информации ИАИ-2010», Киев, 2010
2. *Валькман Ю.Р.* Не-факторы – основа образного мышления // Труды II-го Междунар. Научно-практ. Семинара «Интегрированные модели и мягкие вычисления в искусственном интеллекте». Москва: Физматлит, 2003. С. 26-33.
3. *Валькман Ю.Р.* Целостность образов: о моделировании смысла и понимания // International Journal "Information Technologies & Knowledge" vol.6, Number 1, 2012. С 14-25.
4. *Веккер, Л. М.* Психические процессы: [В 3-х т.]. — Л.: Изд-во Ленинградского университета, 1974. — Т. 1. Ощущение и восприятие. — 334 с.
5. *Веккер, Л. М.* Психические процессы: [В 3-х т.]. — Л.: Изд-во Ленинградского университета, 1976. — Т. 2. Мышление и интеллект. — 342 с.
6. *Веккер, Л. М.* Психические процессы: [В 3-х т.]. — Л.: Изд-во Ленинградского университета, 1981. — Т. 3. Субъект. Переживание. Действие. Сознание. — 326 с.
7. *Мельников Г.П.* Системология и языковые аспекты кибернетики. М.: Сов. Радио, 1978. – 368 с.
8. *Дж. Хокинс, С. Блейкси.* Об интеллекте. Изд. Дом «Вильямс» / Москва-Санкт Петербург-Киев. 2007
9. *Лакофф Дж.* Мышление в зеркале классификаторов / Пер. с англ. Р. И. Розиной // Новое в зарубежной лингвистике. Вып. XXIII. Когнитивные аспекты языка. - М., 1988 - С. 12-21.
10. *Лакофф Дж.* Женщины, огонь и опасные вещи: Что категории языка говорят нам о мышлении / Пер. с англ. И.Б.Шатуновского. – М.:Языки славянской культуры, 2004. – 792 с. – (Язык. Семиотика. Культура).

Поступила 21.10.2013р.

УДК 004.832.3

В. В. Зосимов, г. Николаев

ПЛАНИРОВЩИКИ ЗАДАЧ ОПЕРАЦИОННОЙ СИСТЕМЫ LINUX И ПЕРСПЕКТИВЫ ИХ РАЗВИТИЯ ДЛЯ ЭФФЕКТИВНОГО ВЕДЕНИЯ НАУЧНЫХ РАСЧЕТОВ

*Аннотация***.** В данной статье описано развитие планировщиков задач операционной системы Linux. Рассмотрены цели планирования и необходимые свойства планировщиков. Сделаны выводы о необходимости разработки новых планировщиков, которые в отличие от существующих, будут ориентированы в первую очередь не на обеспечение "комфортного" общения пользователя с интерфейсом операционной системы, а на ускорение процесса решения научных задач с использованием сложных ресурсоемких алгоритмов.

*Abstract***.** This article describes the development of a task scheduler of the operating system Linux, planning aims and the necessary properties of planners. There are made the conclusions about the need for new planners, whitc, unlike the existing ones, will be targeted in the first place not to ensure a "comfortable" communication

with user interface of the operating system, but to accelerate the solution of scientific problems using sophisticated resource-intensive algorithms.

Ключевые слова. Планировщик задач операционной системы, операционная система linux, научные расчеты, кластерные технологии, параллельные вычисления, распределенные вычисления.

Актуальность. Операционная система Linux давно зарекомендовала себя как эффективное и надежное средство для решения различных задач. Особенно широкое распространение она получила в сфере серверных приложений, однако существуют дистрибутивы, адаптированные для работы настольных компьютеров и мобильных устройств. Высокая популярность операционной системы Linux обусловлена тем, что она является некоммерческой системой с открытым кодом и над ее усовершенствованием работает все интернет-сообщество.

Одной из основных подсистем ядра Linux является планировщик задач - неотъемлемая часть любой многозадачной операционной системы. Планировщик занимается распределением ресурсов процессорного времени между задачами операционной системы для обеспечения их оптимальной работы. Исследования в напрпавленни усовершенствования работы существующих и разработки новых планировщиков задач для операционной системы Linux могут помочь повысить эффективность работы операционной системы в целом или определенных однотипных задач.

Цели планирования

В ядре операционной системы Linux есть две части, наиболее критичные ко времени исполнения, это подсистема управления памятью и планировщик задач. Связано это с тем, что они влияют на работу практически всех остальных частей ядра и операционной системы в целом. По этой причине они должны быть абсолютно безупречны и оптимальны.

Планировщик Linux преследует несколько целей [1-5]:

1. Беспристрастность :

Планировщик должен беспристрастно выделять процессорное время каждой задаче. В новом ядре был проделан обширный объем работ для того, чтобы обеспечить справедливое распределение квантов времени между процессами.

2. Производительность и загрузка процессора :

Планировщик должен стараться максимизировать производительность и загрузку процессора. Обычно это достигается за счет увеличения объема мультипрограммирования. Но такой подход дает прирост только до определенного момента, после которого становится непродуктивным.

3. Минимальные накладные расходы :

Сам планировщик должен занимать процессор настолько малое время, насколько это возможно. Время реакции планировщика должно быть минимальным. Но при оценке времени реакции необходимо учитывать очень

важный момент. Считается, что процесс планирования не является полезной работой, однако, если планирование произведено безупречно, даже если оно потребовало дополнительных затрат некоторого количества времени, то это определенно стоит затраченных усилий. Но здесь возникает вопрос, как решить, где "золотая середина"? Большинство планировщиков решают эту проблему с помощью эвристических алгоритмов.

4. Планирование на основе приоритетов :

Приоритетное планирование означает, что одни процессы могут иметь превосходство перед другими в конкуренции за процессор. Планировщик, по крайней мере, должен различать процессы занятые вводом-выводом и числовые расчеты. Кроме того должен быть учтен эффект "застаивания" так, чтобы в системе не было "зависших" процессов. Linux поддерживает приоритеты и различает разные категории процессов. Ядро различает пакетные и интерактивные задачи. Каждая из них получает свою долю процессорного времени в соответствии со своим приоритетом.

5. Время цикла обслуживания, время ожидания :

Время цикла обслуживания -- это сумма времени, потраченного на обслуживание процесса и время, проведенное задачей в очереди готовых к запуску процессов. Это время должно быть сведено к минимуму.

6. Время отклика и дисперсия :

Скорость реакции программы должна быть настолько высокой, насколько это возможно. Но при этом не надо забывать и о другом важном показателе -- дисперсии времени отклика, который зачастую игнорируется. Совершенно недопустимо, когда среднее время отклика задачи невелико, но при этом иногда возникают длительные задержки в интерактивных процессах.

7. Прочие :

Планировщик должен преследовать и другие цели, например предсказуемость. Поведение планировщика должно быть предсказуемым для данного множества процессов с назначенными приоритетами. При увеличении нагрузки, падение производительности планировщика должно быть гладким. Это особенно важно, поскольку Linux приобретает все большую популярность на рынке серверов, а серверы могут испытывать серьезные нагрузки в часы пик.

Планировщики задач операционной системы Linux и их развитие

На первый взгляд, распределение процессорного времени между задачами операционной системы является простой задачей, но так как на современных компьютерах могут выполняться сотни, а то и тысячи процессов, неправильная его реализация может снизить общую производительность системы. Важно учитывать, что даже на переключение контекста процесса тратится не малая часть времени. Кроме того,

планировщик постоянно сталкивается с такой проблемой, как ограниченное время ответа для критических задач.

Долгое время в Linux присутствовал один планировщик - O(1) [6]. При этом были и другие предложения, например Staircase от разработчика Кона Коливаса, и Fairsched, в котором процессы разбиты на группы, а стандартный планировщик гарантирует распределение времени в зависимости от веса группы. Но все они не попадали в основную ветку ядра.

В июле 1993 года Линус Торвальдс описал принцип работы планировщика задач Linux [7]. Оригинальный файл sched.c содержал всего 254 строки кода, это был простой и понятный алгоритм. В 1996 году в нем было уже более 6000 строк - Дэйв Грот устранил проблему с семафорами и SMP системами. В 2002 году был представлен новый планировщик "ultra-scalable O(1) scheduler" от Инго Молнара. В настоящее время sched.c содержит уже тысячи строк кода.

Алгоритм работы O(1) очень прост. Каждая задача имеет фиксированное число (tick), которое пересчитывается с каждым системным тиком (по умолчанию 100 Hz) при выходе из режима ядра или при появлении более приоритетной задачи. Алгоритм делит число на 2 и добавляет базовую величину (по умолчанию 15, с учетом величины nice). Когда тик становится равным 0, процесс устанавливает флаг need_resched, и тик пересчитывается. Кроме этого, каждый процесс имеет две очереди. В одной находятся готовые к запуску задачи, во вторую помещаются отработавшие и спящие задачи, которые, например, ожидают недоступный в настоящее время ресурс. Когда первая очередь пустеет, очереди меняются местами. Поэтому время работы алгоритма постоянно и не зависит от количества процессов. Современная реализация O(1) использует более сложные алгоритмы (например, анализируя среднее время сна), чтобы обнаружить интерактивные процессы и постараться задержать их в активном дереве.

В ядро 2.6.23 в качестве основного был включен планировщик **CFS** (**Completely Fair Scheduler**, абсолютно справедливый планировщик), над которым работает Инго Молнар [8]. В нем для хранения процессов используется red-black дерево, где ключом является значение wait_runtime каждого процесса. Эта переменная определяет количество наносекунд, которое переработал или недоработал процесс. В зависимости от этого значения процесс и получает свое место в дереве. В CFS используются наносекунды, а не time slices или тики, в его работе нет никакой эвристики. Извлечение процесса и его вставка в дерево требует перестройки, что при большом количестве процессов приводит к увеличению накладных расходов. Поэтому CFS рекомендуется в первую очередь для десктопов, где нет большого количества одновременно запущенных процессов. В отличие от O(1), CFS равномернее планирует процессорное время (фактически если задачи две, то каждая получит ровно 50% CPU), распределяет задачи по нескольким ядрам и имеет меньшее время отклика. Для настройки CFS используется файл /proc/sys/kernel/sched_granularity_ns, в котором по

умолчанию установлен режим desktop (меньшее время задержки), но при необходимости его можно переключить в server, обеспечив лучшую группировку.

Кон Коливас, остановив разработку ветки ck, в марте 2007 года анонсировал совершенно новый планировщик **RSDL** (**Rotating Staircase DeadLine scheduler**), который в последствии был переименован в SD (Staircase Deadline) [9]. За его основу был взят Staircase. Задача нового планировщика - исключить зависания, присущие O(1). Здесь все процессы равны, каждому выделяется своя квота, исчерпав которую, он опускается на следующий уровень приоритета, где получает новую квоту. Причем каждый уровень также имеет свою квоту, если ее исчерпает хотя бы один процесс, все перейдут на следующий уровень, вне зависимости от того, отработали ли они свою квоту, или нет. Планировщик также пытается определить интерактивные процессы, автоматически повышая им приоритет. Версия SD показывала неплохие результаты на серверах. После этого разработчик Роман Зиппель, которого не устраивала сложность планировщика CFS, представил рабочий прототип базового алгоритма нового планировщика **RFS** (**Really Fair Scheduler**), который помещает задачу в виртуальную (нормализованную) временную линию, где имеет значение только относительное расстояние между двумя любыми задачами. После большого количество споров, в ответ на этот прототип Инго Молнар представил свою версию, которую он назвал **RSRFS** (**Really Simple Really Fair Scheduler**), реализованную поверх CFS и включающую алгоритм из RFS.

Заключение. Исследование показало, что все существующие планировщики задач для операционной системы Linux разработаны как универсальные инструменты для управления любыми типами задач. Но при этом уклон делается в сторону интерактивных задач и обеспечения минимальных задержек при работе пользователя с интерфейсом операционной системы, что продиктовано все возрастающей популярностью мобильных приложений. Такой подход позволяет получить неплохую производительность и стабильность работы для большинства домашних и офисных компьютеров. Минусом такого подхода является отсутствие возможности оптимизации управления ресурсами операционных систем при решении специфичных задач например, при ведении научных расчетов, особенно если эти расчеты ведутся на больших кластерах под управлением операционной системы Linux. Все возрастающая популярность операционной системы Linux и использование ее на самых разных электронных устройствах, в том числе и на вычислительных кластерах, делает актуальной задачей разработку новых планировщиков для оптимизации работы операционной системы при ведении сложных научных расчетов с учетом минимального общения пользователя с интерфейсом операционной системы.

1. *Планировщик задач в Linux (proccess cpu kernel linux):* http://www.opennet.ru
2. *Лав Р*. Разработка ядра Linux, 2-е издание / Р. Лав // Пер. с англ. – М.: ООО И.Д. Вильямс. – 2006. – 448 с.
3. *Fujiyama R., Scherpelz J., Ferlo M.*. Analyzing the Linux schedulers's tunables, 2003 – 19 с.
4. *Inside the self-tuning "Genetic" Linux.:* http://apcmag.com/3095 /inside_the_self_tuning_genetic_linux
5. *Jake Moilanen's Linux Kernel Homepage:* http://kernel.jakem.net
6. *Планировщики процессов:* http://www.xakep.ru
7. *Планировщики процессов:* http://ck.kolivas.org/patches/staircase-deadline/ rsdl_scheduler.readme
8. *Kolivas C*. Linux Kernel CPU Scheduler Contributor, IRC conversations, no transcript, 2004.
9. *Aas J*. Understanding, the Linux 2.6.8.1 CPU Scheduler/ Silicon Graphics, Inc. (SGI), 2005.

Поступила 2.10.2013р.

УДК: 519.237.8(045)

Т. І. Олешко, Н. В. Ратушна, м.Київ

ВИКОРИСТАННЯ МЕТОДІВ МАТЕМАТИЧНОГО МОДЕЛЮВАННЯ В СИСТЕМІ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ

Аннотация. В статье рассмотрены основные методы математического моделирования, которые используются в системе поддержки принятия решений.
Ключевые слова: моделирование, иерархия, система поддержки принятия решений, метод анализа иерархий.

Постановка проблеми. Використання методів математичного моделювання у різних сферах людської діяльності призвели до розуміння багатьох принципових труднощів, що виникають при їх впровадженні в реальну практику прийняття рішень. Особа, яка приймає рішення, при прийнятт і рішення враховує величезну кількість різноманітних показників, уявити які у вигляді єдиного критерію вдається тільки в деяких випадках. Методики природничих наук зовсім недостатньо для вирішення більш складних проблем, які по суті своїй багатокритеріальні. При пошуку "кращого" плану або альтернативи істотне значення мають фактори, що не піддаються формалізації. Тому керівник (ОПР), що аналізує рішення розуміє, що формалізовані фактори можуть надати більш сильний вплив на результат, ніж, наприклад, оптимальний розподіл ресурсів. Якщо, крім того, врахувати, що ОПР зазвичай має в голові величезне число обмежень, які він не хотів би порушити, то стане ясно, чому він схильний прийняти власне рішення, відмінне від отриманого за допомогою комп'ютера. Один зі способів

