динамического планирования ресурсов компьютерной системы [5] , энергопотребление которой также является компонентом и показателем эффективности решения: следует обеспечить максимально возможную производительность, минимизируя при этом энергозатраты. При этом, ввиду нестационарности нагрузок во времени, имеет смысл рассматривать динамическое управление системой. Таким образом, переходят к задаче динамического управления питанием (Dynamic Power Management, DPM) [1] в задаче динамического планирования ресурсов (Dynamic Resource Scheduling, DRS)[5] .

DRS, DPM — конкурирующие и в некоторой степени антагонистические , возможны конфликты между консолидацией и балансировкой нагрузок, и важно соблюдать баланс в суммарном решении[1][5].

2.1. Энергетическая модель. Осуществление локального DPM.

2.1.1. Структура энергопотребления. Стандартизация состояний энергопотребления устройств.

В работах [1][6] показано, что энергопотребление платформы можно структурировать, разделив его на 2 части:

- Статическое энергопотребление, включающее затраты на поддержание платформы в рабочем состоянии (например, энергия, потребляемая материнской платой ФМ).
- Динамическое энергопотребление — составляющая, связанная с исполнением приложений. Ввиду сильной зависимости от характеристик нагрузок, использования ресурсов и политики энергопотребления, данная составляющая может сильно варьироваться. Оптимизации именно этой части энергопотребления и посвящена задача локального DPM[1] - динамического переключения устройства в различные состояния энергосбережения, в соответствие с выбранной политикой.

Общий интерфейс управление питанием представлен промышленным стандартом ACPI[7]. Задача данной спецификации — обеспечить взаимодействие между управляющим ПО (операционная система, виртуализационное ПО), аппаратным обеспечением и BIOS материнской платы в области конфигурирования работы устройств и управления питанием. В данном стандарте также описан интерфейс определения аппаратного обеспечения.

В соответствии с ACPI, управление питанием передаётся операционной системе, что позволяет, в отличие от способов управление через BIOS, применять прямы методы детализированного программного управления аппаратным обеспечением. Ряд требований, предъявленный к программному интерфейсу, также должен быть согласован с аппаратной

составляющей. Большинство современных устройств имеет поддержку ACPI на уровне материнской платы и CPU. На уровне ОС ACPI реализуется через размещение в оперативной памяти таблиц описания аппаратных ресурсов и методов управления. Таблицы, содержащие методы управления устройствами и обработчики событий ACPI, содержат код в машинно-независимом наборе инструкций AML (ACPI Machine Language). ОС транслирует AML в инструкции рабочей платформы и передаёт на исполнение.

Набор состояний устройств регламентируется наличием C/D — состояний - состояний функционирования - от C/D0 до C/D3. Состояние C/D0 называется состоянием полной активности. Состояния C/D3 — устройство выключено. C/D1,2 — промежуточные состояния, регламентированные устройством.

Также, состояние C/D0 разбивается на состояния производительности, P-состояния, характеризующиеся различными уровнями напряжения/частоты, описанными в следующем разделе. P0 — состояние максимальной производительности, Pn — максимального энергосбережения.

Примеры работы устройств в различных состояниях (на примере C/P состояний CPU) и связанными с ними мощностями энергопотребления приведёны автором в работе[1].

2.1.2. Динамическое изменение частоты и напряжения (состояния производительности) центрального процессора.

Динамическое изменение напряжения/частоты (Dynamic Voltage/Frequensy Scaling, DVFS) — технология уменьшения энергопотребления устройства в зависимости от загруженности путём снижения тактовой частоты и напряжения питания[8].

Выше указана связь динамической мощности потерь и суммарной мощности затрат на переключение КМОП — микросхемы. Зависимость последней экспериментально выражается как:

$$P \approx C V^2 F \quad \text{, где}$$

C - ёмкость затворов КМОП-транзисторов (условно считаем триггеры гомогенными).

V — напряжение питания.

F — тактовая частота.

Пусть V_f - напряжение формирования инверсионного слоя. Вообще говоря, в случае $V_f \ll V$ наблюдается практически линейная зависимость F от V. При этом, очевидно, уменьшение F приведёт к возрастанию задержки триггера на $t \approx \frac{kV}{V - V_f}$, где k — аппаратная константа (функция ёмкости). Отсюда следуют выводы:

1) оптимальность работы схемы на максимальной рабочей частоте: не имеет смысле уменьшать частоту F при V=const.

2) имеет смысл вести одновременное пропорциональное изменение рабочей частоты и напряжения. Максимальная рабочая частота, ввиду сказанного выше, выражается как

$$F = \frac{1}{L t} \quad \text{, где}$$

L — максимальное количество последовательно соединённых элементов.

На программном уровне, DVFS представляется:

1) сборщиком метрик активности устройства в предыдущие интервалы времени.

2) управляющей программой, осуществляющей предсказание загруженности и оптимальной конфигурации устройства.

3) CPUFreq — драйвером, осуществляющим работу с аппаратным обеспечением.

2.2. Источники и вклад в энергопотребление.

В соответствии с докладом Intel Labs[9], основной вклад в энергопотребление ФМ вносит CPU, а также, в меньшей степени, оперативная память и потери, связанные с эффективностью питания. Однако, ввиду широкого применения DVFS и энергосберегающих техник [10], современные ФМ могут потреблять менее 30% своей пиковой мощности в энергосберегающих состояниях, таким образом, находясь в динамическом диапазоне 70% от пиковой мощности - суммарный вклад CPU в энергопотребление уменьшается. В противоположность этому, динамические диапазоны мощности остальных серверных компонентов становятся уже: менее 50% для оперативной памяти (DRAM), 25% для жестких дисков, 15 % для сетевых коммутаторов и незначительная часть для остальных компонент. Причина заключается в том, что процессор поддерживает активное энергосбережение, в то время как остальные компоненты могут быть только полностью или частично отключены. В то же время, затраты производительности на переходы между активными и неактивными состояниями являются существенными. К примеру, жесткий диск в режиме глубокого сна почти не расходует мощности, но переход в активное состояние создаёт задержку, превышающую в 1000 раз задержку доступа в активном состоянии. Неэффективность элементов сервера на таких переключениях приводит к узким динамическим диапазонам общей мощности: если сервер полностью свободен, он потребляет более 70% пиковой мощности. В текущей работе ряд элементов системы исключён из задачи DPM, а их энергопотребление считается статическим: хранилище информации (образов VM, конфигураций и прочих данных) подключается отдельно, сетевые маршрутизаторы и карты ФМ работают без динамической оптимизации методом «sleep»-«doze»[11] и т. д. С другой

стороны , такое построение обеспечивает минимизацию времени живой миграции ВМ в ходе последней, о чём сказано в разделе 3.

Итак, задача локального DPM фокусируется, в первую очередь, на динамическом энергопотреблении CPU: либо профилированием через считывание соответствующих модель-специфичных RAPL-счетчиков [12], либо соответствии с моделью энергопотребления по загруженности определяется оптимальное состояние.

2.2.1. Модель динамического энергопотребления CPU.

Для разработки эффективных политик энергосбережения и понимания вклада последнего в общую эффективность системы, важно создать адекватную модель динамического энергопотребления. На основании такой модели можно будет получать актуальные значения энергопотребления непосредственно во время работы. Как указано выше, одним из путей решения является использование встроенных счетчиков для мониторинга энергопотребления и сбора статистики. Отталкиваясь от полученных данных, можно определить модель. Тем не менее, данное решение требует сбора данных от каждой ФМ.

Авторы работы [13] определили строгое отношение между загруженностью CPU и общим энергопотреблением. Их идея заключается в том, что энергопотребление ФМ растёт линейно вместе с ростом загруженности CPU от уровня энергопотребления в состоянии простоя до уровня при полной загруженности:

$$P = P_{idle} + (P_{busy} - P_{idle})u \quad \text{, где}$$

P - искомое значение, P_{idle} - потребление простоя, P_{busy} - потребление при полной загруженности, u — загруженность. Авторы также предлагают экспериментальную нелинейную модель:

$$P = P_{idle} + (P_{busy} - P_{idle})(2u - u^r) \quad \text{, где}$$

r — экспериментальная калибровочная постоянная, большинстве случаев близкая к 1. В соответствии с исследованиями авторов на большом наборе различных нагрузок, линейная модель даёт ошибку менее 5%, а нелинеi

результаты объясняются фактом основополагающего потребления CPU, существенно более узкими диапазонами энергопотребления остальных компонент ФМ, а также скоррелированностью активности компонент с нагрузкой CPU.

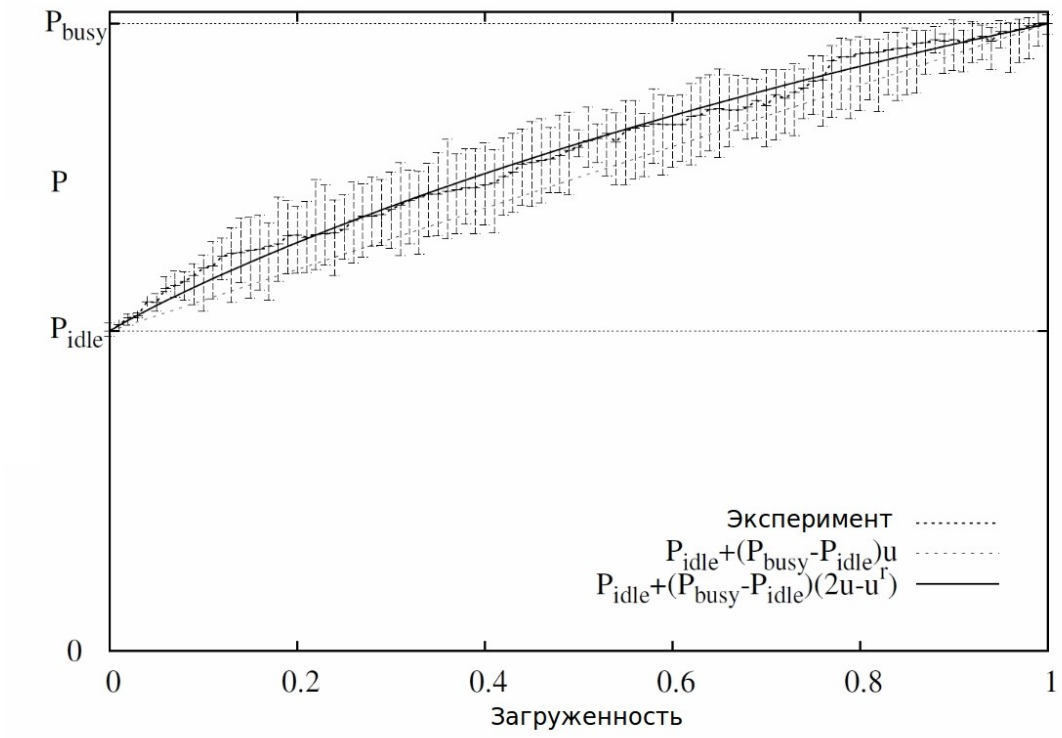

Рис. 1. Мощность энергопотребления в зависимости от загруженности[13].

В данной работе автор использует представленную выше линейную модель, а также результаты экспериментальной калибровки аппаратного обеспечения с помощью SPECpower[14]. Также представляет интерес работа [15], в которой авторы предлагают методы предсказания нагрузок на

основании регрессионных моделей. Также автором была разработана система сбора метрик энергопотребления через модель-зависимые RAPL-счетчики, доступная на GitHub [16].

3. Построение DRS-DPM.

3.1. Виртуализация: «за и против».

Широкие возможности гибкого использования вычислительных систем предоставляет виртуализация [1][4][17] . Выделение и изоляция вычислительных ресурсов, окружений, инфраструктуры и сервисов, обеспечение отказоустойчивости и масштабирования — основные функции облачной системы — реализуются благодаря виртуализации физических ресурсов, либо ядра операционных систем в случае контейнеров [18]. Ресурсом задачи DRS-DPM в описании такой системы является виртуальная машина (ВМ), потребляющая некоторые физические ресурсы (CPU, RAM, Network bandwidth) ФМ.

Наряду с получением колоссальных преимуществ виртуализации, разработчики политик и алгоритмов DRS-DPM встречают ряд ограничений:

- трудности учёта и построения моделей затрат на живую миграцию ВМ между ФМ [1].
- особенности конфигурации конкретных облачных инфраструктур (к примеру, жесткая привязка ВМ к ФМ или существование специфичных подгрупп в группах ВМ и ФМ).
- прогнозирование энергии, потребляемой ФМ, требует построения и внедрения средств профилирования производительности и энергопотребления, зачастую вносящих высокие накладные расходы[1].
- вариативность нагрузки в зависимости от загруженности ВМ[1] —

различное программное обеспечение на ВМ выдаёт нагрузки неоднородного характера [1][4], что усложняет модель системы. Требуется, в первую очередь, ввести эффективные метрики, независимые от характера нагрузок [4].

- высокую сложность общей задачи оптимизации, которая без конкретизации некоторых свойств системы и ограничений по нагрузкам является NP-трудной задачей.

Автор и др. в обзорной работе [1] разделяет задачу DPM на локальную, решаемую на ФМ, и глобальную, решаемую на уровне всей системы.

Авторы работы [4] называют задачу DRS-DPM задачей динамической энергоэффективной консолидации ВМ.

3.2. Подзадачи DRS-DPM.

Для практического решения задачи и обеспечения децентрализации, DRS-DPM разбивается на несколько подзадач:

1. Определение перегруженности ФМ: локальный планировщик определяет потенциальную перегруженность ФМ с целью недопущения SLAV. По достижению перегруженности некоторые ВМ должны мигрировать с текущей ФМ.

2. Определение недогруженности ФМ: локальный планировщик устанавливает нижнее значение нагрузки на ФМ, по достижении которой все ВМ должны мигрировать с текущей ФМ, с последующим переводом последней в энергосберегающее состояние или выключение в ходе работы локального DPM.

3. Выбор ВМ для миграции: определяются ВМ, которые должны мигрировать.

4. Выбор нового расположения ВМ, отобранных в предыдущей подзазаче. Далее происходит осуществление живых миграций.

Ряд предлагаемых ранее решений [1][4][5] обуславливал тесное взаимодействие глобального и локального планировщиков в подзадачах: глобальный планировщик обязан контролировать весь диапазон ФМ, над которым осуществляется оптимизация, а также полное расположение ВМ на ФМ, в том числе и осуществлять выбор ВМ для миграций[1].

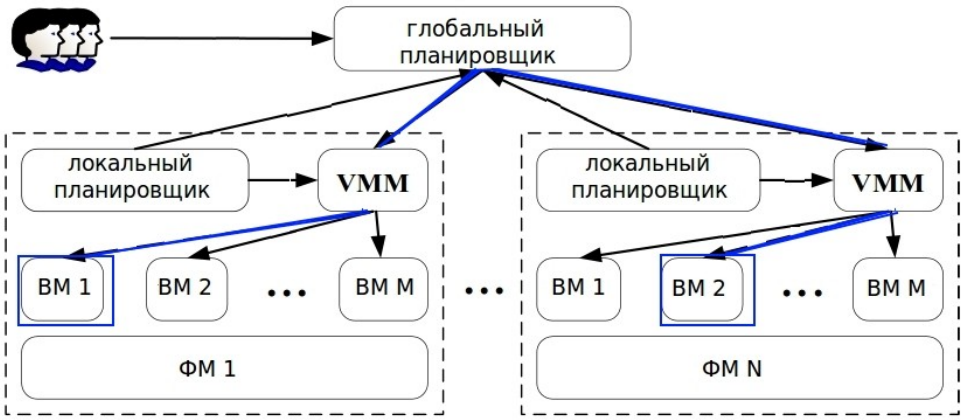

Рис. 2. Архитектура системы и управление миграциями через гипервизор (VMM). ВМ 1 выбрана к миграции с ФМ 1 и ВМ 2 выбрана к миграции с ФМ 2 в ходе работы локального планировщика. Передача списков ВМ глобальному планирощику с целью поиска новых расположений.

С целью обеспечения распределённости и упрощения анализа, подзадачи 1-3 данной работы решаются средствами локального планировщика ФМ. В ходе решения 4, локальный планировщик взаимодействует с глобальным, предоставляя последнему список мигрируемых ВМ. Общая диаграмма работы локального планировщика в подзадачах 1-3 приведена на рис. 3.

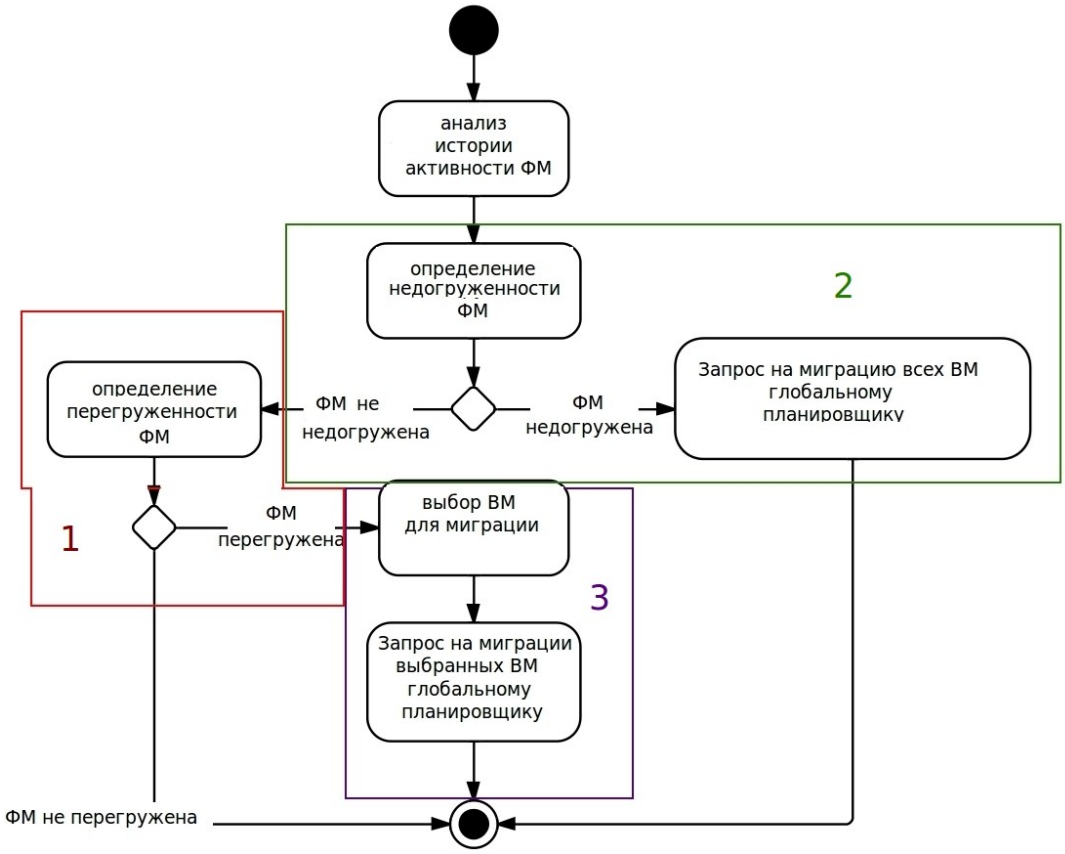

Рис. 3. Диаграмма работы локального планировщика в подзадачах 1-3.

1 — задача определения перегруженности ФМ.

2 — задача определения недогруженности ФМ.

3 — задача выбора ВМ для миграции.

Таким образом, глобально контролируется лишь динамически изменяющееся подмножество мигрирующих ВМ, а не всё расположение, что упрощает решение (рис. 2).

После осуществления миграций и перевода системы в новое состояние, локальный DPM на ФМ координирует уровень энергопотребления с новой нагрузкой, переводит процессоры ФМ в различные C, P, T — состояния.

Таким образом, новое состояние системы представляет собой минимально возможное число активных ФМ, благодаря увеличению плотности упаковки ВМ, и максимально возможной поддержки производительности и соответствия QoS.

3.3. Определение перегруженности ФМ.

Подзадача определения перегруженности ФМ является важнейшей в обеспечении соответствия QoS. Данный подраздел описывает ряд подходов к решению данной проблемы, описывает важные свойства решения относительно миграций ВМ, а также даёт теоретическое обоснование эффективности онлайн-алгоритмов решения.

Решения, приводимые в работе, соответствуют составляющей локального планировщика, диаграмма работы и взаимодействия с глобальным планировщиком которого приведена на рис. 3.3.

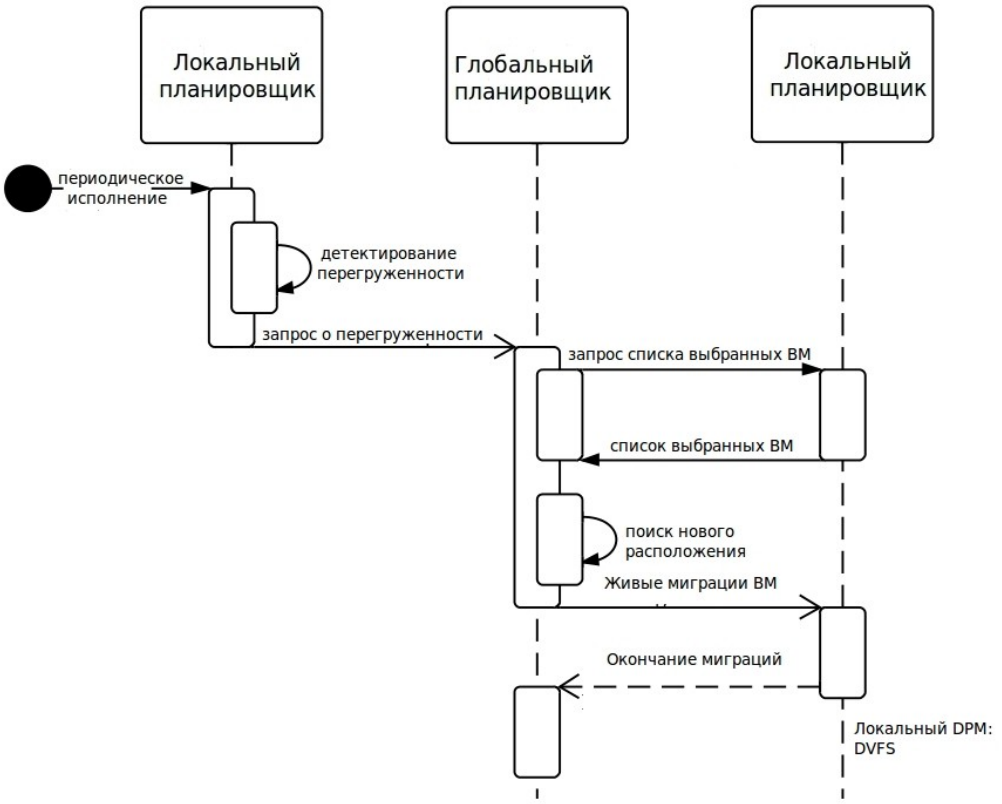

Рис. 4. Диаграмма обработки перегруженности ФМ.

3.3.1. Нестационарные нагрузки: поиск оптимальных онлайн-алгоритмов.

В реальных компьютерных системах алгоритм управления не имеет полных данных о поведении подконтрольных компонент, ровно как и о будущей нагрузке и событиях. Таким образом, мы сталкиваемся с онлайн-проблемой — задачей, вход и выход которой связываются различными процессами в реальном времени. Один из способов поиска онлайн-алгоритмов решения таких задач — соревновательная теория алгоритмов [19]. Сначала находится оптимальный офлайн-алгоритм, затем строится онлайн-алгоритм, в какой-то степени «конкурирующий» с ним по эффективности.

Определение. Алгоритм A называется c-соревновательным оптимальному офлайн-алгоритму OPT , если $\forall I \exists a, c : A(I) \leqslant c\, OPT(I) + a$, I — конечная входная последовательность событий.

Приведём формулировки некоторых задач, возникающих при решении DRS-DPM для виртуализованной системы, и построения онлайн алгоритмов их решения.

3.3.1.1. Оптимальный онлайн-алгоритм: задача Миграции единичной ВМ.

Данный подраздел посвящён соревновательному анализу подзадачи энергоэффективной и производительной консолидации ВМ. Пусть имеется ФМ и M ВМ, размещённых на ней. Время дискретно и разделено на K-кадров, где каждый кадр длится 1 секунду. Поставщик ресурсов расходует C ресурсов на энергозатраты ФМ: $C_p t_p$, где C_p - стоимость энергозатрат в единицу времени, t_p — период. Ресурсы ФМ и использование ресурсов ВМ характеризуются в терминах производительности CPU.

ВМ исполняет динамические нагрузки, следовательно, нагрузка CPU

также варьируется динамически. В случае, если ФМ предоставляет недостаточно ресурсов CPU относительно текущего спроса всех ВМ, случается SLAV между поставщиком и пользователем. SLAV регламентируется затратами поставщика: $C_v t_v$ где C_v - стоимость SLAV-затрат в единицу времени, t_v — период SLAV. Не теряя общности, $C_p = 1$ $C_v = s$ где $s \in R$.

Итак, пусть в момент времени v происходит SLAV и продолжается до K. По определению подзадачи перегруженности ФМ, ВМ должна мигрировать с ФМ. Пусть время миграции равно T. В ходе миграции новая ФМ принимает мигрирующую ВМ, и общие затраты в этом процессе составляют $2C_p T$. Введём также вспомогательную переменную r — время, прошедшее с начала SLAV.

Задача: найти момент времени m, в который нужно инициировать миграцию с целью минимизации общей функции стоимости, имеющей в данной постановке вид:

$$C(v,m) = \begin{cases} (v-m)C_p, & m < v,\ v-m \geq T \\ (v-m)C_p + (r-m+v)C_p + (m-v+T)C_v, & m \leq v,\ v-m < T \\ rC_p + (r-m_v)C_p + rC_v, & m > v \end{cases}$$

Далее, на основании теории[19] , формулируются теоремы (доказательства приведены в работе[20]):

Теорема 1. Оптимальный офлайн-алгоритм для задачи миграции одиночной ВМ даёт затраты $\frac{T}{s}$ и достигается при $\frac{v-m}{T} = 1$.

На основании соревновательного анализа формулируется и доказывается теорема 2.

Теорема 2. Соревновательная точность оптимального онлайн-алгоритма равна $2+s$ и достигается при $m = v$.

3.3.1.2. Оптимальный онлайн-алгоритм: динамическая консолидация ВМ.

В данном подразделе задача 3.3.1.1 обобщается на случай n ФМ ёмкостью A_h каждая и максимальными требованиями ВМ к ресурсам является величина A_v , таким образом, максимальная плотность ВМ на ФМ составляет $m=\frac{A_h}{A_v}$ ВМ, и mn -максимальное число ВМ в гомоменной системе, использующейся для построения самого общего доказательства (гетерогенная система расширяется «приведением» к гомогенной с ёмкостью, равной ёмкости максимально ёмкой ФМ, и увеличением требований ВМ до требований максимально требовательной ВМ). ФМ динамически выключаются и включаются по требованию. Аналогично 3.3.1.1, вводится функция стоимости

$$C=\sum_{t=0}^{T}\left(C_p\sum_{i=0}^{n}a_{ti}+C_v\sum_{j=0}^{n}v_{tj}\right) ,$$

где a_{ti} — индикаторная функция активности ФМ i в момент t, v_{tj} — индикаторная функция SLAV на ФМ j . В соответствии с оптимизацией которой и строится оптимальный офлайн-алгоритм.

Теорема 3. Верхняя граница соревновательного соотношения онлайн алгоритма для динамической консолидации ВМ для любого входа I составляет $\frac{ALG(I)}{OPT(I)}\leqslant 1+\frac{ms}{2(m+1)}$, где OPT — оптимальный офлайн-алгоритм. Доказательство приведено в работе [20].

С одной стороны, введённая модель доказывает эффективность построения взаимосвязанных онлайн-решений подзадач 1 и 3, с другой - вводит дополнительное ограничение на эффективность. Следовательно, для различных типов онлайн-алгоритмов, описанных в следующей главе, теоретически можно ожидать соревновательное соотношение на уровне

границы из теоремы 3.

3.3.2. Эвристические алгоритмы.

В данном подразделе представлен ряд эвристик для подзадачи определения перегруженности ФМ, основанных на модели системы, описанной выше. Данные эвристики предоставляют устойчивое к нестационарным нагрузкам решение, которое значительно сокращает энергопотребление всей системы, одновременно обеспечивая высокий уровень поддержки SLA. Высокая эффективность выбранных методов продемонстрирована экстенсивной симуляцией на данных, собранных с более тысячи ВМ, описанной в главе 5.

3.3.2.1. Алгоритм статической верхней границы.

Простейшим подходом к определению перегруженности ФМ является определение статического значения загруженности CPU , по достижении которого ФМ считается перегруженной. Тем не менее, методы, выставляющие жестко фиксированные значения границ плохо применимы в случае динамических и плохо предсказуемых нагрузок, которые могут возникать в ходе работ разнородных приложений на ВМ, что вынуждает исследователей и разработчиков искать методы адаптивного изменения границ.

3.3.2.2. Адаптивная верхняя граница: абсолютное медианное отклонение.

Представим эвристический алгоритм адаптивного изменения верхней границы на основании робастного статистического анализа истории работы

ВМ. Выбор робастных методов осуществляется по причине лучшей теоретической эффективности работы с данными, содержащими большое число выбросов и порожденными распределениями, не являющимися нормальными.

Робастные статистики предоставляют альтернативу классическим методам, существенно теряющим точность предсказания при малых отклонениях от модели.

Предлагается выставлять границу в зависимости от величины отклонения загрузки CPU: чем выше отклонения, тем ниже граница. Обоснование метода лежит в связи роста отклонения с приближением загрузки к 100%, потенциально приводящим к SLAV. Используется MAD-робастная статистика, оперирующая медианой абсолютного отклонения от медианы выборки:

$$MAD = median_i\left|\left(X_i - median_j\left(X_j\right)\right)\right|$$

Граница задаётся как $T_u = 1 - s\,MAD$, где s — параметр жесткости: меньшие значения s приводят к большему «безразличию» к вариациям заргузки CPU и увеличивают уровень SLAV при консолидации ВМ. Зависимость s от значения SLAV-метрик определяется экспериментально. Таким образом, система должна быть предварительно откалибрована для соответствия некоторому QoS, прямое управление параметрами которого в терминах SLAV-метрик метод не предоставляет.

3.3.2.3. Адаптивная верхняя граница: межквартильный размах.

Метод основан на анализе размаха между 1 и 3 квартилями выборки:

$$MR = Q_3 - Q_1 \ ,$$

позволяющего оценить разброс 50% элементов без учёта влияния экстремальных элементов. При вычислении Q_1 и Q_3 не учитывается влияние элементов, меньших Q_1 и больших Q_3 , следовательно, метод

безразличен к наличию выбросов и является робастным. Граница задаётся как $T_u = 1 - s\,MR$, где s — параметр жесткости, обладающий теми же свойствами, что и параметр раздела 3.3.2.

3.3.2.4. Локальная регрессия Кливланда.

Регрессия Кливланда[22] представляет собой робастное улучшение локальной регрессии Лоесса[21], превращающее данный метод в итерационный метод с «размытием» выбросов по невязке. В качестве весовой функции выбирается трикубическая[21]:

$$T(u) = \begin{cases} \left(1 - |u|^3\right)^3 & ,|u| < 1 \\ 0 & ,|u| \geq 1 \end{cases},$$

и вес соседей в рассчитывается по формуле

$$w_i(x) = T\left(\frac{d_i(x)}{d_{(q)}(x)}\right),$$

где $d_i(x)$ — расстояние от x до x_i, $d_{(q)}(x)$ — расстояние между двумя максимально удалёнными элементами из всех q элементов данной выборки.

Производится поиск функции из параметрического семейства $a+bx$, где коэффициенты a, b методом наименьших квадратов:

$$\sum_{i=1}^{n} w_i(x)\left(y_i - a - bx_i\right)^2 \to Min$$

Применим данный поиск к $k = \lceil q/2 \rceil$ последним наблюдениям CPU. Пусть x_k — последнее наблюдение, x_1 — k-е наблюдение от правой границы. Пусть нагрузка возрастает монотонно: $x_1 \leq x_i \leq x_k$, $d_t(x_k) = x_k - x_i$, так что значение аргумента T строго меньше 1 по модулю. Поэтому весовые коэффициенты имеют вид:

$$w_i(x) = \left(1 - \left(\frac{x_k - x_i}{x_k - x_1}\right)^3\right)^3$$

В немодифицированном методе для каждого нового наблюдения находится новая линия тренда $\hat{g}(x)=\hat{a}+\hat{b}x$, используемая для оценки нового наблюдения $\hat{g}(x_{k+1})$. ФМ считается перегруженной, если выполнены следующие условия:

$$s\hat{g}(x_{k+1})\geq 1; x_{k+1}-x_k\geq t_m \ ,$$

где s — параметр жесткости, t_m — максимальное время миграции ВМ из расположенных на данной ФМ.

Немодифицированный метод достаточно эффективен и широко распространён, однако плохо применим в при наличии выбросов, обусловленных остроконечными или «тяжелохвостными» распределениями[22].

Представим модифицированный метод, ликвидирующий вышеописанную уязвимость. Пусть задано начальное приближение в точке x_i , равное $\hat{y}_i$, так что невязка $\hat{e}_i=y_i-\hat{y}_i$. На следующем шаге каждому наблюдению (x_i,y_i) ставится в соответствии робастный вес r_i , зависящий от $\hat{e}_i$:

$$r_i=B\left(\frac{\hat{e}_i}{6\,median\widehat{|e_i|}}\right) \ , \text{где}$$

B — биквадратная весовая функция вида:

$$B(u)=\begin{cases}\left(1-|u|^2\right)^2 & ,|u|<1\\ 0 & ,|u|\geq 1\end{cases}$$

После нахождения коэффициентов по «размытым» невязочным весам, получается «робастный тренд», к которому уже применяется оригинальный метод оценки нового измерения.

Трудности применения того или иного регрессионного метода заключаются в компромиссе между реакцией на резкие изменения загрузки CPU и инертностью в случае «размытия» по Кливланду. Экспериментальное сравнение немодифицированного и модифицированного методов приведены в разделе 5.

3.3.2.5. Заключение.

Рассмотренные методы позволяют оперативно анализировать состояние ФМ и принимать решение о необходимости миграции ВМ, однако не позволяют вы явно выставить уровень QoS в терминах той или иной SLAV-метрики: достижение QoS целей обеспечивается калибровкой и подбором параметров алгоритма. В этом и заключается эвристичность всех вышеперечисленных подходов.

3.4. Марковские алгоритмы.

Несмотря на то, что данная работа предлагает вывод определённых зависимостей из теории Марковских цепей именно для подзадачи определения перегруженности ФМ, данный подраздел выделен отдельно. Это связано, во-первых, с более строгим теоретическим подходом и получением некоторых важных свойств, влияющих на задачу DRS-DPM вцелом, а во-вторых, с целью выделить данный класс алгоритмов по построению, на базе которых автор в будущем предполагает строить различные решения по оптимизации в IaaS.

3.4.1. Марковский алгоритм определения перегруженности ФМ.

Момент обнаружения перегруженности ФМ напрямую влияет на QoS, т. к. если объём ресурсов практически полностью используется, вероятно снижение производительности приложений. Сложность обнаружения перегруженности заключается в необходимости оптимизации «усреднённого по времени» поведения системы, обрабатывающей различные гетерогенные нагрузки. При этом эвристические и адаптивные

методы, описанные выше, имеют существенный недостаток с точки зрения управления: их применение приводит , вообще говоря, к нестрого оптимальным результатам, вследствие чего у системного администратора нет возможности явно задать цель QoS. Иными словами, производительность в отношении QoS задаётся неявно вариацией параметров алгоритма.

Подход, предлагаемый в данном разделе, позволяет системному администратору явно указать цель QoS в терминах нагрузконезависимой QoS-метрики. Аналитическая модель, лежащая в его основе, позволяет вывести оптимальную политику управления для любой известной стационарной нагрузки и состояния системы. Обработка нестационарных нагрузок осуществляется конкатенацией стационарных политик.

3.4.2. Вывод предельных зависимостей.

В данном разделе покажем, что эффективный алгоритм консолидации ВМ должен, одновременно с минимизацией числа активных ФМ, максимизировать интервалы между миграциями ВМ. Также представляет интерес вывод предельных зависимостей данных критериев.

Введём H, среднее число активных ФМ за n временных шагов:

$$H=\frac{1}{n}\sum_{i=1}^{n} a_i \quad \text{, где}$$

a_i — число активных ВМ на соответствующем временном шаге $i=1,2,\ldots,n$.«Качество» консолидации ВМ обратно пропорционально H: меньшие значения H соответствуют лучшей консолидации. Введём также H^{add} - среднее число ФМ, включённых дополнительно за эти n шагов для обеспечения расположений мигрирующих ВМ:

$$H^{add}=\frac{1}{n}\sum_{i=1}^{n}\left(a_i-a_1\right) \quad .$$

Физический смысл H^{add} - скорость деконсолидации ВМ по целевому диапазону ФМ в связи с миграциями.

Для исследования влияния решений, принимаемых алгоритмами обнаружения перегруженности, поставим мысленный эксперимент:

Пусть в любой шаг времени алгоритм может инициировать миграцию с целью избежания перегруженности ФМ. Возможны 2 последствия этого решения:

1. Для расположения выбранной ВМ невозможно найти подходящую ФМ, и, в результате, новая ФМ должна быть активирована с целью расположить ВМ.
2. ВМ может быть расположена на активной в данный момент ФМ.

Уточним, что в данном эксперименте, в отличие от реальных ситуаций, ни одна ФМ не может быть выключена, т.е. Если ФМ уже была активна в период

$i=1,2,\ldots,n$, она остаётся активной до n включительно.

Пусть p — вероятность последствия 1. Тогда вероятность последствия 2— (1-p). Пусть T - случайная величина, описывающая время между 2 последовательными миграциями ВМ, инициированными алгоритмом. Ожидаемое число миграций на n временных шагах $\frac{n}{E[T]}$, где $E[T]$ — ожидаемое время между миграциями. На основании вышесказанного, введём

$X \sim B\left(\frac{n}{E[T]}, p\right)$ - биноминально распределённую случайную переменную, описывающую число дополнительно включённых ФМ за n временных шагов. Соответственно X , ожидаемое число дополнительно включённых ФМ $E[X]=\frac{np}{E[T]}$. Пусть A — случайная величина, описывающая время, которое дополнительная ФМ была активна между шагом 1 и n. Тогда

ожидаемое время

$$E[A]=\sum_{i=1}^{\left\lceil \frac{n}{E[T]} \right\rceil}\left(n-(i-1)E[T]\right)p \quad .$$

(3)

Просуммируем (*):

$$E[A]=\left\lceil \frac{n}{E[T]} \right\rceil \frac{p}{2}\left(n+n-\left(\left\lceil \frac{n}{E[T]} \right\rceil -1\right)E[T]\right)\leqslant \frac{np}{2}\left(1+\frac{n}{E[T]}\right) \qquad (4)$$

Запишем H как: $H=\frac{1}{n}\sum_{i=1}^{n}a_i=\frac{1}{n}\sum_{i=1}^{n}a_1+H^{add}$

Оценим ожидаемое значение H^{add} . Очевидно, $E\left[H^{add}\right]$ пропорциональна произведению E[X]E[A]:

$$E\left[H^{add}\right]\propto \frac{1}{n}E[X]E[A]\leqslant \frac{1}{n}\frac{np}{E[T]}\frac{np}{2}\left(1+\frac{n}{E[T]}\right)=\frac{np^2}{2E[T]}\left(1+\frac{n}{E[T]}\right) \quad (5)$$

Ввиду цели улучшения консолидации ВМ, необходимо минимизировать H, следовательно, нужно минимизировать $E\left[H^{add}\right]$. Из (5) видно, что этого можно добиться только максимизацией E[T].

3.4.3. Оптимальный офлайн-алгоритм.

Зависимости, доказанные в 3.4.2, позволяют теоретически получить оптимальный офлайн-алгоритм определения перегруженности, оперирующий «индивидуально» временами между миграциями ВМ. Для начала, ограничим время в пределах от конца одной миграции до конца другой.

Задача может быть сформулирована как оптимизационная:

$$t_a\left(t_m,u_t\right)\rightarrow max$$

$$t_o\frac{\left(t_m,u_t\right)}{t_a\left(t_m,u_t\right)}\leqslant M \quad ,$$

где t_a — время инициирования миграции, u_t — уровень загруженности ФМ, t_o - время, которое ФМ была перегружена, t_a — время

активности ФМ, M — ограничение по OTF-метрике, отвечающая за цель QoS.

В выводе офлайн-алгоритма состояние системы известно в любой момент времени, следовательно, сконцентрируемся на поиске оптимизирующего t_m:

Алгоритм 1:

input: history, M

output: t_m

```
while moment in history:
    if OTF(m) <= M:
        return t_m(m)
    else:
        drop(m)
    m=m->prev;
```

Листинг 1. Оптимальный офлайн-алгоритм определения t_m .

Алгоритм проходит по истории измерений в обратной последовательности, рассчитывая OTF-метрику, сверяя с M , и возвращая искомое время, в случае успеха.

Теорема 1. Алгоритм 1 является оптимальным офлайн-алгоритмом для задачи определения перегруженности ФМ.

Доказательство. Пусть t_m - интервал времени внутри [t_0, t_n], полученный из алгоритма 1. Тогда на полуинтервале (t_m, t_n] нагрузка не соответствовала ограничению по OTF M . Ввиду того, что t_m — правая граница [t_0, t_m], t_m - максимально возможное время, удовлетворяющее ограничению задачи, одновременно с этим максимизирующее t_a , **что и требовалось доказать.**

3.4.4. Марковский алгоритм. Стационарные нагрузки.

Результаты предыдущих глав дают возможность построения оптимального онлайн-алгоритма определения перегруженности ФМ: сначала на базе имеющегося математического аппарата строится офлайн-алгоритм, а далее, применяя соревновательную теорию онлайн-алгоритмов, происходит построение и анализ онлайн-алгоритма.

В качестве математического аппарата будем использовать цепи Маркова, как эффективный механизм работы со случайными процессами. Аппарат марковских цепей исчерпывающе описан в научной литературе [23-27] и широко используется в задачах оптимизации ресурсовыделения и энергопотребления[1].

Для построения Марковского офлайн-алгоритма, работающего с априори известными нагрузками, введём следующие детали в модель ФМ:

1)нагрузки стационарны и соответствуют марковости процесса.

2)загруженность ФМ, измеренная в дискретные моменты времени, соответствует цепи Маркова с дискретным временем.

3)диапазон загруженности CPU ФМ разбивается на N состояний, которые можно описать полуинтервалами: к примеру, состояние 1 отвечает всевозможным уровням загруженности в пределах [0%,10%), состояние 2 — [10%,20%) и т.д.

На основании введённой модели, зная нагрузки на ФМ, можно, применяя метод максимального правдоподобия, определить матрицу переходных вероятностей:

$$p_{ij}=\frac{c_{ij}}{\sum_{k=S} c_{ik}}$$, где

c_{ij} - число переходов между состояниями i и j .

Пространство состояний S, формируемое данными полуинтервалами, расширяется состоянием «миграция ВМ» - A. Результирующее пространство назовём S^{plus} . Состояние «миграция ВМ» - финализирующее: $p_{N+1,N+1}=1$ и соответствует инициированию процесса

миграции. Результирующая, расширенная матрица переходов имеет вид:

$$\begin{pmatrix} p_{11}^{norm} & \dots & m_1 \\ \dots & \dots & \dots \\ 0 & \dots & 1 \end{pmatrix}, \text{ где}$$

$p_{ij}^{norm} = p_{ij}(1 - m_j)$ - нормированные, m_j — вероятность перехода из соответствующего состояния в A. Таким образом, политика управления представляется в виде вероятностей переходов в состояние «миграция ВМ».

Оптимизационная задача в такой форме имеет вид:

$$\sum_{i \in S} L_i \to max$$

$$\frac{T_m + L_N}{T_m + \sum_{i \in S} L_i} \le M \quad ,$$

где L_N — ожидаемое время до перехода в состояние A. T_m - время миграции, M — ограничение по метрике. Решение данной задачи — вектор $(m_{1,}\dots, m_N)$: ВМ мигрирует с вероятностью m_i , где i - текущее состояние. Политика детерминистична, если $\forall i \in 1..N \exists k \in 1..N : m_k = 1 \to m_i = 0$. Офлайн-алгоритм позволяет препросчитать $(m_{1,}\dots, m_N)$ для различных типов нагрузок, а затем пользоваться результатами просчёта.

3.4.5. Марковский алгоритм. Нестационарные нагрузки.

Оптимальный офлайн-алгоритм 3.4.5, очевидно, не подходит для использования в условиях нестационарных нагрузок, которые возникают в реальной IaaS-системе. Однако данный метод позволяет вывести на его основании алгоритм, эффективно обрабатывающий нестационарные нагрузки средствами мультиразмерного скользящего окна [28].

Соревновательность метода показана экспериментально в работе [20].

3.4.6. Марковский алгоритм. Заключение.

Приведённые в 3.4.3-3.4.6 рассуждения предлагают базис для построения эффективного оптимального онлайн-алгоритма определения перегруженности ФМ с явным заданием ограничения по какой-либо SLAV-метрике. Данный алгоритм демонстрирует[28] хорошие результаты адаптации к нестационарным нагрузкам и снижает суммарные затраты на миграции по построению.

Реализация и интеграция такого алгоритма в общей задаче DRS-DPM — вопрос дальнейших исследований и работ автора.

3.5. Определение недогруженности ФМ.

Существуют достаточно сложные стратегии для подзадачи определения недогруженности ФМ, однако на практике предлагается использовать простое решение, ввиду того, что точное предсказание недогруженности менее важно, по сравнению с подзадачей 3.4. Процесс обнаружения недогруженности осуществляется следующим образом.

1)ФМ, уровни загруженности которых существенно ниже остальных по какому-либо критерию(среднее, медиана и др.), предлагают глобальному планировщику все свои ВМ для поиска нового расположения. 2)если такое расположение найдено, осуществляются миграции, а ФМ переводятся в энергосберегающие состояния или отключаются. В противном случае ФМ остаётся активной.

1*)можно также предлагать ВМ для миграции по достижения некоторой статической нижней границы загруженности, однако это может заметно увеличить число миграций.

Применение 1 или 1* является компромисс-движимым: в приоритете к жесткому снижению энергопотребления возможно использование 1*,

однако предлагается использовать 1 с целью поддержания высокого уровня QoS.

Диаграмма обработки ситуации недогруженности локальным планировщиком и взаимодействия с глобальным планировщиком приведена на рис. 5.

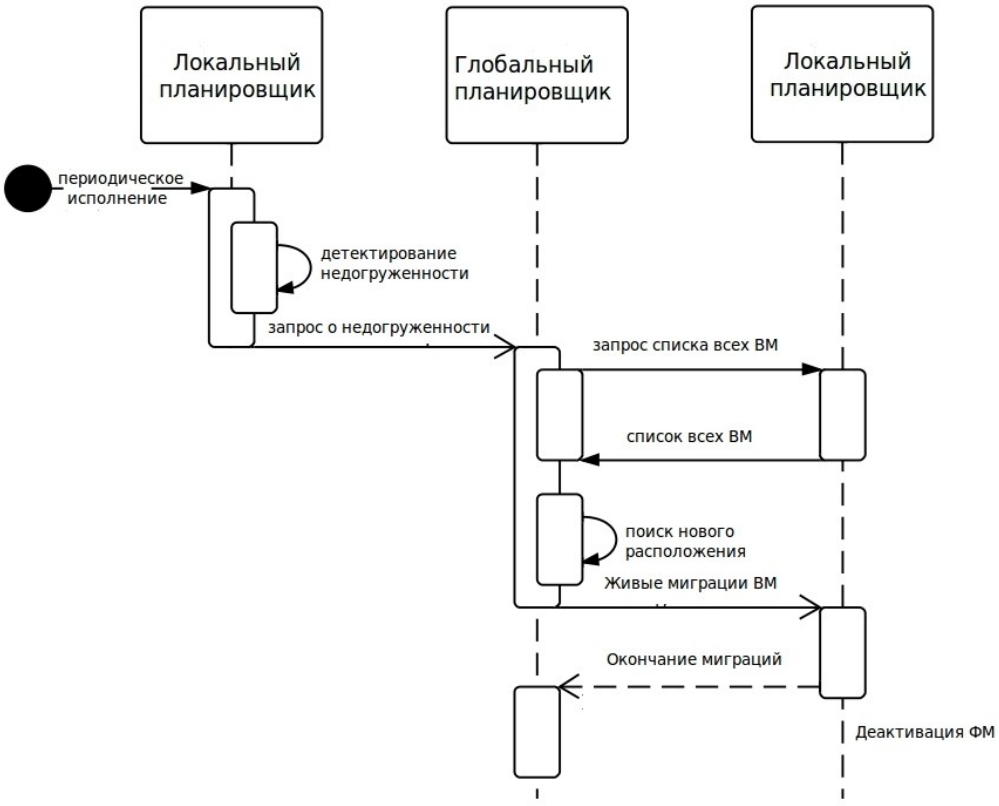

Рис. 5. Диаграмма обработки ситуации недогруженности ФМ.

3.6. Выбор ВМ.

В случае перегруженности ФМ, следующим шагом является выбор локальным планировщиком ВМ для разгрузки ФМ путём живых миграций. В исследовательских работах предлагается 3 варианта решения данной

подзадачи.

3.6.1. Выбор ВМ. Случайный выбор.

Данная политика подразумевает выбор ВМ на основании равномерно распределённой дискретной случайной случайной величины $X^d = U(0, |V_j|)$, значения которой индексируют набор ВМ V_j, расположенных на ФМ j.

3.6.2. Выбор ВМ. Минимальное время миграции.

Политика подразумевает миграцию ВМ v, требующую меньшего времени миграции относительно других ВМ.

Живая миграция ВМ — многоэтапный процесс, оценка времени исполнения которого — достаточно сложная задача, описанная в ряде исследований[29-30]. На время живой миграции, в отличие от простой миграции, влияет множество факторов, например, загрязнение страниц памяти, связанных с активностью ВМ в промежутке между инициализацией копирования и даун-таймом. Загрязнённые страницы должны быть синхронизированы и перекопированы заново. Поэтому ряд точных исследований оперирует таким понятием как «Скорость загрязнения страниц».

В симуляции, представленной в разделе 5, ничего не известно об активности ВМ в произвольный момент времени, поэтому используется самая грубая модель, оперирующая временем копирования страниц памяти ВМ $RAM_u(v)$ по сети с пропускной способностью BW_j : для миграции выбирается ВМ из набора V_j, для которой $\forall a \in V_j, \frac{RAM_u(v)}{BW_j} \leq \frac{RAM_u(a)}{BW_j}$. Если таких ВМ несколько, выбирается ВМ, наиболее активно использующая ресурсы CPU в текущий момент или в среднем за

определённый промежуток времени.

Возможна также политика более активного отбора по критерию максимизации использования CPU: в первую очередь, список ВМ сортируется в порядке увеличения требований к RAM. Затем по какому-либо признаку (среднее значение объема RAM, медиана, др.) отбирается поднабор ВМ, в котором выбирается максимально использующая CPU. Такой подход позволяет, теоретически, найти компромисс между эффективностью «снятия» с ФМ существенной доли нагрузки путём миграции относительно легковесной ВМ, и теоретически более эффективен, чем оригинальный метод. Однако на практике могут возникнуть сложности, в первую очередь, с ошибками оценки затрат и времени миграции по простой модели, осложняемые корреляцией высоких нагрузок на CPU с поведением ВМ в ходе живой миграции, в том числе и с необходимостью репликации загрязняемых страниц памяти. Экспериментальное сравнение отбора по такой политике со стандартной приведено в разделе 5.

3.6.3. Выбор ВМ. Максимальная корреляция.

Политика базируется на идее возникновения ситуаций перегруженности : при интенсивном использовании общих ресурсов, либо при выполнении одинаковых задач. Для определения корреляции между использованием CPU ВМ, рассчитывается коэффициент множественной корреляции. Миграции подлежит ВМ, нагрузки которой максимально скоррелированы с остальными ВМ.

3.6.4. Заключение.

Исследование [30] показывает, что наиболее эффективная политика —

политика минимального времени миграции. С другой стороны, данный метод существенно ограничивается выбранной миграционной моделью. Ряд исследований посвящено выводу более точных и оптимальных политик, в том числе и средствами машинного обучения.

3.7. Расположение ВМ.

Подзадача поиска расположения для выбранных ВМ может быть рассмотрена как задача об упаковке[31] с переменными вместимостями и ценами рюкзаков, представляющих ФМ, и объемами предметов, представляющих ВМ.

Размеры рюкзаков соответствуют доступным CPU и RAM-ресурсам ФМ, цены соответствуют энергопотреблению. В общем случае, задача является NP-трудной, и целесообразно применение эвристики, например, BFD(Best-fit decreasing, «Наилучший подходящий по убыванию») - алгоритм, использующий , в соответствии с выводом[31], не более чем $\frac{11}{9}OPT+1$ рюкзаков (где OPT — наименьшее число рюкзаков, найденных полным перебором) .

Input: vmList,hostList

Output: vmPlacement

```
sort(vmList,order=decreasing_utilization)
for VM in vmList:
    minPower = MAX
    allocatedHost = NULL
    for host in hostList:
        if enough_resources(VM,host):
            power = estimatePower(VM,host)
```

```
            if power < minPower:
                allocatedHost = host
                minPower = power
    if allocatedHost != NULL:
        vmPlacement.add(allocatedHost, vm)
return vmPlacement
```

Листинг 2. Алгоритм BFD для подзадачи энергоэффективного расположения ВМ.

Алгоритм сортирует список ВМ в порядке убывания загрузки текущего CPU и назначает новые расположения для каждой ВМ на ФМ, обеспечивающие наименьший прирост энергопотребления. Данное условие позволяет использовать гетерогенность ФМ, выбирая для расположения наиболее энергоэффективные. Сложность алгоритма составляет $O(nm)$, где n — число ФМ и m — число располагаемых ВМ.

4. Метрики эффективности.

Применяя алгоритмы раздела 3, будем задавать цели и оценивать эффективность одновременно по двум типам метрик:

1. QoS-метрики, формально заданные в терминах SLA (SLAV-метрики).
2. Энергосберегающие метрики.

На основании метрик 1 и 2 введём балансную метрику, которая является целевой для данной задачи.

4.1 SLAV-метрики.

Выполнение требований QoS имеет важное значение для поставленной

задачи. Данные требования, как правило, формализованы в виде SLA и определены в терминах минимальной пропускной способности или максимального времени ответа, предоставляемых системой. Так как эти характеристики могут существенно отличаться для различных приложений, необходимо определить нагрузконезависимые метрики, которые могут быть использованы для оценки QoS любой ВМ, развернутой в IaaS.

Данная работа определяет удовлетворение SLA, если 100% запрошенной приложениями на ВМ производительности предоставляется в любое время и регламентируется только параметрами ВМ. Перегруженность влечёт за собой несоответствие SLA. Миграции ВМ также вносят ухудшение производительности. Поэтому требуется рассмотреть 2 метрики:

1. OTF (Overload time fraction, «Доля времени перегрузки») - метрика, описывающая отношение времени 100%-й загруженности ФМ к общему времени работы:

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

T_{si} — общее время, проведённое ФМ i в состоянии полной загруженности, в котором ВМ не предоставляется требуемый уровень производительности, T_{ai} - общее время работы ФМ i.

2. PDM (Perfomance degradation due to migrations, ухудшение производительности, связанное с миграциями) :

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

C_{dj} — ожидаемое ухудшение производительности ВМ j в связи с миграцией,

C_{rj} — общие ресурсы CPU, запрошенные ВМ.

Ожидаемое ухудшение производительности существенно зависит от

даун-тайма, и , как результат, от модели живой миграции. Ряд работ [4][29] оперирует величиной, равной 10% от производительности CPU в MIPS в ходе всех миграций ВМ j.

Введение данных метрик также следует из опеределения источника потерь производительности: метрика OTF описывает процессы, происходящие с совокупностью ФМ, в то время как PDM — ВМ.

Независимость характеризации SLAV введёнными метриками позволяет ввести комбинированную метрику, отражающую все процессы:

$$SLAV = OTF \cdot PDM \quad .$$

4.2. Метрики энергопотребления.

Метрики определения энергосберегающего качества системы напрямую зависят от используемой модели энергопотребления. Данная работа рекомендует к использованию одну из моделей, введённых в подразделе 2.2.1. В разделе 5 использована экспериментальная модель, построенная на основании SPECpower[14]-тестов ФМ на базе HP ProLiant G4 и G5, ввиду источника данных для симуляции[32].Результаты тестов при шаге нагрузки 10% приведены в табл.1. Значения предоставляются в Ваттах.

Нагрузка ФМ	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
HP ProLiant G4	86	89,4	92,6	96	99,5	102	106	108	112	114	117
HP ProLiant G5	93,7	97	101	105	110	116	121	125	129	133	135

Табл. 1. Экспериментальные данные энергопотребления HP ProLiant G4/G5 в Вт.

Энергетической метрикой предлагается суммарная потреблённая всеми ФМ энергия E, измеренная в Квт-Ч.

4.3. Комбинированная балансная метрика и число миграций.

Метрики QoS и энергопотребления, как правило, негативно коррелируют между собой: обычно дополнительное энергосбережение достигается за счёт увеличения уровня SLAV. С другой стороны, целью задачи DRS-DPM является снижение энергопотребления и SLAV одновременно. С целью оценки баланса между DRS и DPM и оптимизации обоих процессов, введем комбинированную
метрику:

$$ESV = E \cdot SLAV \quad .$$

Другой важной метрикой является число миграций в ходе адаптации расположения ВМ. При прочих равных величинах, значение по метрике должно быть минимальным (раздел 3).

5. Симуляции.

В качестве проверки практической применимости вышеописанных методов в решении задачи DRS-DPM, были проведены симуляции средствами симулятора облачной инфраструктуры CloudSim, частично переделанного и дополненного под нужды выбранных алгоритмов и методов анализа. В качестве нагрузок были использованы трейсы нагрузок более чем 1000 реальных ВМ, измеренные в ходе работы проекта PlanetLab[32]. Данный набор нагрузок является референсным для многих исследовательских работ в сфере построения IaaS-решений[4][20][32]. Используя предложенные данные, симулировались комбинации эвристических алгоритмов определения перегруженности (ИКР, МАО, ЛР, ЛРР) и двух алгоритмов выбора ВМ (Минимальное время миграции, Минимальное время миграции — максимальное использование CPU). Кроме того, для определения оптимальных параметров каждого из

алгоритмов, были проведены отдельные симуляции для каждого из методов:

- ИКР, МАО: s=0.5..3.0 с шагом 0.5;
- ЛР, ЛРР: s=1.0..1.5 с шагом 0.1;

В соответствии с тестом Раяна-Джойнера[33], значения ESV-метрики, полученные в комбинациях алгоритмов, не соответствуют нормальному распределению с P<0.01 — значением. Следовательно, для сравнения алгоритмов с различными параметрами и получения оптимального как параметра, дающего минимальное значение среднего, были использованы средние значения ESV-метрики. Оптимальные значения: 1.5 для ИКР и 2.5 для МАО, 1.2 для ЛР и 1.2 для ЛРР.

Результаты проведения симуляции комбинаций алгоритмов с оптимальными параметрами приведены на рис 6-9.

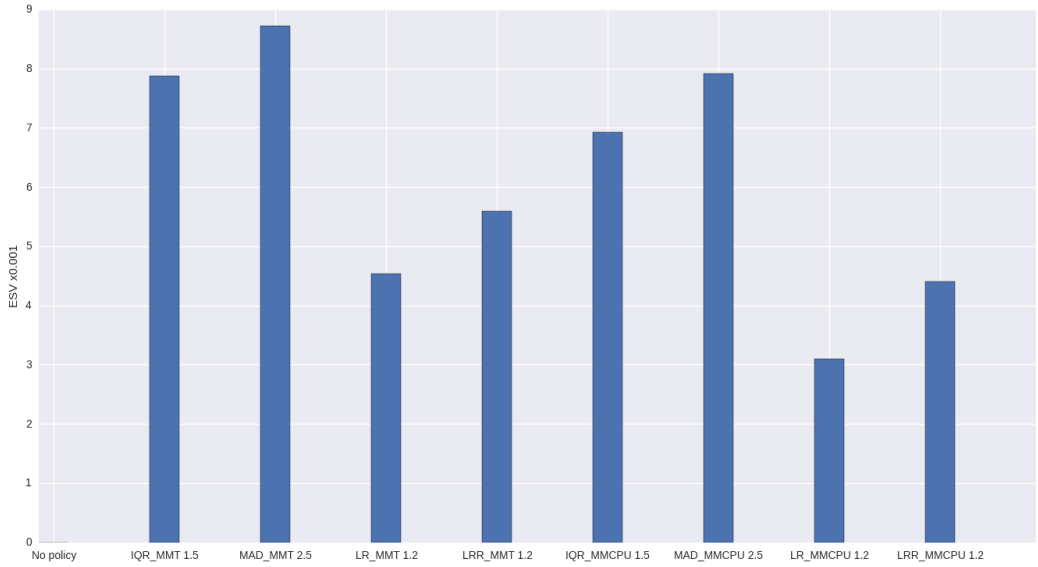

Рис. 6. ESV-метрика.

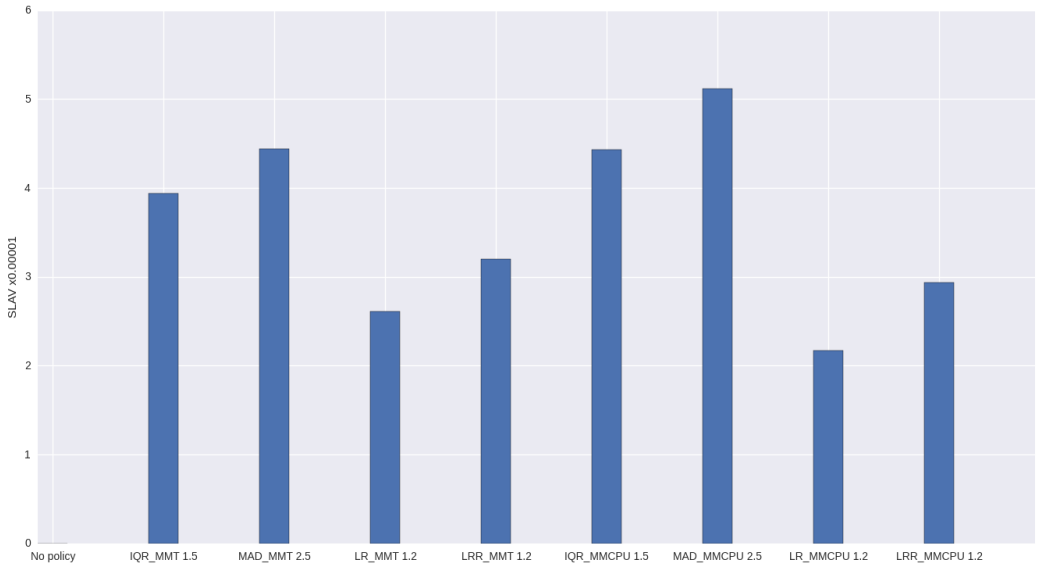

Рис. 7. SLAV-метрика.

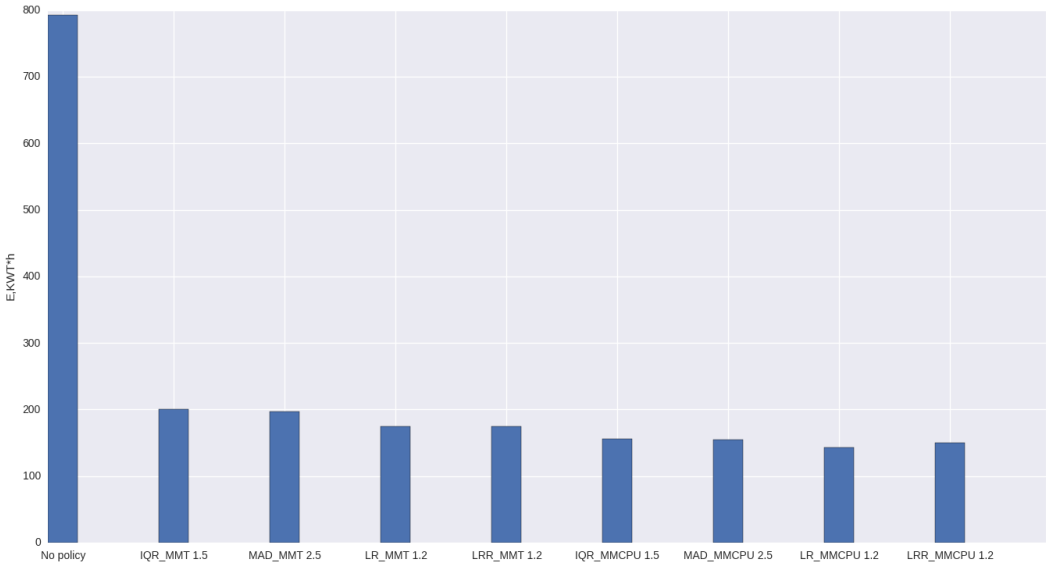

Рис. 8. Энергопотребление, Квт-Ч.

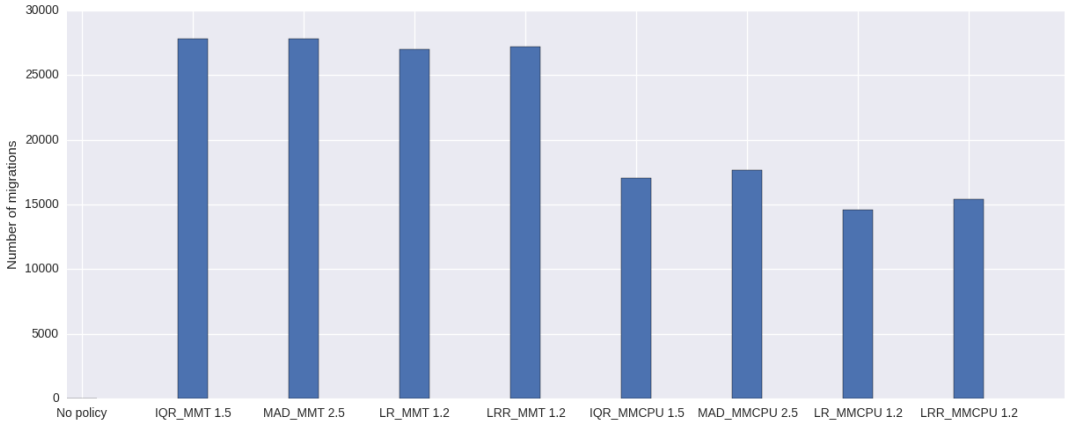

Рис. 9. Число миграций.

	IQRMT	MADMT	LRMT	LRRMT	IQRMU	MADMU	LRMU	LRRMU	NONE
E, КВт*Ч	200.06	196.53	173.98	174.90	156.29	154.46	142.83	49.60	792.11
ESV, $x10^{(-3)}$	7.87	8.72	4.54	5.6	6.93	7.91	3.1	4.4	0
SLAV, $x10^{(-5)}$	3.94	4.44	2.61	3.2	4.43	5.12	2.17	2.94	0
OTF %	4.92	4.93	3.73	4.0	5.54	5.69	3.62	4.2	0
PDM %	8	9	7	8	8	9	6	7	0
MIG_N	27788	27849	26998	27178	17064	17667	14601	15431	0

Табл. 2. Результаты симуляций.

6. Выводы.

На основании полученных результатов, следует сделать несколько выводов:

1. Динамическая консолидация ВМ в комбинации с локальным DPM даёт существенный выигрыш в минимизации энергопотребления (рис. 5.3).
2. Методы локальной регрессии превосходят методы с адаптивной границей ввиду меньшего уровня SLAV и числа миграций ВМ.
3. Локальная регрессия даёт лучшие результаты по сравнению с модификацией Кливланда, что может быть объяснено тем, что для симулируемой нагрузки важнее оперативно реагировать на «быстрые»

пики нагрузок, нежели размывать последние по невязке. Вопрос практического компромисса между ЛР и ЛРР остаётся открытым и будет исследован автором в дальнейшем.

4. Вопрос выбора ВМ также остаётся открытым в практическом смысле. Использование смешанной политики выбора ВМ «Минимальное время миграции при максимальном использовании CPU» даёт статистически лучшие результаты, снижая число миграций, однако, ввиду применения простейшей модели для миграции, без учёта корреляции нагрузки CPU со сложностью живой миграции, не учитывается ряд важных факторов, которые потребуется анализировать при построении реального практического решения.

7. Заключение.

Проведённая автором работа определяет модель динамического управления облачной системой с целью соответствия QoS и минимизации энергопотребления. Децентрализованность решения трёх подзадач из четырёх обеспечивает распределённость, высокую доступность и отказоустойчивость системы в случае выхода из строя некоторых компонент. Также происходит упрощение работы по расположению ВМ: глобальный планировщик контролирует весь набор ВМ и весь диапазон ФМ, вообще говоря, только в начальном расположении. Далее активность по определению перегруженности, недогруженности, мигрируемых ВМ и переключения ФМ, за исключением команды приведения выключенной ФМ в активное состояние и поиска расположения подмножества ВМ, передаётся локальным планировщикам ФМ.

Граничный метод определения недогруженности позволяет оперативно решать подзадачу отключений незагруженных ФМ, адаптивные эвристические методы определения перегруженности — осуществлять

разгрузку и поддерживать QoS. К сожалению, недостатком данных методов является отсутствие прямого управления уровнем QoS в виде явной SLAV-константы. Теоретическое обоснование существования эффективного онлайн-алгоритма, реализующего прямое управление, вывод свойств данного алгоритма (максимизация межмиграционного времени, оптимальная реакция на SLAV) также приведены в работе, а практическая реализация метода — вопрос дальнейших исследований автора.

8.Дальнейшие исследования.

В дальнейшем, автором планируется проведение ряда исследований по выявлению практических особенностей интеграции приведённых алгоритмов в реальные облачные системы на основе различных технологий виртуализации, построение оптимального онлайн-алгоритма с явным управлением QoS-уровнем. В поле исследований также попадает уточнение модели живой миграции, учёт процесса репликации загрязняемых страниц памяти и других особенностей, необходимых для построения адекватного и эффективного решения по динамической оптимизации в IaaS.

9. Литература:

1. Ефанов Н. Н., Мелехова А. Л., Бондарь А. О. Алгоритмы динамического управления энергопотреблением в облачной системе, «Программная инженерия», №4, 2015, С. 20 – 30
2. ASHRAE publicatios, 2014 . Электронный источник: https://www.ashrae.org/resources--publications
3. J. Koomey, “Worldwide electricity used in data centers”. Environmental Research Letters. vol. 3, no. 034008. September 23

4. Anton Beloglazov, Rajkumar Buyya, “Energy Efficient Resource Management in Virtualized Cloud Data Centers”, *IEEE TCSC Doctoral Symposium, Proceedings of the 10th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing (CCGrid 2010)*, Melbourne, Australia, May 17- 20, 2010.
5. VMWare Russia: vSphere и vSphere with Operations Management, 2016 Электронный источник: http://www.vmware.com/ru/products/vsphere/features/drs-dpm
6. Карпов Д.В., Бондарь А.О., Энергосбережение изнутри: что в действительности могут измерить профилировщики , RSDN Magazine, 2013.
7. ACPI specification. Электронный источник: http://www.acpi.info
8. S. Mittal, "A survey of techniques for improving energy efficiency in embedded computing systems", IJCAET, 6(4), 440–459, 2014.
9. Minas and B. Ellison,Energy Efficiency for Information Technology: How to Reduce Power Consumption in Servers and Data Centers. Intel Press, 2009.
10. Bishop Brock, Karthick Rajamani, «Dynamic Powe Management for Embedded Systems, IBM Research, 2001.
11. Wireless Local-Area Network Fundamentals. — М.: «Вильямс», 2004. — С. 304. — ISBN 5-8459-0701-2
12. Intel® 64 and IA-32 Architectures Developer's Manual: Vol. 3B, Intel Labs, 2016.
13. Fan, W. D. Weber, and L. A. Barroso, “Power provisioning for a warehouse-sized computer,” in Proceedings of the 34th Annual International Symposium on Computer Architecture (ISCA) , 2007, pp. 13–23.
14. The SPECpower benchmark. Электронный источник: http://www.spec.org/power_ssj2008/

15. F. Farahnakian, P. Liljeberg and J. Plosila, "LiRCUP: Linear Regression Based CPU Usage Prediction Algorithm for Live Migration of Virtual Machines in Data Centers," *2013 39th Euromicro Conference on Software Engineering and* Advanced *Applications*, Santander, 2013,pp. 357-364.
16. Электронный источник: https://github.com/nefanov/
17. Susanta Nanda Tzi-cker Chiueh, A Survey on Virtualization Technologies, Department of Computer Science SUNY at Stony Brook, Stony Brook, NY, 2015.
18. Mike Day, A Survey of High-Performance Virtualization Techniques, IBM RESEARCH, 2014.
19. A. Borodin, R El-Yaniv, Online computation and competitive analysys. Cambridge University Press, New York, 1998, vol. 53.
20. Anton Beloglazov and Rajkumar Buyya, “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 (TPDS)*, 24(7), pp. 1366-1379, 2013.
21. Cleveland, William S.; Devlin, Susan J. (1988). "Locally-Weighted Regression: An Approach to Regression Analysis by Local Fitting". Journal of the American Statistical Association, vol. 83 (403): 596–610.
22. W. S. Cleveland, “Robust locally weighted regression and smoothing scatterplots”, Journal of the American Statistical Association, vol. 74, no. 368, pp. 829–836, 1979.
23. Norris, James R. Markov chains. Cambridge University Press, 1998.
24. Everitt,B.S. The Cambridge Dictionary of Statistics, CUP, 2002.
25. Parzen, E. , Stochastic Processes, Holden-Day, 1962.
26. Dodge, Y. , The Oxford Dictionary of Statistical Terms, OUP, 2003.
27. S. P. Meyn, R.L. Tweedie, Markov Chains and Stochastic Stability, 2005.
28. O. Luiz, A. Perkusich, A. M. N. Lima, “Multisize sliding window in

workload estimation for dynamic power management," *IEEE Transactions on Computers*, vol. 59, no. 12, pp. 1625–1639, 2010.

29. Christopher Clark, Keir Fraser, Steven Hand, Jacob Gorm Hansen, Eric Jul, Christian Limpach, Ian Pratt, Andrew Warfield Live Migration of Virtual Machines, University of Cambridge Computer Laboratory Department of Computer Science 15 JJ Thomson Avenue, Cambridge, UK, 2005.
30. William Voorsluys , James Broberg , Srikumar Venugopal , Rajkumar Buyya, Cost of Virtual Machine Live Migration in Clouds: A Performance Evaluation, School of Computer Science and Engineering, The University of New South Wales, Sydney, Australia, 2009.
31. M. Yue, "A simple proof of the inequality FFD (L)<11/9 OPT (L)+ 1,for all l for the FFD bin-packing algorithm,"Acta Mathematicae Applicatae Sinica (English Series) ,vol. 7, no. 4, pp. 321–331, 1991.
32. S. Park and V. S. Pai, "CoMon: a mostly-scalable monitoring system for Planet-Lab", ACM SIGOPS Operating Systems Review , vol. 40, no. 1, pp. 65–74,2006.
33. Ryan T.A., Joiner B.L. Normal Probability Plots and Tests for Normality, Technical Report, Statistics Department, The Pennsylvania State University, 1967.