На правах рукописи

Емельянов Павел Владимирович

Математическое моделирование процессов управления потреблением памяти в многопользовательских средах и системах виртуализации

Специальность 05.13.18 —
Математическое моделирование, численные методы
и комплексы программ

АВТОРЕФЕРАТ
диссертации на соискание ученой степени
кандидата физико-математических наук

Москва — 2008

Работа выполнена на кафедре информатики Московского физико-технического института (государственного университета).

Научный руководитель: кандидат физико-математических наук
ТОРМАСОВ Александр Геннадьевич

Официальные оппоненты: доктор технических наук
профессор
СЕМЕНИХИН Сергей Владимирович

кандидат технических наук
КОНЬКОВ Александр Константинович

Ведущая организация: Институт Автоматизации
Проектирования РАН

Защита состоится «16» октября 2008 года в 10:20 часов на заседании диссертационного совета Д 212.156.05 при Московском физико-техническом институте (государственном университете) по адресу: 141700, Московская область, г. Долгопрудный, Институтский пер., д. 9, ауд. 903 КПМ.

С диссертацией можно ознакомиться в библиотеке Московского физико-технического института (государственного университета).

Автореферат разослан «15» сентября 2008 года.

Учёный секретарь диссертационного совета Д 212.156.05,
кандидат физико-математических наук Федько О. С.

Общая характеристика работы

Актуальность работы

В диссертационной работе рассматривается задача управления потреблением физической памяти группами процессов с учетом следующих требований: подсчет объема используемой процессами физической памяти, ограничение процессов в объеме используемой памяти, гарантированное выделение определенного объема памяти процессам, корректная работа в случае присутствия участков разделяемой памяти. Задача управления ресурсами и родственная ей задача обеспечения гарантированных параметров качества обслуживания на уровне операционной системы весьма актуальны для современных вычислительных систем коллективного пользования (например, для центров хранения и обработки данных). На сегодняшний день, в эпоху большой популярности систем виртуализации и изоляции компонентов вычислительных систем, задача разделения ресурсов стоит особенно остро.

Операционная система с поддержкой виртуализации дает возможность запускать на одном компьютере несколько виртуальных машин или серверов[1], позволяя, таким образом, более экономно и полно использовать имеющиеся вычислительные мощности.

Уже больше года компания Intel поставляет все свои настольные процессоры со встроенной технологией для аппаратной поддержки виртуализации под названием VT™, которая существенно повысила производительность существующих программных реализаций этой технологии. В то же самое время уже создан огромное количество различных программных комплексов, работающих с поддержкой этой технологии, например, VMWare, Parallels, KVM и т. д. Если раньше основными пользователями этой технологии были компании с большим парком техники, то сейчас

[1] В англоязычных источниках популярностью пользуется понятие «контейнера» для определения виртуального сервера

она доступна на каждом персональном компьютере.

Вопросы распределения ресурсов, их планирования и гарантированного выделения являются одними из самых важных в подобного рода системах, ведь от решения этих вопросов напрямую зависит не только качество обслуживания пользователей этих систем, но и вопросы безопасности. Один из основных ресурсов вычислительной системы – физическая память – не исключение.

Также следует заметить, что вопрос разделения и гарантированного выделения ресурсов возникает не только в системах виртуализации, но и в самих многопользовательских операционных системах (ОС). Например ни в Windows, ни в Linux, которые позволяют огромному количеству пользователей одновременно работать и выполнять свои приложения, до сих пор нет удовлетворительных средств для управления ресурсами: памятью, процессором, диском и др. Для отдельных ресурсов существуют средства контроля за отдельными процессами или нитями, для некоторых – на уровне отдельных пользователей, но в общем случае нет никаких средств контролировать произвольную группу процессов, которой, в конечном счете, является виртуальная машина. Тонкие различия между различными ресурсами ОС и их использованием не позволяет эффективно решить эти задачи с помощью единого подхода.

Цель работы

Целью работы является построение математических моделей, описывающих процессы потребления физической памяти в многопользовательских и виртуализационных системах, разработка новых алгоритмов и комплекса программ по управлению памятью в операционной системе, который должен обладать следующими свойствами:

- обеспечивать изоляцию виртуальных сред друг от друга;
- ограничивать виртуальные среды в потреблении физической памяти;

- обеспечивать требуемый уровень сервиса для виртуальных машин.

Для достижения поставленных целей формулируются основные требования и исследуется соответствие им существующих моделей и подходов. Среди рассмотренных моделей выделяется одна, которая наиболее полно, по сравнению с остальными, соответствует поставленным требованиям. Затем анализируются ее недостатки и разрабатываются математические модели и алгоритмы, призванные дополнить ее и довести до полного соответствия поставленным требованиям.

Научная новизна

В работе предложена стратегия управления памятью, представляющая собой развитие нескольких ранее опубликованных подходов и ряд оригинальных решений.

1. Предложена новая математическая модель предоставления гарантий выделения физической памяти группе процессов, позволяющая управлять распределением памяти в современных многопользовательских операционных системах и системах с поддержкой виртуализации. Показана принципиальная применимость предложенной модели к другим типам ресурсов.

2. Предложена математическая модель, позволяющая существенно повысить производительность алгоритмов, обеспечивающих подсчет количества используемой физической памяти. С использованием предложенной модели разработан новый алгоритм, позволяющий ограничивать группы процессов по объему используемой памяти, который является расширением уже существующего алгоритма. Вычислительные эксперименты подтвердили более высокую производительность нового алгоритма.

3. Предложена модель «виртуальной выгрузки страниц на диск», позволяющая создавать виртуальные среды, не отличающиеся по принципам администрирования и поведению от физических.

Научная и практическая ценность работы

Известные на сегодня стратегии решения задач управления памятью в системах виртуализации обладают различными недостатками и часто вообще неудовлетворительны, поэтому в этой области ведутся интенсивные исследования.

Разработанная в работе модель гарантированного выделения ресурсов оказалась пригодной не только в подсистеме управления памятью, но и в других подсистемах разделения ресурсов ОС – например, в подсистеме планирования выполнения задач и разделения процессорного времени.

Часть реализованного программного комплекса была внедрена автором совместно со специалистами из компаний IBM и Google в основное ядро Unix-подобной операционной системы Linux[2], которая является одной из наиболее популярных операционных систем и установлена как на домашних компьютерах, так и на крупных серверах по всему миру.

Кроме того, реализация всех описанных моделей была внедрена в продукты компании Parallels Virtuozzo Containers[3] и OpenVZ[4], представляющие собой виртуализационные решения на базе ядра Linux. Их использование показало, что модели представляют не только теоретический интерес, но и успешно решают поставленные перед ними практические задачи, а именно – позволяют контролировать потребление памяти виртуальной машиной.

[2] Исходные тексты доступны по адресу http://git.kernel.org/?p=linux/kernel/git/torvalds/linux-2.6.git

[3] http://www.parallels.com/en/virtuozzo/

[4] http://openvz.org

Методы исследования

Для построения моделей в работе были использованы теория алгоритмов, методы системного программирования и теории операционных систем, методы линейной алгебры и теории вероятности.

Для проведения численных исследований предложенные модели реализованы как составная часть ядра операционной системы Linux, и проведены эксперименты с использованием модельных и реальных приложений.

Аппробация результатов работы

Все модели, подходы и результаты, изложенные в работе опубликованы, в том числе в изданиях, рекомендованных ВАК РФ [1, 2, 3].

Результаты диссертации доложены на XLVII и XLIX научных конференциях Московского физико-технического института (Москва-Долгопрудный, 2004 – 2006) и на международных конференциях Linux-разработчиков «Linux Symposium» (Оттава, Канада, 2007) и «LinuxConf Europe» (Кэмбридж, Великобритания, 2007). Доложенные материалы получили одобрение специалистов, включая ключевых разработчиков ядра Linux. Так же модель предоставления гарантий в сокращенной форме опубликована в документации к проекту OpenVZ[5].

Реализации алгоритмов, внедренные в продукты Parallels Virtuozzo Containers и OpenVZ, установлены на тысячах серверов по всему миру и обеспечивают корректную работу и изоляцию почти полумиллиона виртуальных серверов.

Структура и объем работы

Диссертация состоит из введения, четырех глав, заключения и списка использованных источников из 105 наименований. Диссертация изложена

[5] http://wiki.openvz.org/Containers/Guarantees_for_resources

на 101 странице и включает 20 иллюстраций.

Содержание работы

Во введении описаны текущее состояние исследований в области систем виртуализации, их практические приложения, отмечена популярность этих систем, обоснована актуальность темы диссертации, научная и практическая ценность работы, ее новизна. Также во введении кратко изложены содержание и структура диссертации.

В первой главе приведен обзор существующих моделей управления памятью, продемонстрирована неприменимость их в чистом виде в системах с поддержкой виртуализации и приведен анализ того, как должна выглядеть такая подсистема в современной ОС.

Выявлены четыре основных требования к подсистеме управления памятью в операционной системе с поддержкой виртуализации:

- ограничение группы процессов в использовании физической памяти;
- гарантированное выделения памяти заранее заданного объема;
- учет возможного существования участков разделяемой памяти между группами процессов;
- учет существования участков разделяемой памяти между процессами одной группы.

Кроме того, показана принципиальная выгода от использования подобной подсистемы в обычных многопользовательских средах.

В процессе разбора существующих моделей и реализаций особое внимание уделено т. н. модели «счетоводов»[6]. Несмотря на то, что эта модель была разработана несколько лет назад, она до сих пор наиболее полно,

[6]Оригинальное название модели англоязычное – “beancounters”

по сравнению с остальными существующими разработками, соответствует поставленным требованиям. Показано, что несмотря на это, в модели присутствуют ряд недостатков.

В рамках обзора рассмотрена эволюция подсистем управления памятью в операционных системах, дано необходимое описание технологии «виртуальной памяти» и архитектур построенных на ее основе подсистем управления памятью в современных операционных системах общего назначения.

В начале главы приведена необходимая терминология.

Во второй главе описан подход, который позволил существенно улучшить производительность используемого в выбранной модели «счетоводов» алгоритма по подсчету количества используемой памяти.

Подсчет этого количества необходим для выполнения первого из поставленных требований – ограничения групп в использовании памяти. Задача ограничения выглядит следующим образом.

Пусть в системе с n группами вектор $\vec{r} = (r_0, \ldots, r_{n-1})$ показывает количество страниц, которое используется группами, а вектор $\vec{l} = (l_0, \ldots, l_{n-1})$ – это набор ограничений, выставленный администратором системы. Тогда в соответствии с поставленным требованием система управления памятью при запросах на выделение Δr_i единиц памяти для группы i должна отказывать в выделении тогда и только тогда, когда $r_i + \Delta r_i > l_i$ (после этого необходимо пересчитать компоненты $\vec{r}$).

В главе описан центральный принцип принятой *во всех* современных операционных системах модели выделения памяти, которая в упрощенном виде состоит из двух частей[7]:

- Сначала процесс запрашивает у ядра т. н. «регионы», в рамках которых ему будет разрешено выделять физическую память. Этот запрос

[7] См. например *Love R.* Linux Kernel Development. — Novell Press, 2005. — Ch. 14: The Process Address Space.

осуществляется процессом явно с помощью системного вызова `mmap`. На данном этапе ОС не выделяет процессу физической памяти.

- Затем процесс запрашивает у ядра физическую память в рамках выделенных ранее регионов. Этот запрос производится неявно – процесс просто обращается по некоторому виртуальному адресу в рамках созданного региона (например, записывая туда данные). При таком обращении генерируется аппаратное исключение типа «страничный отказ», которое перехватывается ядром операционной системы. Результатом выполнения этого запроса ядром является выделенный процессу физический участок памяти.

Важно заметить, что в процессе обработки этих запросов ядро операционной системы и процессор оперируют участками памяти фиксированного размера – страницами памяти.

Нетрудно видеть, что $\vec{r}$ может измениться только в результате выполнения запроса *второго* типа, поэтому рассматриваемый алгоритм «счетоводов» перехватывает все запросы этого типа и подсчитывает количество физических страниц, выделенных группе процессов по этим запросам. При этом алгоритм использует точные, но весьма громоздкие счетчики для корректной работы с участками разделяемой памяти. Как показали измерения, эта процедура заметно снизила производительность выполняющихся процессов – память по второму запросу стала выделяться до 20% медленнее, что является серьезным недостатком алгоритма. Для преодоления указанного недостатка, предложено обходить дополнительные вычисления когда это возможно.

Предложенная модель улучшения алгоритма выглядит следующим образом. Рассмотрим группу процессов с некоторым множеством из m_i регионов отдельных процессов с длинами M_j, $j = \overline{0, m_i - 1}$. Обозначим также за R_j количество физических страниц, выделенных внутри региона j.

Очевидно, что

$$R_j \leq M_j, \forall j. \tag{1}$$

На этом соотношении построено предлагаемое изменение алгоритма – для каждого региона j *введен новый булевый признак* δ_j, который показывает, нужно ли выполнять точные (и медленные) вычисления во время обработки запросов второго типа. Также для региона j построена оценочная величина

$$E_j = \delta_j R_j + (1 - \delta_j) M_j, \tag{2}$$

которая используется для проверки того, не превысила ли группа процессов свои ограничения по использованию памяти. Это возможно, благодаря тому, что

$$R_j \leq E_j \leq M_j. \tag{3}$$

По мере приближения оценки к ограничению требуется производить уточнение этой оценки, пересчитывая величины R_j и δ_j для отдельных регионов.

В главе вычислено математическое ожидание времени, которое уйдет у ядра для обработки одного запроса второго типа (τ) в ситуации, когда в системе присутствуют m_s регионов, для которых $\delta_i = 1$ и выяснено, при каких условиях это время достигает своего минимума.

Искомая величина задается выражением

$$E\tau = \sum_j p_j \tau_j, \tag{4}$$

где p_j – вероятность того, что запрос придет в регион j, а τ_j – время выполнения запроса в этом регионе.

Требуется решить задачу поиска набора регионов $(j_1, \ldots, j_{m_s})$, для которых $\delta_{j_i} = 1$, таких, что

$$E\tau \to min. \tag{5}$$

Легко показать, что для τ_j справедливо

$$\tau_j = \delta_j T + (1 - \delta_j)t, \tag{6}$$

где T – время медленных вычислений, а t – время быстрых, для которых справедливо $T \gg t$, а для вероятности p_j имеет место

$$p_j \sim (M_j - R_j)\, A_j, \tag{7}$$

где $A - j$ – активность использования региона. Под активностью региона в главе понимается величина, характеризующая частоту обращений процессов к страницам в регионе. Точное значение этой величины не известно для произвольного приложения, и в главе приведен ряд подходов к ее вычислениям. Например, рассмотрены регионы с одинаковой активностью ($Aj = 1, \forall j$) и предложена оценка активности регионов по относительному уровню их использования ($A_j = R_j M_j^{-1}$).

Опуская константу в выражении (7), обозначим $\alpha_j = (M_j - R_j)\, A_j$. Таким образом для $E\tau$ имеет место

$$E\tau = \sum_{j:\delta_j=1} \alpha_j T + \sum_{j:\delta_j=0} \alpha_j t. \tag{8}$$

Отсюда индуктивно несложно доказать, что решением задачи (5) является такой набор регионов, для которого вектор $(\alpha_{s1}, \ldots, \alpha_{sm_s})$ состоит из *наименьших* значений α_j. Таким образом, на каждом шаге, когда требуется найти очередной регион s, для которого нужно установить $\delta_s = 1$ нужно выбрать регион с минимальных из существующих α_s.

Реализация алгоритма потребовала также перехвата обработки и запросов *первого* типа, однако код, отвечающий за обработку этих запросов выполняется не часто и сам по себе относительно долог. В отличие от запросов второго типа, код обработки которых выполняется гораздо чаще, чем первый и настолько хорошо оптимизирован, что любые внедрения в него могут привести к заметным потерям в производительности.

В главе доказана теорема

Теорема 1 (об одновременном достижении оценкой точного значения и уровнем потребления своего ограничения)**.** *Невозможна ситуация, при которой для части регионов все еще используется неточная верхняя оценка уровня потребления памяти, в то время, как ядро операционной системы начинает отказывать процессам в выделении памяти.*

Эта теорема показывает, что предложенное улучшение не нарушит корректности работы алгоритма, который до него не отказывал процессам в памяти, пока уровень потребления группы $r_i = \sum_j R_j$ не превышал ограничения l_i.

Результатом предложенной модернизации алгоритма явилась почти полная компенсация потери производительности на стандартных конфигурациях, применяемых пользователями этой подсистемы в рамках проекта OpenVZ, что продемонстрировано результатами тестирования[8] некоторых типовых нагрузок на стандартные приложения (WEB-сервер, система резервного копирования данных, архиватор, система контроля версий и генератор фракталов) (см. рис. 1).

В третьей главе рассмотрена математическая модель, показывающая как можно обеспечивать гарантии выделения ресурсов ОС группам процессов. Введено определение гарантии, применительно к рассматриваемой ситуации.

Определение. *Гарантией* называется объем ресурсов, которое будет предоставлено группе по запросу *независимо от того, сколько ресурсов уже предоставлено другим группам.*

Показано, что достичь такого результата можно двумя способами – резервированием ресурса или специально подобранным ограничением в использовании этого ресурса (или их вариациями или комбинациями).

[8] В экспериментах в качестве такой оценки было использовано значение

$$A_j = \frac{R_j}{M_j}.$$

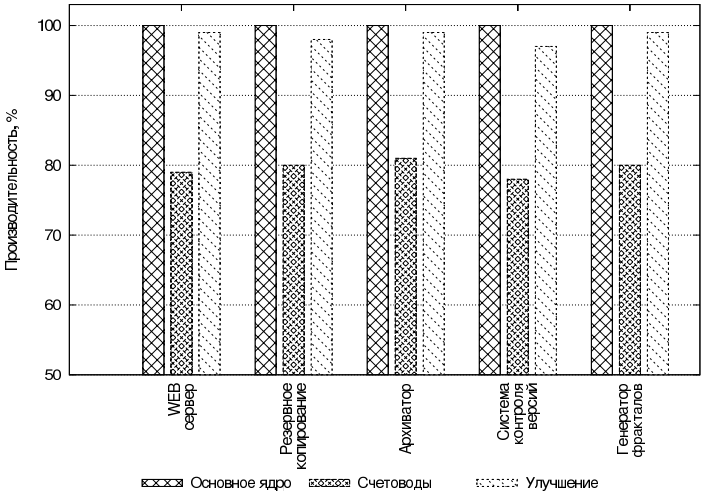

Рис. 1: Производительность ядер на стандартных конфигурациях.

В главе продемонстрирована ограниченность применимости первого способа – например, не каждый ресурс можно зарезервировать. Поэтому предпочтение отдано второму подходу, идея которого состоит в следующем. Если для какой-то группы требуется гарантировать x единиц ресурса, то можно ограничить все остальные группы в потреблении этого ресурса таким образом, что в системе всегда будет оставаться неиспользованный резерв требуемого объема x.

Модель построена на примере гарантированного выделения памяти. Рассмотрена система с R единицами (страницами, байтами, килобайтами) памяти, на которой требуется разделить ее между n группами. Количество памяти, которое требуется гарантированно предоставлять группам задается вектором $\vec{g} = (g_0, \ldots, g_{n-1})$.

При этом, естественно, требуется, чтобы набор g_i был корректен, то есть

$$\sum_i g_i \leq R. \tag{9}$$

Для достижения этого результата предложено вычислить вектор $\vec{l} =$

$(l_0, \ldots, l_{n-1})$, задающий ограничения потребления ресурсов в группах, таким образом, чтобы при любом уровне потребления любых $n-1$ групп в системе оставалось необходимое для оставшейся группы количество ресурса (т.е. g_i). При этом ожидается, чтобы полученные ограничения не будут противоречить требуемым гарантиям, т.е.

$$l_i \geq g_i, \forall i. \tag{10}$$

Другими словами, требуется, чтобы для каждой компоненты вектора $\vec{g}$ выполнялось соотношение

$$g_i = R - \sum_{j \geq 0, j \neq i, j < n} l_j. \tag{11}$$

В главе показано, что система неравенств (11) в рамках рассматриваемой задачи эквивалентна системе линейных уравнений, которая в матричной форме выглядит следующим образом

$$A \cdot \vec{l} = R \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} - \vec{g}, \tag{12}$$

где

$$A = \begin{pmatrix} 1 & \ldots & 1 \\ & \ddots & \\ 1 & \ldots & 1 \end{pmatrix} - E, \tag{13}$$

и находя обратную матрицу

$$A^{-1} = \frac{A - (n-2)E}{n-1}. \tag{14}$$

можно получить, что искомые величины l_i задаются соотношением

$$l_i = g_i + \tilde{R}, \tag{15}$$

где

$$\tilde{R} = \frac{R - \sum_i g_i}{n-1}. \tag{16}$$

Решение этой задачи позволило доказать следующую теорему.

Теорема 2. *Введенные ранее условия корректности гарантий (9) и корректности вычисляемых ограничений (10) являются эквивалентными.*

Таким образом, для группы определены три зоны уровней потребления ресурсов (см. рис. 2) – зона «жестких» гарантий, в которой группе память будет гарантированно[9] предоставлена; зона «мягких» гарантий, в которой группе не гарантируется в выделении памяти, но и не отказывается при наличии свободных страниц; и запрещенная зона.

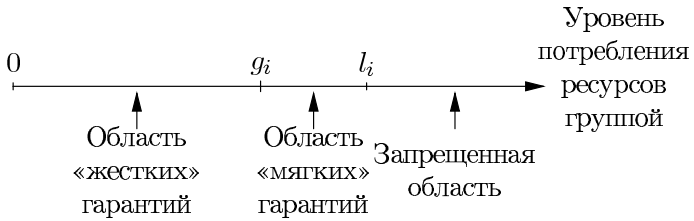

Рис. 2: Зоны уровней потребления ресурсов.

Особое внимание уделено проблеме **присутствия ядра операционной системы**. А именно – поскольку ядро тоже потребляет физическую память, то эту память тоже нужно учитывать в приведенных выше расчетах. Кроме того, отмечено, что создание большого количества объектов ядра может быть инициировано процессом, что дает возможность одному процессу потребить всю *ядерную* память, что, в свою очередь, может являться вариантом атаки типа «отказ в обслуживании».

Предложено определить вектор $\vec{l}$ как сумму ограничений двух типов – ограничение в использовании памяти ядра ($\vec{l_k}$) и ограничение в использовании памяти процесса ($\vec{l_u}$):

$$\vec{l} = \vec{l_k} + \vec{l_u}. \tag{17}$$

[9] Кроме случаев выхода системы из строя

В главе приведены алгоритмы, позволяющие учитывать выделение объектов ядра, связывать эти объекты с группой, которая инициировала их создание, и, таким образом, решать указанные проблемы.

Также в главе продемонстрирована применимость модели к другим типам ресурсов на примере планировщика процессов. Для задачи разделения процессорного времени уже есть эффективные алгоритмы, позволяющие ограничить группу процессов в доле потребляемого процессорного времени, так что комбинируя эти алгоритмы с описанным, можно достичь гарантий и в выделении процессорного времени.

В четвертой главе предложена модель «виртуальной выгрузки страниц на диск», которая является обобщением уже существующей технологии выгрузки на случай виртуальных машин. Такое обобщение позволило сделать управление и поведение виртуальных сред таким же, как у физических машин.

Технология выгрузки страниц на диск является логическим следствием модели виртуальной памяти – когда в системе заканчивается оперативная память, а процесс требует еще, можно часть страниц из оперативной памяти скопировать на другой, более емкий, носитель[10], а освободившееся место использовать для других нужд. Для более эффективной работы подобного алгоритма страницы памяти хранятся в списке, отсортированном по времени «недавнего использования» (LRU[11]).

В предлагаемой модели группе задано еще одно ограничение – ограничение в использовании дисковой области для хранения страниц. Как только множество страниц физической памяти, используемое группой, превышает заданное ограничение часть страниц перемещается на диск. При этом, для каждой группы необходимо точно знать какие страницы ей принадлежат, для чего они сохраняются в *еще один* список LRU,

[10] Чаще всего для этих целей используется жесткий диск, но иногда применяют сетевые и другие хранилища

[11] От англ. least recently used

в котором хранятся только страницы, используемые этой группой.

Таким образом, каждой странице памяти p ставится в соответствие две числовые характеристики: $age(p,t)$ – «возраст» страницы, равный времени, прошедшему с момента, когда ядро зарегистрировало факт обращения к странице до текущего момента времени t *любым* процессом; и $age_g(p,t)$ – «возраст» страницы в группе, вычисляемый аналогично, но лишь для обращений процессов *из группы* g.

В главе доказана важная теорема.

Теорема 3. *В любой момент времени t для любых двух страниц p_1 и p_2 и группы g свойство $age(p_1,t) \leq age(p_2,t)$ равносильно свойству $age_g(p_1,t) \leq age_g(p_2,t)$ в том и только том случае, когда к странице обращаются только процессы из той же группы.*

Данная теорема показывает, в каком случае расширение алгоритма на случай виртуальных машин не приводит к нарушениям в его работе. А именно, в случае, когда к страницам памяти группы обращаются процессы, не принадлежащие этой группе, алгоритм может выбрать неправильную, с точки зрения группы страницу для выгрузки.

В главе также рассмотрены возможные проблемы, связанные с применением изложенного во второй главе алгоритма и описано примененное улучшение выгрузки – при наличии свободной памяти страницы на диск не записываются, а кэшируются и выгружаются на диск позже.

Описанный в главе алгоритм был реализован автором в виде комплекса программ – расширения к ядру ОС Linux. Был проведен ряд экспериментов с модифицированным ядром и в главе также приведены результаты этих экспериментов.

В заключении приведены основные результаты и выводы диссертации.

Основные результаты и выводы диссертации

1. Разработана математическая модель предоставления гарантий выделения группам процессов физической памяти. Показана применимость модели к другим типам ресурсов на примере задачи распределения процессорного времени.

2. Предложена математическая модель и разработан использующий ее алгоритм ограничения групп процессов в потреблении физической памяти. Теоретически обоснована и показана в ходе численных экспериментов более высокая производительность нового алгоритма по сравнению с существующими аналогами.

3. Разработан и реализован в виде комплекса программ алгоритм выгрузки страниц памяти на диск для групп процессов, позволяющий объединить принципы управления физическими и виртуальными серверами. Выведены условия корректности работы алгоритма.

Список публикаций по теме диссертации

1. *Емельянов П. В., Коротаев К. С.* Математическая модель обеспечения гарантий выделения ресурсов операционной системы // Информационные технологии. — М. 2008 — № 3 — С. 83–85.

2. *Коротаев К., Емельянов П.* Многоуровневый планировщик процессорного времени для групп процессов, обеспечивающий гарантии в обслуживании // Информационные технологии. — М. 2006. — № 6. — С. 58–63.

3. *Емельянов П. В.* Оптимизация модели контроля потребления памяти группой процессов // Вестник Новосибирского Гос. Унив-та. Серия: Информационные Технологии — 2008 — Т. 6, вып. 1 – С. 37–46.

4. *Коротаев К., Емельянов П.* Модифицированный SFQ алгоритм (MSFQ) для честного распределения процессорного времени на многопроцессорных системах // Проблемы вычислительной математики, математического моделирования и информатики: Сб.н.тр. — М.: МЗ Пресс, 2006. — С. 214–233.

5. *Емельянов П., Савочкин А.* Расширение функциональности существующей модели контроля памяти в ядре Linux // Процессы и методы обработки информации: Сб.ст. — М.: Моск. физ.-техн. ин-т, 2005. — С. 77–80.

6. *Емельянов П., Коротаев К., Луковников И.* Основные проблемы реализации алгоритмов пропорционального планирования // Процессы и методы обработки информации: Сб.ст. — М.: Моск. физ.-техн. ин-т, 2006. — С. 86–91.

7. *Емельянов П. В., Коротаев К. С., Лунев Д. В.* Расширение функциональности существующей модели контроля памяти в ядре Linux //

Соврем. проблемы фундаментальных и прикладных наук. Часть VII. Управление и прикладная мат-ка: Труды XLVII научной конф-ции. — М. — Долгопрудный: Моск. физ.-техн. ин-т, 2004. — С. 37.

8. *Емельянов П. В., Лунев Д. В.* Математическая модель предоставления гарантий выделения ресурсов на базе жестких ограничений // Соврем. проблемы фундаментальных и прикладных наук. Часть VII. Управление и прикладная мат-ка: Труды 49-й научной конф-ции. — М. — Долгопрудный: Моск. физ.-техн. ин-т, 2006. — С. 37–38.

9. *Pavel Emelianov, Denis Lunev, Kirill Korotaev* Resource Management: Beancounters // Proceedings of the Linux Symposium — Ottawa: 2005. — Pp. 285–292.

10. *Pavel Emelianov, Denis Lunev, Kirill Korotaev* Resource Management: Beancounters // Proceedings of the LinuxConf Europe — Cambridge: 2007. — 7 Pp.

11. *Balbir Singh, Pavel Emelianov* Memory Controller: рабочая программа [Электронный ресурс]: хранилище содержит исходные тексты ядра ОС Linux. — Электрон. дан. (1 файл). — 2007. — Режим доступа: http://git.kernel.org/?p=linux/kernel/git/torvalds/linux-2.6.git;a=blob;f=mm/memcontrol.c

В работах с соавторами соискателю принадлежат следующие результаты. [1] – разработка математической модели предоставления гарантий выделения физической памяти; [2, 4] – проведение численных экспериментов над реализованным комплексом программ и изложение результатов; [5] – разработка комплекса программ по управлению распределением памяти ядра; [6] – анализ проблем, возникающих в задачах пропорционального планирования; [11] – реализация алгоритмов подсчета и ограничения использования физической памяти и выгрузки страниц на диск.

Емельянов Павел Владимирович

МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ ПРОЦЕССОВ УПРАВЛЕНИЯ ПОТРЕБЛЕНИЕМ ПАМЯТИ В МНОГОПОЛЬЗОВАТЕЛЬСКИХ СРЕДАХ И СИСТЕМАХ ВИРТУАЛИЗАЦИИ

Подписано в печать 09.09.2008 г. Формат 60x84 $^1/_{16}$. Печать офсетная.
Усл. печ. л. 1,0. Уч.-изд. л. 1,0. Тираж 80 экз. Заказ №

Государственное образовательное учреждение
высшего профессионального образования
Московский физико-технический институт (государственный университет)
Отдел автоматизированных систем «ФИЗТЕХ-ПОЛИГРАФ»

141709, Московская обл., г. Долгопрудный, Институтский пер. 9