На правах рукописи

МЕЛЕХОВА Анна Леонидовна

УПРАВЛЕНИЕ ФИЗИЧЕСКОЙ ПАМЯТЬЮ ВИРТУАЛЬНОЙ МАШИНЫ

Специальность 05.13.11 – математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей

Автореферат диссертации на соискание
ученой степени кандидата технических наук

Москва — 2015

Работа выполнена на кафедре информатики Федерального государственного автономного образовательного учреждения высшего профессионального образования «Московский физико-технический институт (государственный университет)»

Научный руководитель: Тормасов Александр Геннадиевич,

доктор физико-математических наук, профессор, зав. кафедрой «Прикладной информатики» Федерального государственного автономного образовательного учреждения высшего профессионального образования «Московский физико-технический институт (государственный университет)», ректор автономной некоммерческой организации высшего образования «Университета Иннополис»

Официальные оппоненты: Гергель Виктор Павлович,

доктор технических наук, профессор, директор Института Информационных технологий, математики и механики Федерального государственного автономного образовательного учреждения высшего образования «Нижегородский государственный университет им. Н.И. Лобачевского», директор Научно-исследовательского института прикладной математики и кибернетики, руководитель Приволжского научно-образовательного Центра суперкомпьютерных технологий

Ицыксон Владимир Михайлович,
кандидат технических наук, доцент, и.о. зав. кафедрой "Компьютерные системы и программные технологии" Института Компьютерных Наук и Технологий Санкт-Петербургского Политехнического университета имени Петра Великого

Ведущая организация: Федеральное государственное учреждение “Федеральный исследовательский центр “Информатики и управления” Российской академии наук”

Защита состоится «28» января 2016 года в 15 часов на заседании диссертационного совета Д 002.087.01 при Федеральном государственном бюджетном учреждении науки Институте системного программирования Российской академии наук по адресу: 109004, г. Москва, ул. А. Солженицына, дом 25, конференц-зал (комната 110).

С диссертацией можно ознакомиться в библиотеке и на сайте ИСП РАН http://www.ispras.ru/dcouncil/
Автореферат разослан «____» _____ 2015 г.
Учёный секретарь диссертационного совета

кандидат физико-математических наук /Зеленов С.В./

ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ

Актуальность темы

За последнее десятилетие виртуализация перестала быть узкопрофессиональной технологией с ограниченным применением. Поддержка виртуализации появилась как в распространенных операционных системах, так и в аппаратном обеспечении.

Одним из основных применений виртуализации является консолидация серверов. Виртуализация позволяет разместить несколько виртуальных серверов на одном физическом и ограничить их в фактическом потреблении ресурсов. Операционная система (ОС), исполняющаяся внутри виртуальной среды, - гостевая ОС – будет полагать, что в ее распоряжении вся полнота назначенного ресурса. Но виртуализационное программное обеспечение (монитор виртуальной машины (ВММ) или гипервизор) - будет изменять объем ресурса, предоставляемого в пользование гостевой ОС незаметным для нее образом. Этот процесс можно назвать мультиплексированием или виртуализацией ресурса. Близкой аналогией здесь является выполнение процессов в традиционной системе, где ОС незаметно для исполняемых процессов перераспределяет между ними память, вычислительную мощность и пропускную способности сети в соответствии с квотами и фактическими потребностями. Так и гипервизор управляет виртуальными машинами[1], перераспределяя ресурсы в соответствии с их фактическими потребностями.

Практическая польза от виртуализации становится особенно заметной, когда суммарный объем ресурсов, назначенных виртуальным машинам, превосходит доступный объем ресурсов на физической машине. Этот режим работы получил названия переподписки (oversubscription) или переназначения (overcommitment) ресурса. На практике, режим переподписки позволяет существенно повысит плотность виртуальных серверов .

Приведенная выше аналогия про процессы оказывается неточной, когда речь заходит о режиме переподписки. Исполняемые в системе процессы не занимаются распределением физических ресурсов и полагаются в этом на операционную систему. При виртуализации виртуальные машины получают ресурсы от гипервизора, но внутри них гостевые ОС самостоятельно выполняют распределение этих ресурсов между своими процессами. При этом алгоритмы гостевых ОС существенно полагаются на то, что ОС монопольно владеет физическими ресурсами и может обращаться к ним с минимальными задержками. В режиме переподписки гостевая ОС неизбежно столкнется с тем, что из выделенных ей ресурсов только часть доступна физически, а доступ к оставшейся части связан с большими задержками и

[1] Под виртуальной машиной мы подразумеваем комплекс из гостевой ОС, виртуальной среды исполнения и монитора виртуальной машины.

накладными вычислительными расходами (что, например, происходит при подкачке страниц памяти). В результате, алгоритмы распределения перестают эффективно работать, и общая производительность гостевой ОС резко снижается.

Чтобы обеспечить работу в режиме переподписки без существенной потери эффективности, необходимо оценить фактический уровень использования ресурса виртуальной машиной и выделить ей ресурс соответственно оценке. Самым сложным для оценки является ресурс физической памяти. В архитектуре процессоров Intel x86 нет возможности, помечать страницы физической памяти как неиспользуемые. Поэтому требуются особые алгоритмы оценки размера рабочего набора – страниц физической памяти, к которым гостевая ОС обращается. И по мере распространения систем виртуализации и облачных сервисов востребованность данной оценки возрастает.

Память назначается виртуальной машине при запуске гостевой ОС. В режиме переподписки происходит вытеснение части физической памяти у виртуальной машины. В ранних системах виртуализации использовались алгоритмы вытеснения, применяемые для вытеснения памяти у процесса в традиционных ОС. В современных системах применяется другой подход, называемый фиктивным занятием физической памяти гостевой ОС (англ. ballooning). Для его реализации в гостевую ОС внедряется специальный агент, который получает (или «захватывает») определенный объем физической памяти, используя стандартные средства операционной системы. Агент передает информацию о захваченных страницах гипервизору, который может затем перераспределить их в пользу другой виртуальной машины.

Вытеснение по первому способу происходит незаметно для гостевой ОС, в то время как фиктивное занятие памяти для нее очевидно и, поэтому, более эффективно обрабатывается ее алгоритмами распределения памяти. В современных системах виртуализации основная работа по вытеснению памяти реализуется через фиктивный захват, а более старый механизм урезания рабочего набора виртуальной машины является вспомогательным и не играет существенной роли.

Управление агентом фиктивного захвата (balloon) важно для сохранения эффективной работы гостевой ОС и всей системы виртуализации в целом. При малом объеме фиктивно захваченной памяти другие виртуальные машины в системе могут испытывать недостаток физической памяти, что снижает их производительность. При чрезмерном фиктивном захвате растет интенсивность страничных промахов гостевой ОС (гостевая подкачка): чем больше страниц физической памяти изымается, тем больше увеличивается нагрузка (давление) на остальные страницы системы.

Корректная оценка размера рабочего набора способна решить эту проблему. При наличии подобной оценки можно было бы фиктивно

занимать объем памяти, равный назначенной памяти за вычетом размера рабочего набора, что позволило бы избежать гостевой подкачки. Однако, в силу искусственности самого понятия рабочего набора, использование подобной оценки требует специальных мер.

Фиктивный захват изменяет работу гостевой ОС, что отражается так называемыми «гостевыми счетчиками» – статистическими параметрами, автоматически вычисляемыми гостевой ОС для целей внутреннего мониторинга. Оценка размера рабочего набора гостевой операционной системы может базироваться как на предыдущих наблюдениях, так и на гостевых счетчиках. Кроме того, оценка может использовать данные гипервизора, которая отражает запросы гостевой ОС к привилегированным операциям и внешним устройствам – так называемые «виртуализационные счетчики».

Сложность оценки размера рабочего набора связана с высокой скоростью изменения виртуализационной нагрузки, что сильно снижает точность оценки и ее применимость. При использовании оценки размера рабочего набора для расчета объема фиктивно занятой памяти, система виртуализации должна быть готова к быстрой коррекции в случае неблагоприятных последствий. То есть, система должна не просто использовать оценку, но и адаптировать свое решение в зависимости от наблюдаемого поведения гостевой ОС. *Построение системы, регулирующей объем фиктивно занятой памяти на основании оценки рабочего набора и корректирующей его в зависимости от реакции гостевой операционной системы, поможет сэкономить физическую память для достижения большой плотности размещения виртуальных машин без привнесения существенных накладных расходов, тем самым повысив экономическую эффективность использования технологий виртуализации без снижения уровня обслуживания.*

Цель работы, задачи исследования

Целью данной диссертационной работы является разработка методов и алгоритмов, ограничивающих потребление физической памяти гостевой операционной системы через управление агентом фиктивного занятия физической памяти (balloon), с последующей реализацией их в виде программного комплекса. Разрабатываемый программный комплекс должен осуществлять:

1. динамическую оценку размера рабочего набора гостевой ОС на основании статистических параметров гостевой ОС («гостевых счетчиков»);
2. корректировку оценки в соответствии с виртуализационной и гостевой статистиками, используя методы обучения с подкреплением;

3. управление агентом фиктивного занятия физической памяти (balloon) в соответствии с полученной оценкой для достижения целевых параметров.

Задачи исследования:

- отбор значимых параметров гостевой и виртуализационной статистики;
- проверка гомогенности отобранных виртуализационных параметров для различных версий операционных систем;
- построение оценки размера рабочего набора на основании гостевой статистики;
- разработка системы штрафов для корректировки полученной оценки размера рабочего набора через виртуализационную статистику;
- разработка программного комплекса, управляющего агентом фиктивного занятия физической памяти (balloon) в соответствии с предложенной оценкой;
- настройка и расширение имеющейся автоматизированной системы тестирования для подбора эмпирических параметров системы.

В ходе выполнения исследований автором были проведены эксперименты, подтверждающие эффективность разработанной системы. Диссертационное исследование выполнялось с использованием облачной инфраструктуры Индустриального партнёра ООО «Акронис» по договору с ООО «Проект ИКС» о выполнении прикладных научных исследований при финансовой поддержке Министерства образования и науки Российской Федерации. Соглашение о предоставлении субсидий № 14.579.21.0010. Уникальный идентификатор Соглашения RFMEFI57914X0010.

Методы исследования

При оценке размера рабочего набора гостевой операционной системы широко использовались методы машинного обучения, включая регрессионные модели и байесовские методы, методы анализа данных. При корректировке оценки применялись методы обучения с подкреплением. При разработке программного комплекса потребовались методы виртуализации и облачных систем хранения данных. При проведении экспериментов и оценке их результатов применялись методы теории вероятностей.

Для проведения численных исследований предложенная программная система была реализована как составная часть программного

комплекса Parallels Server. Реализация в системе с открытым кодом Linux KVM подтверждает переносимость на другие платформы. Были осуществлены эксперименты с использованием реальных программных комплексов от сторонних производителей и модельных приложений внутренней разработки.

Научная новизна

В работе получены следующие основные результаты, обладающие научной новизной:

1. Исследована возможность оценки объема потребляемой памяти гостевой операционной системой исключительно на основании виртуализационной статистики. Полученные результаты свидетельствуют о недостаточности информации, полученной только от систем виртуализации, для построения корректной оценки.
2. Подтверждено эмпирически, что объединение виртуализационной и гостевой статистик дает достаточно информации для адекватного описания поведения гостевой системы.
3. Разработан и реализован в рамках Parallels Server алгоритм адаптивной коррекции оценки размера рабочего набора через штрафы, назначаемые на основании виртуализационной статистики.
4. Подтверждена переносимость алгоритма посредством его реализации в системе с открытым исходным кодом Linux (модуль гостевых расширений KVM).

Теоретическая и практическая значимость

Разработанный программный комплекс успешно интегрирован в систему управления ресурсами в программном комплексе Parallels Server, который является широко распространенным среди хостинг-провайдеров программным решением и обеспечивает бесперебойную работу нескольких сот тысяч виртуальных выделенных серверов.

Отдельные наработки также перенесены в виртуализационную систему с открытым исходным кодом (Linux KVM). Предложенный подход корректировки объема фиктивно занятой памяти посредством штрафов от виртуализационной статистики может быть применен в других виртуализационных платформах.

Предложенные и разработанные в диссертации методы и программные средства адаптивного управления ресурсом физической памяти

виртуальной машины включены в разрабатываемые ООО «Проект ИКС» технологии и программное обеспечение распределенных и высокопроизводительных вычислительных систем для хранения и обработки больших данных, отвечающие за балансировку нагрузки в облаке виртуальных машин.

Публикации

По материалам данного диссертационного исследования были опубликованы работы [1-10]. Работы [4, 6-8] выполнены автором лично. В работе [5] автору принадлежит роль исполнителя. В работах [2, 3, 10] автор выполнял постановку задачи, вносил редактуру и дополнения по предметной области. В работах [1, 9] автору принадлежит постановка задачи, редактура модели, консультации по возникающим трудностям при разработке модели, методы верификации модели. В списке изданий присутствуют издания рекомендованные ВАК ([1, 2, 3]), индексируемые системами WebOfScience и SCOPUS ([8, 9, 10]), и один патент ([5]).

Апробация результатов работы

Результаты диссертационного исследования докладывались и обсуждались на российских и международных научных конференциях:

- Гагаринские чтения (Москва, 2010);
- IEEE Sixth International Conference on Cloud Computing CLOUD (USA, Santa Clara, 2013);
- Second International Conference "Cluster Computing" (Ukraine, Lviv, 2013);
- «Облачные вычисления. Образование. Исследования. Разработка 2014» (Москва, 2014);
- IARIA Cloud Computing (France, Nizza, 2015).

Актуальность задачи и ее возможные решения также были обсуждены на популярной технической конференции Highload++ (Москва, 2011).

Результаты работы реализованы в виде компоненты программного комплекса Parallels Cloud Server. Также подготовлен набор изменений для внедрения алгоритма в Linux KVM.

Положения, выносимые на защиту

На защиту выносятся следующие основные положения:

1. Алгоритм управления физической памятью виртуальной машины на основании оценки размера рабочего набора гостевой операционной системы, корректируемой через штрафы, которые назначаются на основании виртуализационной статистики;

2. Обоснование эффективности алгоритма управления как имеющего приемлемые накладные расходы, дающего высокий выигрыш по потреблению памяти и не приводящего к снижению производительности гостевой ОС при быстро меняющихся нагрузках;
3. Реализация алгоритма в виде комплекса программ системы управления памятью как часть программного комплекса Parallels Server.

Структура и объем диссертации

Диссертация состоит из введения, пяти глав, заключения и двух приложений. Работа изложена на 105 страницах. Список использованных источников содержит ссылки на 110 публикаций.

Содержание работы

Во введении обосновывается актуальность темы, представляется технологический контекст проблемы, вводится используемая терминология. Также формулируются цели исследования и основные результаты, выносимые на защиту, обосновывается научная и практическая значимость выполненного исследования.

В последние десять лет спрос на виртуализацию в информационных технологиях увеличивается. Виртуализационные решения появились в большинстве распространенных операционных систем ориентированных на конечного пользователя, как то: Microsoft Windows Hyper-V, Linux KVM, Apple OS X hypervisor. Производители аппаратного обеспечения один за другим также добавляют поддержку виртуализационных технологий: Intel VT-x, AMD AMD-V, ARM. И даже крупные производители видеокарт предлагают свои виртуализационные решения, как пример можно привести NVIDIA GRID и Intel gVirt.

Одним из основных применений виртуализации является консолидация серверов. Такое применение обосновано пикообразным характером большинства современных рабочих нагрузок. Под пикообразным характером понимается, что в основную часть времени используется лишь небольшая доля ресурса, но в определенные моменты такого же объема ресурса перестает хватать, за счет чего возникают существенные падения производительности. Традиционный подход при фиксированных ресурсах предполагает компромисс между пиковой производительностью и стоимостью ресурсов в простое. С помощью виртуализации становится возможным разместить несколько виртуальных серверов на одном физическом и ограничить их в фактическом потреблении ресурса. Такое уплотнение серверов называется консолидацией, а режим, при котором количество назначенных виртуальных ресурсов превышает объем физических, носит название переподписки (oversubscription).

Достижение эффективного распределения ресурса между виртуальными машинами в условиях дефицита ресурса есть задача управления физической памятью.

В **главе 1** дан краткий обзор наиболее актуальных и значимых современных алгоритмов для управления физической памятью виртуальных машин. Для каждого алгоритма рассмотрены его достоинства и недостатки, приведены возможные комбинации. Особый акцент ставится на практической применимости. В частности, приводится обоснование, почему некоторые из алгоритмов, часто цитируемые в академической среде, не нашли широкого практического внедрения.

Задача управления памятью в виртуальной машине включает нескольких подзадач:

1. Оценка размера рабочего набора (то есть число страниц, к которым гостевая ОС обращалась в недавнем прошлом);
2. Виртуальная подкачка:
    a. выбор страницы памяти, которую вытеснят из рабочего набора с сохранением ее содержимого в файле подкачки;
    b. механизм сброса выбранной страницы в файл подкачки;
    c. подгрузка страниц из файла подкачки (по необходимости или с предвыборкой);
3. Сокращение общего объема потребляемого ресурса.

При решении задачи управления важно задать критерий эффективности. Когда говорится об эффективности алгоритмов управления памятью, то наиболее значимым показателем оказывается соотношение между объемом сэкономленного ресурса и снижением производительности гостевой ОС в сочетании с возникшими издержками – то есть вычислительными ресурсами и ресурсом оперативной памяти, необходимыми для исполнения алгоритма. Так, алгоритмы компрессии и дедупликации, решающие задачу 3 (сокращение общего объема потребляемой памяти), стабильно вносят значительные издержки, в то время как объем сэкономленного ими ресурса не является постоянным и зависит от конкретного набора данных и от задач, исполняемых в гостевой ОС. Поэтому не все системы виртуализации реализуют эти подходы.

Самым востребованным решением, встречающимся во всех распространенных виртуальных машинах, является алгоритм фиктивного занятия физической памяти (ballooning). Алгоритм реализует подзадачу виртуальной подкачки (задача 2.a) – он выбирает страницы памяти для вытеснения из текущего набора страниц физической памяти виртуальной машины. Несомненное достоинство алгоритма в гарантированном отсутствии обращений гостевой ОС к страницам, вытесненным через него. Эта техника не привносит больших издержек, до определенного момента

почти не снижает производительность гостевой ОС и может «сэкономить» половину объема памяти, назначенной виртуальной машине. Термин ballooning (дословно «раздувание», «создание воздушного шара») ведет свое происхождение от метафоры воздушного шарика, которая очень хорошо описывает происходящее. В гостевой ОС работает специальный агент (balloon), который выделяет память средствами гостевой операционной системы и резервирует ее, тем самым блокируя возможное обращение к ней других программ. Полученную таким образом память агент отдает гипервизору. С точки зрения гостевой операционной системы и программ, работающих в ней, страницы физической памяти заняты в работе. С точки зрения фактического использования ресурса страницы свободны и пусты: то есть в виртуальной физической памяти возникает пустотность – воздушный шар (рис.1). Недостатком техники является потенциально возникающая гостевая подкачка: чем больше страниц высвобождается через надувание (через ballooning), тем больше увеличивается нагрузка (давление) на остальные страницы системы. Тем не менее, корректная оценка размера рабочего набора (задача 1) способна решить эту проблему. Тогда размер фиктивно занятой памяти можно было бы ограничить указанной оценкой, что снизило бы потери производительности гостевой ОС.

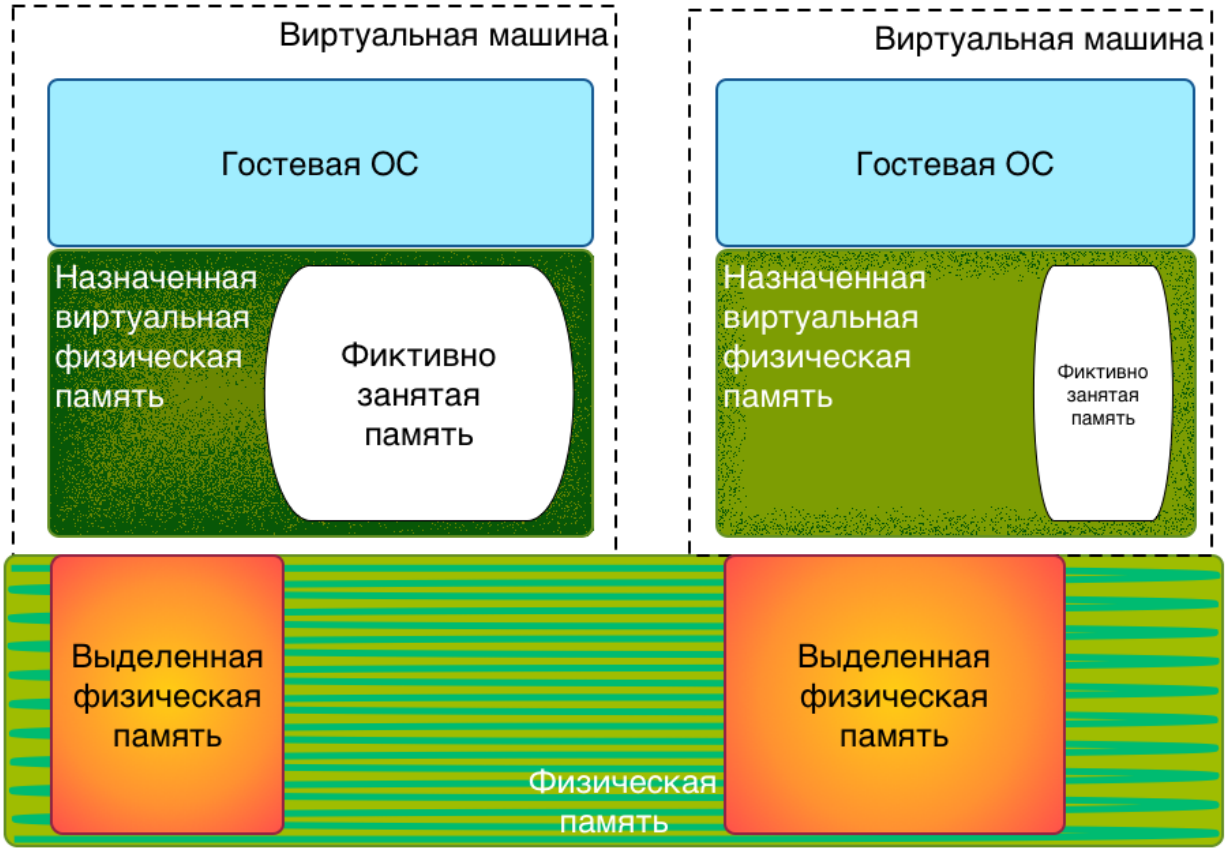

Рис. 1. Фиктивное занятие памяти

Однако, сложность состоит в искусственности самого понятия рабочего набора. Рабочий набор – это подмножество страниц физической памяти, к которым было обращение в течение некоторого времени. Память сервисного

процесса, который ставится на исполнение раз в 10 минут, окажется вне рабочего набора, если он оценивается по последней минуте. Запуск нового процесса изменит рабочий набор, добавив в него новые страницы. При этом и сама гостевая система может дополнительно применять алгоритмы для улучшения своей производительности и времени отклика. Так, при активном потребителе система может предвосхищать будущие запросы на выделение памяти, а при прекращении нагрузки система может не сразу передавать высвободившийся ресурс в пул свободных страниц. Все вышеперечисленное делает любую оценку размера рабочего набора неточной. Кроме того, сам агент фиктивного выделения памяти (balloon) инертен: выделяя память средствами гостевой ОС, он ограничен в скорости получения ресурса. Таким образом, управление объемом фиктивно занятой памяти оказывается сложной задачей. В главе 1 представлен анализ решения этой проблемы в разных гипервизорах.

Глава 2 посвящена упорядочиванию информации о доступных счетчиках и анализу их применимости для нашей задачи. Выделяют две основные категории счетчиков – гостевые и виртуализационные. Гостевые счетчики – это статистические данные, собираемые гостевой операционной системой для отладочных/сервисных нужд. Виртуализационные счетчики производят учет и категоризацию виртуализационных событий (привилегированных операций гостевой операционной системы), осуществляемых гипервизором.

На первый взгляд использование гостевых счетчиков выглядят более оправданным вариантом. В самом деле каждая операционная система так или иначе информирует пользователя о количестве свободной памяти. Но у большинства систем подобные счетчики недостаточно информативны. Например, пул свободных страниц во многих системах наполняется по требованию – то есть, если какой-то процесс использовал память, а потом прекратил её использование, его страница остается прикрепленной к нему до тех пор, пока другому процессу не потребуется память. Более того, для страниц внутри пула может быть запланирована операция заполнения страницы нулями, которая с точки зрения виртуализационной системы есть возобновление использования страницы памяти. А если учесть разнообразные кэши операционных систем, алгоритмы дедупликации и сжатия, то становится ясным, что гостевые счетчики, не предоставляют возможности для простого единообразного решения задачи с небольшими накладными расходами на осуществление оценки..

Спектр событий, видимый через призму двух статистик, упомянутых выше, существенно различается. В отношении памяти гостевые счетчики более информативны, когда речь идет о суммарном объеме виртуальной памяти процессов, о размере закрепленной памяти (locked memory - памяти, принадлежащей одному процессу и не подлежащей перераспределению), но

не всегда адекватно говорят о количестве страничных промахов (page fault). Важно отметить, что количество страничных промахов свидетельствует о скорости изменения рабочих наборов процессов гостевой ОС, о возможной подгрузке страниц из страничного файла (свопинге). Иными словами, количество страничных промахов является важным индикатором состояния системы. Этот параметр можно отслеживать в гипервизоре при незначительных изменениях кода и практически при нулевых накладных расходах. Кроме того, некоторые информативно важные показатели в принципе недоступны для гостевой ОС. К таковым относится, например, число виртуализационных страничных промахов – показатель, описывающий как часто обращение к гостевой странице порождает запрос к гипервизору на выделение новой страницы памяти (запрос о предоставлении ресурса). Количество счетчиков гостевой операционной системы существенно превосходит количество виртуализационных счетчиков. В то же время накладные расходы от считывания этих показателей значительно превышают издержки на сбор виртуализационной статистики. И, конечно, гостевые счетчики имеют существенную вариативность в зависимости от семейства операционных систем (Windows, Linux, OS/2, OSX) и конкретных версий системы (Windows 2008 и Windows 10). В таблице ниже представлены основные характеристики сравнения гостевой и виртуализационной статистик

Таблица 1. Сравнение счетчиков

Характеристика	Гостевые счетчики	Виртуализационные счетчики
Количество счетчиков	10^3-10^4	50-200
Издержки на сбор статистики	Много обращений к памяти, передача информации из гостевой ОС в управляющий виртуализационный модуль (гипервизор)	Время доступа к памяти + несколько дополнительных тактов
Стабильность счетчиков	Меняются от версии к версии гостевой ОС	Количество счетчиков постоянно сокращается, новые не появляются
Сложность поддержки кода, собирающего счетчики	Код специфичен для каждой гостевой ОС и даже для разных версий одной ОС	Код простой, не зависит от гостевой ОС
Информативность счетчиков	Детально отражает информацию о гостевой ОС, но не о состоянии монитора виртуальной машины	Косвенно отражает состояние гостевой ОС, детально описывает виртуализацию

Как результат анализа, проведенного в главе 2, было принято решение попытаться построить оценку на основании виртуализационных счетчиков, так как они вносят меньше накладных расходов и реализация их сбора

технически проще.

В **главе 3** приведен анализ гомогенности виртуализационных выборок гостевых систем семейства Windows. Целью исследования была проверка гипотезы, что виртуализационные выборки от близких версий Windows принадлежат одному распределению. Позитивный ответ означал бы, что с точки зрения виртуализации внутрисистемные алгоритмы близких версий ОС выглядят одинаково. А значит при выборе в пользу виртуализационных счетчиков количество исследуемых операционных систем было бы меньше, что, несомненно, дополнительное преимущественно виртуализационной статистики перед гостевой.

Для проводимого исследования был выбран критерий согласия Андерсона – как критерий, работающий на выборках с неизвестным законом распределения, и как наиболее мощный из рассматриваемых двухвыборочных критериев согласия. Однако, расчеты также приводились и для широко применяемого критерия Вилкоксона, а также критерия Зигеля-Тьюки. Исследования проводились относительно пар систем (Windows 2008 x64 и Windows 7 x64) и (Windows XP x32 и Windows 7 x32). При этом проверялась как однородность типовых схем потребления памяти, так и однородность типовых схем приращения памяти.

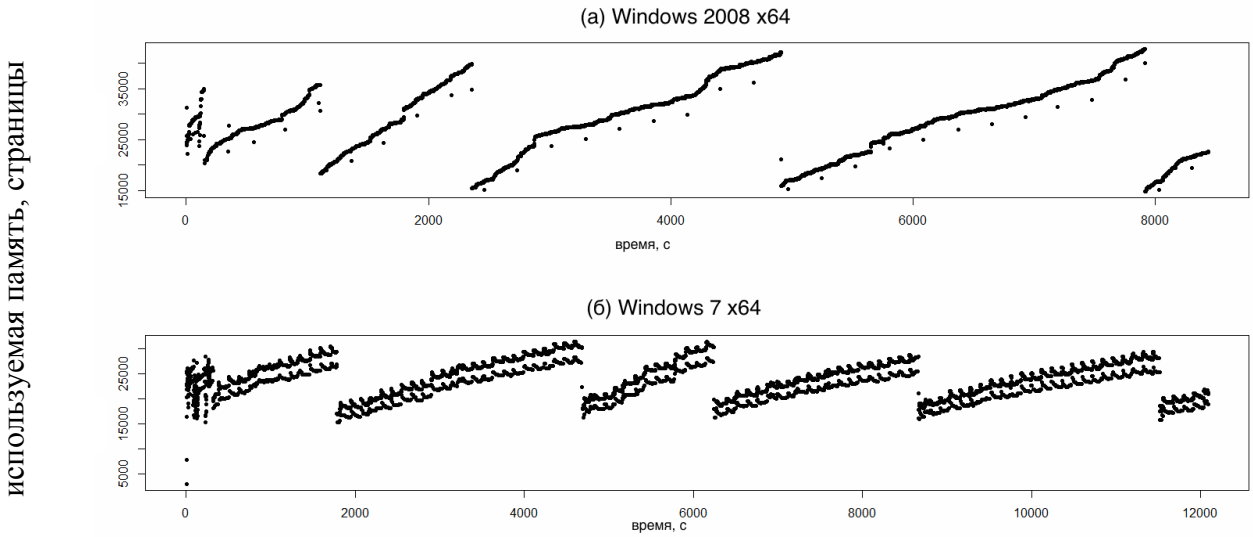

Рис. 2. График потребления памяти в Windows 2008 x64 (а) и Windows 7 x34 (б)

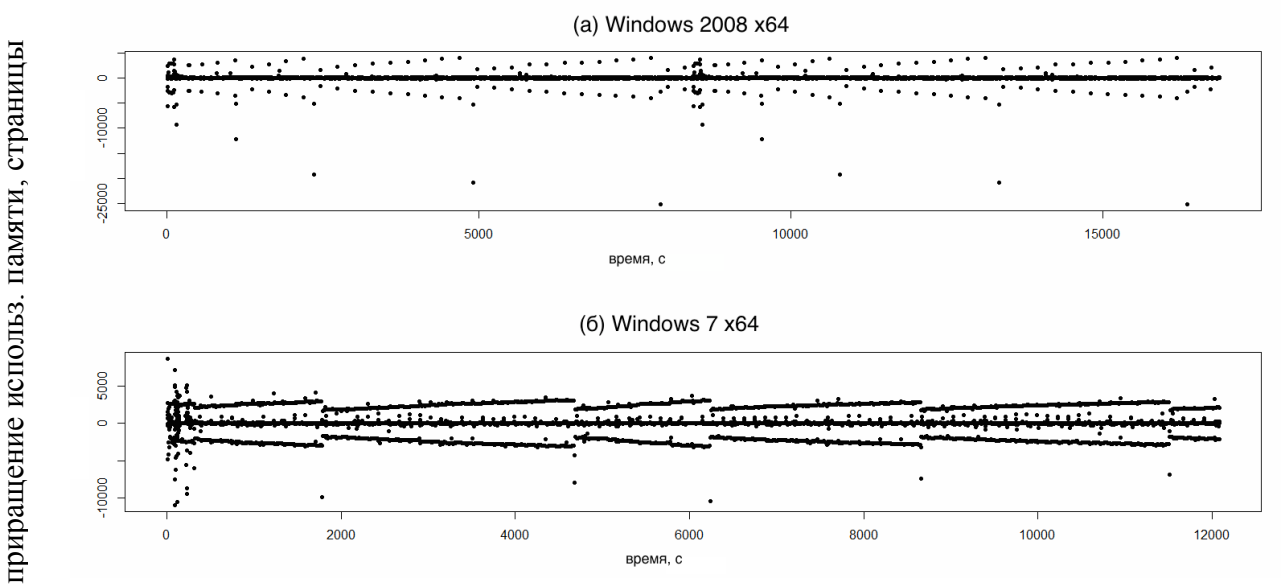

Рис. 3. График приращения используемой памяти (рабочего набора) в ОС Windows 2008 x64 (а) и Windows 7 x64 (б)

Переломы графиков на рис.2(а) и рис.2(б) и на рис.3(а) и рис.3(б) происходят в разные моменты времени на одинаковой нагрузке. Это визуально подтверждает результат анализа данных, выполненный с помощью указанных критериев: ни типовые схемы потребления памяти, ни типовые схемы приращения объема потребления памяти не принадлежат к одному распределению для разных операционных систем даже близких семейств. Из этого следует, что, даже полагаясь только на виртуализационные счетчики, оценку размера рабочего набора необходимо проводить отдельно для всех основных версий гостевых ОС. Кроме того, стандартными средствами не получилось выделить корреляцию между виртуализационной статистикой и изменением объема рабочего набора. Таким образом, виртуализационные счетчики, как выяснилось, сами по себе не могут обеспечить получение адекватной оценки. Поэтому было принято решение строить оценку на базе гостевых счетчиков, несмотря на указанные выше сложности этого подхода.

Глава 4 концентрируется на детальном рассмотрении оценки размера рабочего набора на основании гостевых счетчиков ОС Microsoft Windows 7 и Windows 8. Путём исключения нерелевантных показателей был сформирован список из наиболее значимых счетчиков. Ими оказались следующие параметры (названия соответствуют наименованиям статистических показателей в ОС Windows):

- Commit total – число страниц закрепленной физической памяти;
- Working set size – размер рабочего набора;
- Physical memory usage – количество используемой физической памяти;
- Page file usage – объем использования страничного файла.

В ходе исследований (рис.4) выяснилось, что: (1) размер рабочего набора гостевой операционной системы Windows 8 фактически равен Commit Total за вычетом системных кэшей, (2) оценка справедлива и для Windows 7 x64 и для Windows 8 x64 систем, что в некоторой степени упрощает разработку и внедрение системы управления памятью. То есть можно сформулировать эмпирическую (инженерную) зависимость для оценки рабочего набора (WSS)

$$WSS = CommitTotal - BalloonSize \qquad (1)$$

где BalloonSize – это размер фиктивно занятой физической памяти (ballooned memory).

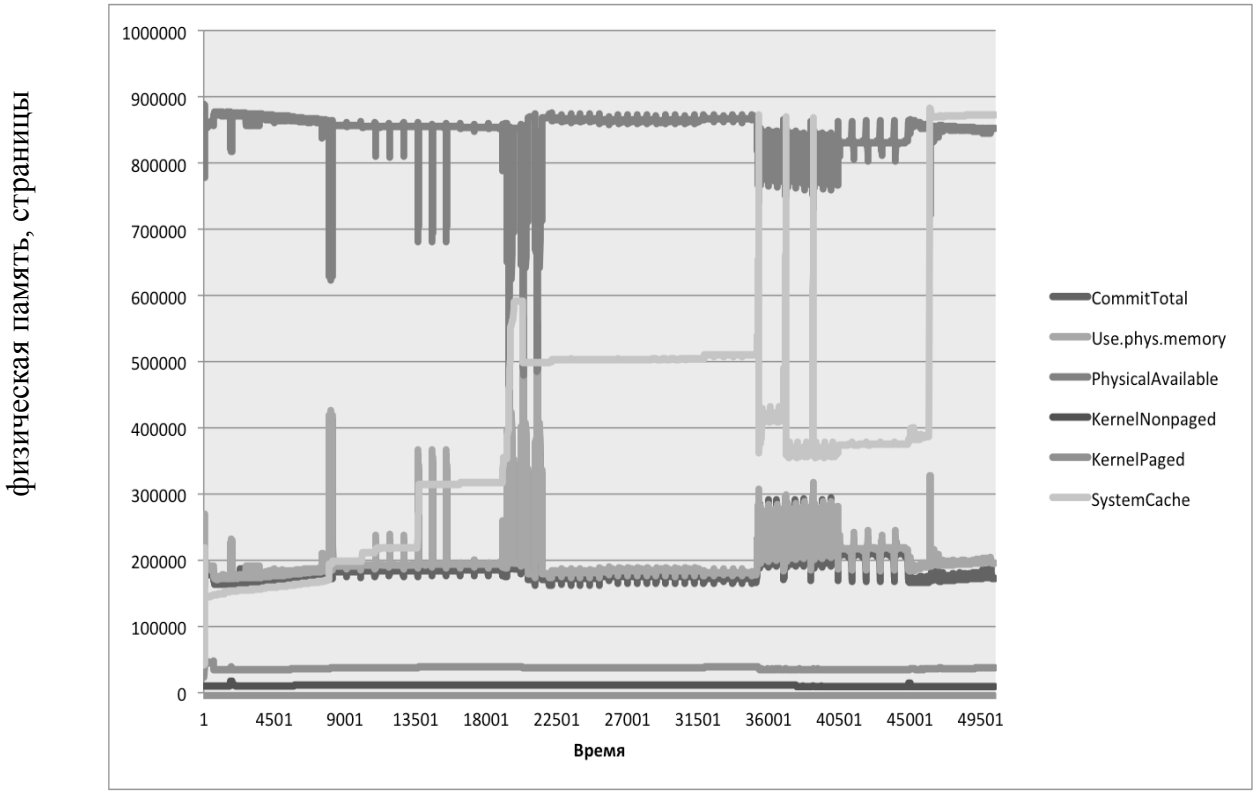

Рис. 4. Гостевые счетчики Windows 7

Дальнейшие практические исследования показали, что эта оценка не обладает достаточным уровнем точности и ей требуется корректировка на размер кэшей гостевой операционной системы. Такая корректировка была предпринята в соответствии с формулами:

$$WSS = CommitTotal - BalloonSize + \sigma * PhysicalTotal \qquad (2)$$

где σ - это эмпирический коэффициент. Подбор коэффициента осуществлялся на известном тесте PCMark. На рис.5 представлен график, где по вертикали представлен балл теста (чем больше, тем лучше), а по горизонтали отложена величина коэффициента σ (на графике коэффициент представлен в процентах, в то время как в формуле это величина от 0 до 1;

преобразование от единиц в формуле к единицам в графике происходит через умножение на 100).

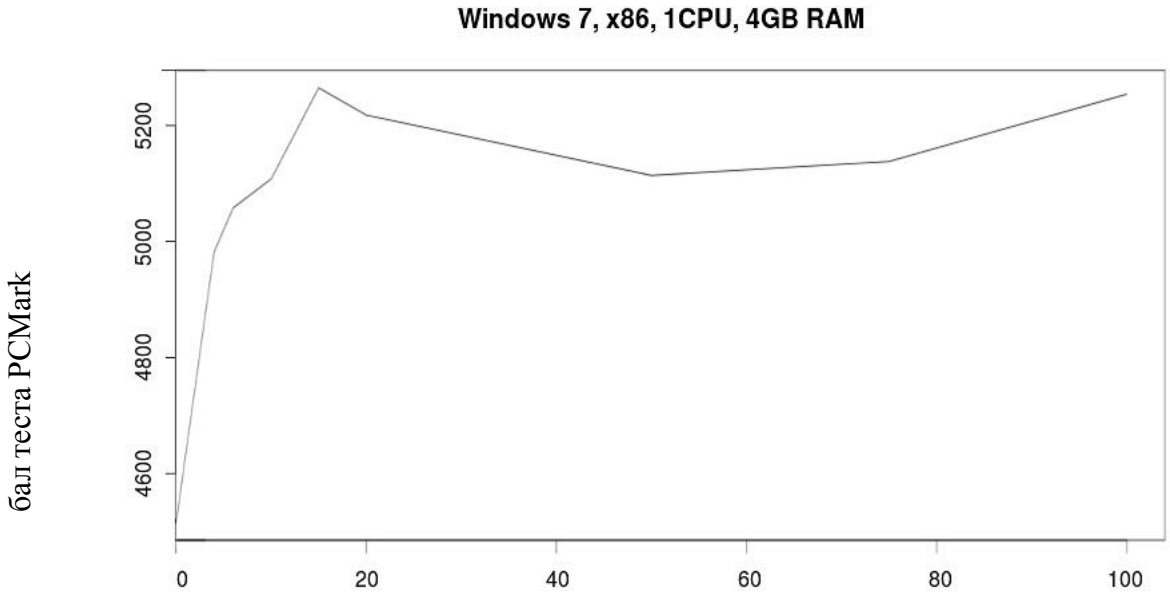

Рис. 5. Производительность в зависимости от эмпирического коэффициента кэша

Как видно на рис.5, корректировка формулы на размер кэша в соответствии с неким эмпирическим коэффициентом повысила производительность системы. Но при этом сохранялось существенное падение производительности при интенсивной дисковой нагрузке. Как вывод из исследований, предпринятых в главе 4, для устранения влияния эмпирических допущений, повышения уровня производительности и улучшения адаптивных свойств оценки было принято решение уточнять оценку методами обучения с подкреплением, где штраф определялся бы на основе виртуализационных счетчиков.

В **главе 5** описывается алгоритм корректировки оценки на основании виртуализационной статистики методами обучения с подкреплением, предложенный автором. Основная идея алгоритма – редактировать размер фиктивно занимаемой физической памяти (размер balloon) в соответствии с формулой

$$BalloonSize = PhysicalTotal - WSS - gap \quad (3)$$

где WSS – это оценка рабочего набора по формуле 1 (в качестве размера фиктивно занимаемой физической памяти (balloon) используется значение из предыдущего расчета), PhysicalTotal – это назначенный объем памяти гостевой ОС, gap – это некая величина, смягчающая погрешность

оценки, зазор. Необходимость введения зазора объясняется тем, что завысить объем фиктивно занятой памяти значительно опаснее, чем занизить. Иными словами, положительная ошибка существенно хуже отрицательной: в случае завышенного размера возможно резкое падение производительности гостевой ОС, а в случае заниженного - ожидается пропорциональное снижение сэкономленного ресурса (и соответственно снижение плотности размещения виртуальных сред). Автором были предложены к учету следующие факторы для формирования величины зазора (gap):

1. Колебание объема фиктивно занимаемой физической памяти (ballooned memory) негативно сказывается на производительности. Нужно сохранять объем постоянным, пока вознаграждение от альтернативного зазора (gap) не превысит проигрыш от изменения объема.
2. Величина зазора и скорость его изменения не должны выходить за определенные границы, иначе возрастают накладные расходы на содержание системы.
3. Возросшее число виртуализационных страничных промахов и операций ввода/вывода могут служить показателями неэффективности текущего зазора.

В соответствие с третьим предположением была сформирована следующая формула для расчета штрафа на изменение зазора

$$fine = (\Delta io - io\_thres) * io\_fine + (\Delta pgin - pgin\_thres) * pgin\_fine \quad (4)$$

где Δio , $\Delta pgin$ – это изменение числа операций ввода/вывода/числа виртуализационных страничных промахов за оцениваемый интервал, io_thres , $pgin\_thres$ - это пороговое значение, приемлемый уровень возрастания числа операций ввода/вывода / числа виртуализационных страничных промахов, io_fine , $pgin\_fine$ – штраф за возрастание/приз за снижение.

Обучение с подкреплением обеспечивает выбор лучшего из вариантов приращения gap в (100 – ε) случаев. В (ε) случаях приращение зазора выбирается произвольно.

Псевдокод алгоритма выбора зазора выглядит как

```
(1)  gap_chng[GAP_TO_IDX(prev_gap_change)] -= fine();
(2)  if( greedy_degree && (++iter)%greedy_degree == 0 )
(3)      gap = ALIGN_GAP(gap + rand64()%GAP_CHANGE_MAX);
(4)  else
(5)  {
(6)      gap = ALIGN_GAP( ws - prev_ws + prev_gap );
(7)      gap_change = gap – prev_gap;
```

```
(8)          tgt_i = GAP_TO_IDX(gap_change); //target index
(9)         for( i = 0; i < GAP_CHANGE_ARRAY; i++ )
(10)              if( gap_chng[i] – gap_chng[tgt_i] > gap_chng_threshold )
(11)                  tgt_i = i;
(12)         if( tgt_i !=  GAP_TO_IDX(gap_change) )
(13)             gap = ALIGN_GAP( prev_gap + gap_chng[tgt_i] );
(14)     }
(15)     prev_gap_change = prev_gap - gap;
(16)     prev_gap = gap;
(17)     prev_ws = ws;
```

В алгоритме многое зависит от значения ε - степени «жадности» алгоритма (в псевдокоде *greedy_degree)*. Чем больше ε, тем быстрее алгоритм обучается, но тем хуже его производительность на время обучения. Значения ε подбирались экспериментальным путем на наборе тестов, в которые включались

- простой/занятый цикл (busy loop);
- тесты с дисковой активностью;
- тесты на переключение процессов/потоков;
- тесты с сетевой активностью;
- тесты-потребители памяти;
- инсталляция MS Office;
- симуляция работы в Office приложениях;
- тесты на 3D графику;
- PCMark.

Этот набор тестов оказывается достаточно полным с точки зрения различных сценариев использования виртуальных машин, что следует из нашего опыта работы с хостинг-провайдерами.

Программная реализация существует для двух виртуализационных систем: Parallels Server и qemu-kvm. На рис. 6 представлена архитектура реализации для Parallels Server. Изменения вносились в изолированные модули системы, написанные на языке C. В агент фиктивного занятия памяти (balloon) добавлен сбор гостевой статистики. В него же внесен расчет размера рабочего набора гостевой ОС. Подсчитанный размер через виртуализационный интерфейс VirtIO передается в гипервизор, где раз в секунду корректируется объем фиктивно занятой памяти на основании полученной оценки и виртуализационной статистики, присутствующей в модуле. В случае изменения объема фиктивно занятой памяти агент на стороне гостевой ОС информируется об этом гипервизором через VirtIO

интерфейс. Информация об изменениях передается также в гипервизор, который будет ее учитывать при распределении памяти между виртуальными машинами. Со стороны интерфейса командной строки внесена поддержка перевода системы на «автоматический» режим, представленный в данной работе.

Рис. 6. Реализация алгоритма в виртуализационной системе Parallels Server

Реализация в qemu-kvm в части модификации агента фиктивного занятия физической памяти архитектурно не отличается от реализации в Parallels Server. Отличия возникают из-за разделения виртуализационной логики между компонентом kvm и qemu: kvm не работает напрямую с агентом фиктивного занятия памяти, а qemu изолирован от виртуализационной статистики. Предложенная реализация добавляет сбор виртуализационной статистики в kvm и передачу ее в qemu. Модуль qemu расширен для передачи через HMP интерфейс полученной от kvm статистики и полученного от агента через VirtIO интерфейс значения размера рабочего набора. Алгоритм коррекции осуществляется во внешнем скрипте, написанном на языке Python. Этот скрипт на основании виртуализационной статистики и оценки размера рабочего набора определяет рекомендуемый размер фиктивно занятой памяти и через HMP интерфейс передает его qemu для применения.

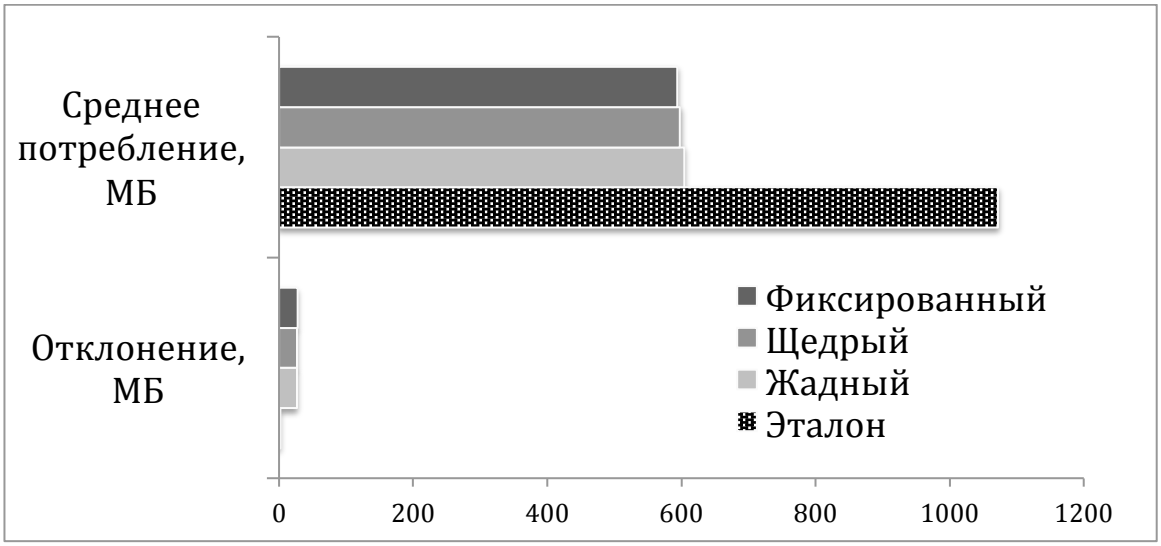

Рис. 7. Уровень потребления памяти

Тесты показали, что при использовании алгоритма интеллектуального управления объемом фиктивно занятой памяти существенно снизилось потребление памяти (рис. 7). Здесь эталон – виртуальная машина без фиктивного занятия памяти (без ballooning). «Жадный» алгоритм - это ε=2, «щедрый» алгоритм - ε=20, а фиксированный алгоритм – это постоянное значение зазора без адаптации (gap=64MB). Примечательно, что для разных параметров алгоритма и среднее потребление памяти и ее отклонения примерно одинаковы. Это обусловлено длительностью теста и разнообразием его нагрузок. Объем «сэкономленной» памяти на скромной конфигурации (1GB RAM) составил 40%, что является хорошим результатом. При этом снижение производительности незначительное, как видно из результатов тестов, представленных в таблице 2

Таблица 2. Тестирование производительности с разными параметрами алгоритма

	Эталон	Жадный		Щедрый		Фиксированный	
Тест	Scores	Scores	vs #1	Scores	vs #1	Scores	vs #1
system_busyloop	2523929549	2443138253	-3%	2592330445	+3%	2551971520	+1%
system_syscall	1538780	1534676	0%	1595706	+4%	1557842	+1%
system_exchandler	128072	126602	-1%	135751	+6%	133692	+4%
cpu_cpuid-fast	2222113	2197019	-1%	2250789	+1%	2183735	-2%
cpu_cpuid-slow	1133195	1166001	+3%	1209299	+7%	1187391	+5%
cpu_cpuid-touch	799583	879250	+10%	897229	+12%	886960	+11%
system_tlbflush	21371	21236	-1%	19417	-9%	21874	+2%
system_registry	93171	90140	-3%	71582	-23%	89861	-4%
system_messenger	145976	146663	+0%	123421	-15%	145189	-1%
time_msleep	529,37612	534,87959	+1%	554,34478	+5%	524,48431	-1%
system_guest2host	133744	137208	+3%	130490	-2%	134038	+0%
system_guest2host	108617	110439	2%	108049	-1%	107739	-1%
process_exec	364,449854	364,199162	0%	360,899238	-1%	364,11059	0%
process_ctxswitch	240968	242521	+1%	238883	-1%	238816	-1%
thread_create	9951,74707	9821,30468	-1%	9701,72929	-3%	9690,58476	-3%
thread_ctxswitch(1)	504603	505509	+0%	513614	+2%	518783	+3%

thread_ctxswitch(2)	1151361	1170165	+2%	1152244	+0%	1151633	+0%
ext_cpu-irql	54675364	53702585	-2%	54325834	-1%	51188779	-6%
ext_cpu-cli	316306682	319805190	+1%	322145331	+2%	290534349	-8%
ext_cpu-lock	120723602	120906822	+0%	120081158	-1%	115915312	-4%
ext_cpu-invlpg	15505879	15448545	0%	15518289	+0%	14845673	-4%
ext_cpu-rdtsc	98556416	98003154	-1%	97780347	-1%	95416346	-3%
ext_cpu-mmio	434157	438238	+1%	445973	+3%	420119	-3%
ext_cpu-rdmsr	1462940	1465157	+0%	1477567	+1%	1360237	-7%
ext_cpu-inb	1299068	1294616	0%	1308078	+1%	1203080	-7%
ext_cpu-ins	509863	517752	+2%	520224	+2%	488015	-4%
mem_virtalloc-64M	375946	365414	-3%	373817	-1%	368146	-2%
mem_virtalloc-4K	536729	548355	+2%	558552	+4%	536443	0%
mem_read	7794,13047	7776,06679	0%	7807,76061	+0%	7809,45302	+0%
mem_write	6859,48398	6887,64355	+0%	6953,21279	+1%	6065,83115	-12%
mem_pgfault_read	567050	573753	+1%	561435	-1%	548744	-3%
mem_pgfault_write0	606331	607039	+0%	595421	-2%	580814	-4%
mem_copy	6464,35	6525,02334	+1%	6510,45859	+1%	5980,72119	-7%
mem_copy-L1cache	23135	22968	-1%	23208	+0%	22522	-3%

В первом столбце представлен эталон – результаты тестирования без фиктивного занятия памяти (без ballooning). В втором столбце представлен результат «жадного» алгоритма (ε=2). В третьем столбце результат «щедрого» алгоритма (ε=20). В четвертом столбце результат алгоритма с фиксированным размером зазора (gap=64MB). Цифрами в «раскрашенных» столбцах указаны потери производительности по отношению к эталону (в процентах). Светло-серый означает паритет, серый – это прибавка производительности, черный с белыми цифрами – это падение производительности. Из этих результатов видно, что жадный алгоритм практически не теряет производительности. В случае щедрого алгоритма возникают как внезапные улучшения, так и ухудшения производительности (и отклонения в результатах теста существенные). В случае фиксированного размера зазора имеем стабильную потерю производительности, что доказывает необходимость его динамической адаптации.

При обучение с подкреплением важно найти компромисс между исследованием неизученных областей и применением имеющихся знаний. В какой-то момент многие из алгоритмов прекращают обучение с целью увеличения прибыли. Однако, в данном случае время жизни виртуальной машины оказывается незначительным (в соответствии со статистикой от хостинг-провайдеров), поэтому обучение небольшими темпами продолжается непрестанно.

В **заключении** приведены основные результаты исследования.

Личный вклад автора

Автором проведено исследование предметной области, выполнен анализ гостевых и виртуализационных счетчиков, разработана оценка размера рабочего набора на основании виртуализационных счетчиков, сформулирован подход по корректировке размера фиктивно занятой памяти

на основании штрафов от виртуализационной статистики, написан алгоритм корректировки и проведено экспериментальное исследование эффективности его работы.

Разработка оценки на основании гостевых счетчиков и проверка гомогенности виртуализационных параметров проводилась совместно с Маркеевой Л.Б.

Основные результаты и выводы диссертации

1. Проведен анализ существующих проблем и имеющихся решений в области управления памятью в гипервизорах.
2. Выполнен сравнительный анализ виртуализационной и гостевой статистики с точки зрения их применимости для создания оценки рабочего набора.
3. Предложена математическая модель оценки размера рабочего набора операционных систем семейства Windows на основании гостевых счетчиков.
4. Разработан метод корректировки оценки размера рабочего набора на основании виртуализационной статистики методами обучения с подкреплением.
5. Создан комплекс программ, реализующий предложенный автором алгоритм управления объемом фиктивно занимаемой памяти на основании оценки размера рабочего набора, скорректированной методом обучения с подкрепления.
6. Проведен ряд экспериментов на реальных программных комплексах и на модельных тестах для настройки параметров модели.
7. Экспериментально обоснована эффективность данного инженерного решения как повышающего плотность размещения виртуальных машин при сохранении качества обслуживания и приемлемого уровня накладных расходов.
8. Реализация внедрена в программный комплекс Parallels Server.
9. Предложенный алгоритм перенесен на систему с открытым исходным кодом Linux KVM.

Список публикаций по теме диссертации

1. Маркеева, Л. Б. Проверка гипотезы однородности виртуализационных событий, порожденных различными операционными системами / Л.Б. Маркеева, А. Л. Мелехова, А. Г. Тормасов // Труды МФТИ. — 2014. — Т. 6. — С. 57–64.
2. Бондарь, А. О. Алгоритмы решения задачи динамического управления

питанием в облачной системе / А. О. Бондарь, Н. В. Ефанов, А. Л. Мелехова // Программная инженерия. — М. — 2015. — №4. — С. 20–30.

3. Мелехова(Воробьева), А.Л. Стратегия виртуализации физической памяти, применяемые в виртуальных машинах. / А. Л. Мелехова (Воробьева) // Процессы и методы обработки информации. сб. науч. тр. — М.: МФТИ, 2009.
4. Бондарь, А.О. Энергосбережение изнутри: что в действительности могут измерить профилировщики / А. О. Бондарь, Д. К. Карпов, А. Л. Мелехова // RSDN Magazine, М.: 2013
5. Expansion of virtualized physical memory of virtual machine: патент US8135899 B1 США, МПК G 06 F12/00 / N. N. Dobrovolskiy, A. A. Omelyanchuk, A. B. Koryakin, A. L. Melekhova(Vorobyova), A.G. Tormasov, S.M. Beloussov; заявитель и патентообладатель Parallels IP Holdings GmbH — № US 13/084,262; заявл. 11.04.2011; опубл. 13.03.2012.
6. Мелехова(Воробьева), А.Л. Методика определения размера рабочего набора виртуальной машины. / А. Л. Мелехова(Воробьева) // Математическое моделирование информационных систем. сб. науч. тр. — М.: МФТИ, 2012.
7. Мелехова(Воробьева), А.Л. Управление памятью в гипервизоре. Все о виртуализации памяти в Parallels. / А. Л. Мелехова(Воробьева) // Разработка высоконагруженных систем. сб. тр. конф. Highload++, Москва, 3–4 октября 2011. — М.: Изд-во Олега Бунина, 2012.
8. Melekhova, A. Machine Learning in Virtualization: Estimate a Virtual Machine's Working Set Size / A. Melekhova // Proceedings on 2013 IEEE Sixth International Conference on Cloud Computing, CLOUD 2013.
9. Markeeva, L. Estimating working set size by guest OS performance counters means / A. Melekhova, L. Markeeva // Proceedings CLOUD COMPUTING 2015, The Sixth Conference on Cloud Computing, GRIDs, and Virtualization. — 2015. — С. 48.
10. Kudinova, M. CPU prediction models // M. Kudinova, A. Melekhova, A. Verinov // Proceedings of the 11th Central & Eastern European Software Engineering Conference in Russia. — 2015. — (готовится к печати)

Научное издание

Мелехова Анна Леонидовна

УПРАВЛЕНИЕ ФИЗИЧЕСКОЙ ПАМЯТЬЮ ВИРТУАЛЬНОЙ МАШИНЫ

АВТОРЕФЕРАТ
диссертации на соискание учёной степени
кандидата технических наук

Подписано в печать .2015. Формат 60x90/16.
Усл печ л 1.0. Тираж 80 экз. Заказ No .
Московский физико-технический институт
(государственный университет)

Печать на аппарате Rex-Rotary Copy Printer 1280. НИЧ МФТИ.

141700, г Долгопрудный Московской обл, Институтский пер, 9,
тел.: (095) 4088430, факс (095) 5766582