ВЫПОЛНИМОСТЬ ПРИЛОЖЕНИЙ РЕАЛЬНОГО ВРЕМЕНИ НА МНОГОЯДЕРНЫХ ПРОЦЕССОРАХ

НИКИФОРОВ В. В.

УДК 681.3

Никифоров В. В. **Выполнимость приложений реального времени на многоядерных процессорах.**

Аннотация. Дан краткий перечень ключевых результатов по анализу выполнимости приложений реального времени, реализуемых на многоядерных процессорах с использованием различных дисциплин планирования. Для дисциплин планирования со статическими приоритетами задач предложен метод оценки времени отклика задач. Для дисциплин пропорционального планирования с квантованием и без квантования времени даны оценки числа точек перепланирования и числа переключений контекста.

Ключевые слова: системы реального времени, многоядерные процессоры, планирование заданий, анализ выполнимости.

Nikiforov V.V. **Basic Requirements to the SPIIRAS Transactions Paper Format Feasibility of Real-Time Applications on Multicore Processors.**

Abstract. A brief survey of key results is presented on the feasibility analysis of real–time applications that are implemented by multicore processors with various scheduling methods. A method for estimation of task response time is suggested for scheduling with static task priorities. Number of planning points and number of context switches are estimated for quantum–based and non–quantum scheduling methods.

Keywords: real–time systems, multicore processors, task scheduling, feasibility analysis.

0. Введение. Перспектива широкого применения многоядерных процессоров для реализации систем реального времени (СРВ) вызвала проведение ряда исследований, направленных на разработку эффективных методов распределения ресурса процессорного времени между задачами, составляющими программное приложение для таких систем (эффективных дисциплин планирования). Основная цель таких исследований состоит в разработке методов оценки выполнимости приложений — методов, обеспечивающих проверку гарантий своевременности выполнения задач в условиях использования конкретных дисциплин планирования. В ходе этих исследований модели и методы, использовавшиеся для однопроцессорных систем с традиционными одноядерными процессорами, обобщались для применения к системам с многоядерными процессорами. Первые значительные результаты в этом направлении получены для приложений, которые состоят из периодических независимых задач с вытеснениями.

Труды СПИИРАН. 2009. Вып. 8. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2009. Issue 8. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

1. Вычислительные модели приложений.

Анализ выполнимости программных приложений для СРВ выполняется с использованием вычислительных моделей, представляющих те характеристики прикладных задач, которые влияют на особенности разделения задачами ресурса процессорного времени. В простейших моделях отдельная *задача* τ_i характеризуется двумя параметрами:

1) $\tau_i = (C_i, T_i)$ — *объемом* C_i вычислений;

2) *периодом* T_i.

Это означает, что моменты времени *активизации* задачи τ_i (моменты *порождения* соответствующих ей *заданий*) следуют с интервалом T_i и для выполнения каждого из заданий требуется C_i единиц процессорного времени. Предполагается, что программное приложение состоит из ряда независимых задач $\{\tau_1, \tau_2, \ldots, \tau_n\}$. Под *независимостью* задач понимается отсутствие таких межзадачных взаимосвязей (по обмену сообщениями, доступу к разделяемым аппаратным и информационным ресурсам), которые приводили бы к логическим ограничениям на порядок выполнения фрагментов различных заданий.

Очередная, *j*–я активизация задачи τ_i означает порождение очередного *j*–го ее экземпляра — порождение задания $\tau_i^{(j)}$. Каждая задача τ_i является статическим объектом (последовательная программа, представляющая алгоритм исполнения задачи, строится на этапе разработки системы).

Различные задания, порождаемые в результате активизации одной и той же задачи, будем называть однотипными заданиями. Таким образом, задача представляет собой тип заданий, порождаемых в моменты ее активизации. Каждое из заданий $\tau_i^{(j)}$, порождаемых в результате активизации задачи τ_i, является *динамическим* объектом типа τ_i (*j*–м экземпляром задачи τ_i).

Порождение задания означает увеличение числа претендентов на процессорное время, и, следовательно, изменение условий распределения процессорного времени. Если в общем случае изменение условий распределения системных ресурсов называется *системным событием*, то порождение задания следует рассматривать как одну из раз-

Труды СПИИРАН. 2009. Вып. 8. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2009. Issue 8. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

новидностей системных событий. Другим примером системных событий является завершение задания — уменьшение числа претендентов на процессорное время.

2. Интервал существования задания.

В приложении с независимыми задачами задание является *активным* в некоторый конкретный момент времени, если к этому моменту оно уже порождено, но еще не завершено. То есть каждое задание является активным в рамках интервала его существования – от момента порождения $t_{\text{arr}}(\tau_i^{(j)})$ до момента завершения $t_{\text{end}}(\tau_i^{(j)})$. Продолжительность интервала существования задания

$$r_i^{(j)} = t_{\text{end}}(\tau_i^{(j)}) - t_{\text{arr}}(\tau_i^{(j)})$$

зависит от комбинации системных событий, возникающих в ходе исполнения $\tau_i^{(j)}$. Максимально возможная (при любых сценариях системных событий) продолжительность интервала существования заданий типа τ_i называется *временем отклика*

$$R_i = \max\{r_i^{(j)} \mid j = 1, 2, ...\}$$

задачи τ_i. В приложении с независимыми задачами каждое из активных заданий может пребывать в одном из двух *системных состояний*:

1) задание, владеющее исполнительным ресурсом, находится в *текущем состоянии*;

2) в конкретный момент времени часть активных заданий может пребывать в *состоянии ожидания* предоставления ему исполнительного ресурса, т.е. ресурса процессорного времени в виде отдельного одноядерного процессора или отдельного ядра многоядерного процессора.

3. Вытеснение заданий.

В рамках данной работы полагается, что текущее задание допускает возможность *вытеснения*: ход исполнения любого задания может быть прерван (приостановлен) ввиду переключения процессора на исполнение вновь порожденного более приоритетного задания. Через некоторый интервал времени исполнение вытесненной задачи возобновляется для продолжения недовыполненных вычислений. Таким образом, возникает *интерференция* — взаимное влияние активных заданий на сроки их исполнения.

Критическим сценарием для задачи τ_i называется такая комбинация разнесения системных событий во времени, которая в результате интерференции заданий приводит к растягиванию интервала сущест-

Труды СПИИРАН. 2009. Вып. 8. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2009. Issue 8. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

вования очередного задания типа τ_i до максимально возможного значения R_i.

Далее рассматриваются вычислительные модели приложений с жесткими требованиям к СРВ, для которых время отклика R_i каждой из задач строго регламентировано: несмотря на интерференцию исполнение каждого из заданий должно завершиться к заданному моменту времени. В общем случае ограничение на допустимую продолжительность выполнения заданий типа τ_i задается отдельным параметром вычислительной модели, величиной D_i — *относительным предельным сроком* выполнения заданий типа τ_i. Это означает, что для каждой из задач должно выполняться неравенство $R_i \le D_i$.

Параметр D_i является одним из определяющих параметров вычислительной модели, накладывающим ограничение на длину интервала существования каждого из заданий типа τ_i: интервал существования задания $\tau_i^{(j)}$ ограничен моментом времени

$$d_i^{(j)} = t_{\text{end}}(\tau_i^{(j)}) + D,$$

называемым *абсолютным предельным сроком* завершения $\tau_i^{(j)}$.

В простейших вычислительных моделях полагается, что продолжительность $r_i^{(j)}$ выполнения задания $\tau_i^{(j)}$ не должна превышать T_i, т.е. каждое из заданий $\tau_i^{(j)}$ типа τ_i должно быть завершено до момента порождения следующего задания того же типа.

4. Производные характеристики задач и приложений.

Наряду с двумя определяющими модельными параметрами каждой из задач τ_i (объемом вычислений C_i и периодом T_i) при оценке выполнимости приложений используется важная производная характеристика u_i, называемая *нагрузкой* на процессор от задачи τ_i. Значение нагрузки от τ_i определяется как отношение требуемого для выполнения задачи объема вычислений к длине периода активизации задачи:

$$u_i = C_i / T_i .$$

Общей характеристикой приложения в целом является *суммарная нагрузка*

Труды СПИИРАН. 2009. Вып. 8. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2009. Issue 8. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

$$U_i = \sum_{i=1}^{n} u_i$$

от всех составляющих приложение задач. Величина U_i представляет удельный объем процессорного времени, требуемый для исполнения приложения.

В дальнейшем, если не оговорено особо, будем индексировать составляющие приложение задачи τ_i в порядке уменьшения частоты их активизации: $T_i \le T_{i+1}$ для всех τ_i, $1 \le i \le n$.

5. Планирование.

Модельные параметры C_i, T_i, D_i, u_i определяются на этапе разработки системы. На базе этих статических параметров возможен априорный анализ свойств задач и системы в целом.

5.1. Дисциплина планирования. Если в ходе работы приложения в некоторый момент времени множество активных заданий пусто (активные задания отсутствуют), то процессор простаивает. В противном случае необходимо принимать решение, каким из активных заданий следует предоставить процессорное время. Порядок предоставления процессорного времени для обслуживания активных заданий определяется используемой дисциплиной планирования. Таким образом, принятой в системе дисциплиной планирования определяется, каким именно из активных заданий в конкретные моменты работы системы будет выделяться ресурс процессорного времени. Любую дисциплину планирования можно определить как способ наделения заданий целочисленными *приоритетами*: в очередной момент реализации дисциплины планирования активным заданиям присваиваются определенные значения приоритетов и ресурс процессорного времени выделяется наиболее приоритетным из активных заданий.

5.2. Классы дисциплин планирования. Всевозможные дисциплины планирования подразделяются на классы в зависимости от того, какие ограничения накладываются на способ назначения приоритетов активным заданиям. Для самого широкого класса $\mathbf{A_0}$ дисциплин планирования такие ограничения отсутствуют. Дисциплина планирования относится к классу дисциплин с неизменными приоритетами заданий в случае, если при ее реализации приоритет любого задания $\tau_i^{(j)}$ на всем интервале его существования остается неизменным (обозначим этот класс дисциплин планирования символом $\mathbf{A_1}$). Более строгие ограничения на способ назначения приоритетов соблюдаются в классе $\mathbf{A_2}$ (дисциплины планирования с неизменными приоритетами задач). Для

Труды СПИИРАН. 2009. Вып. 8. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2009. Issue 8. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

класса $\mathbf{A_2}$ приоритет задания однозначно определяется его типом: любые два однотипные задания $\tau_i^{(k)}$ и $\tau_i^{(l)}$ порождаются с одним и тем же значением приоритета, при этом приоритет остается неизменным на протяжении всего периода существования задания.

Согласно определениям, имеют место отношения вложенности рассматриваемых классов дисциплин планирования: $\mathbf{A_0} \supset \mathbf{A_1} \supset \mathbf{A_2}$. По сложившейся традиции дисциплины класса $\mathbf{A_1}$ называют *приоритетно–регулируемыми* (*priority–driven*) дисциплинами планирования. Множество $\mathbf{A_1}$ разделяется на два подмножества:

1) подмножество $\mathbf{A_1}/\mathbf{A_2}$ (дисциплины планирования с *динамическими* приоритетами),

2) подмножество $\mathbf{A_2}$ (дисциплины планирования со *статическими* приоритетами).

Примером дисциплины планирования, принадлежащей подмножеству $\mathbf{A_0}/\mathbf{A_1}$ (дисциплины, не относящейся к приоритетно-регулируемым), является RR (*Round Robin*) планирование. В соответствии с RR–дисциплиной планирования активные задания стоят в очереди на использование процессора. Процессорное время выделяется заданиям квантами фиксированной длины. Задание, израсходовавшее предоставленный квант процессорного времени, переводится в конец очереди, процессор передается заданию, выдвинувшемуся в очереди на первое место. Другие примеры дисциплин планирования из подмножества $\mathbf{A_0}/\mathbf{A_1}$ — дисциплины пропорционального планирования (Pfair-дисциплины планирования) рассмотрены далее.

5.3. Оптимальные дисциплины планирования. Задача τ_i, входящая в состав приложения $\{\tau_1, \tau_2, \ldots, \tau_n\}$ с заданными значениями модельных параметров считается *выполнимой* в рамках использования конкретной дисциплины планирования, если гарантируется своевременность выполнения всех заданий типа τ_i при любых допустимых данной дисциплиной планирования сценариях интерференции заданий приложения $\{\tau_1, \tau_2, \ldots, \tau_n\}$.

Приложение $\{\tau_1, \tau_2, \ldots, \tau_n\}$ с заданными значениями модельных параметров считается *выполнимым* в рамках конкретного класса дисциплин планирования, если в этом классе найдется такая дисциплина планирования, при использовании которой гарантируется выполнимость каждой из задач $\tau_1, \tau_2, \ldots, \tau_n$.

Труды СПИИРАН. 2009. Вып. 8. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2009. Issue 8. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

Дисциплина планирования считается *оптимальной* в своем классе, если для любого выполнимого в этом классе приложения $\{\tau_1, \tau_2, \ldots, \tau_n\}$ использование этой дисциплины обеспечивает выполнимость каждой из задач $\tau_1, \tau_2, \ldots, \tau_n$.

5.4. RM–планирование. Одним из вариантов дисциплин планирования класса $\mathbf{A_2}$ является частотно-монотонная (rate monotonic — RM) дисциплина планирования: чем меньше период задачи, тем выше ее приоритет. В одной из первых работ по анализу выполнимости для однопроцессорных систем [1] показано, что RM–планирование является оптимальным в классе $\mathbf{A_2}$ при реализации приложений из независимых задач на классических одноядерных процессорах и выполнении для каждой из задач условия $D_i = T_i$ (условие исполнения приложений *без буферизации заданий*). В той же работе сформулировано достаточное условие выполнимости таких приложений. Проверка этого условия для приложения, состоящего из n задач (называемая UB-тестом), состоит в сопоставлении суммарной нагрузки от всех задач приложения с граничным значением $UB(n)$:

$$\sum_{i=1}^{n} u_i \le UB(n) = n(2^{1/n} - 1) \qquad (1)$$

С ростом числа задач от единицы до бесконечности величина $UB(n)$ монотонно убывает от единицы до $\ln 2 \approx 0.693$. Выполнимость приложения гарантируется при положительном результате UB-теста и использовании RM–планирования.

Условимся индексировать задачи в порядке понижения их приоритетов: задача τ_1 имеет наивысший приоритет, с увеличением индекса приоритет заданий снижается. В таких обозначениях критическим сценарием для задачи τ_i является одновременная активизация задач $\tau_1, \tau_2, \ldots, \tau_n$.

5.5. DM–планирование. Рассмотренный класс вычислительных моделей приложений может быть расширен путем замены условия $D_i = T_i$ условием $D_i \le T_i$, что означает наложение более строгих ограничений на допустимое время отклика задач: для некоторых задач (или для всех задач) допустимое значение R_i должно быть строго меньше соответствующего значения периода. Для таких приложений оптимальным в классе $\mathbf{A_2}$ является DM–планирование: чем короче относительный предельный срок завершения задачи, тем выше ее приоритет.

Труды СПИИРАН. 2009. Вып. 8. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2009. Issue 8. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

При этом критическим сценарием для задачи τ_i по–прежнему является одновременная активизация задач τ_1, τ_2, …, τ_n.

Проверка выполнимости приложения при использовании DM–планирования сводится к вычислению времени отклика для каждой из задач путем решения рекуррентного выражения [2]:

$$R_i = C_i + \sum_{prio(\tau_j) > prio(\tau_i)} \left\lceil R_i \Big/ T_j \right\rceil C_j \qquad (2)$$

где $prio(\tau_i)$ — приоритет задачи τ_i. Приложение выполнимо, если для всех задач выполняется неравенство $R_i \le D_i$.

5.6. EDF–планирование. В рамках класса $\mathbf{A_1}$ различные задания типа τ_i порождаются с различными значениями приоритетов, но в рамках интервала существования задания $\tau_i^{(j)}$ его значение его приоритета остается неизменным. В классе $\mathbf{A_1}$ для независимых периодических вытесняемых задач, реализуемых на классических однопроцессорных системах, наиболее популярно EDF–планирование. В соответствии с EDF–планированием приоритеты заданий снижаются в порядке увеличения абсолютных предельных сроков $d_i^{(j)}$ их выполнения.

В работе [1] показано, что для приложений из независимых вытесняемых периодических задач EDF–планирование оптимально в классе $\mathbf{A_1}$ и что выполнимость приложения обеспечивается в случае справедливости неравенства

$$\sum_{i=1}^{n} u_i \le 1 \qquad (3)$$

6. Симметричные многопроцессорные системы и многоядерные процессоры.

Построение многопроцессорных систем или многоядерных процессоров ставит вопрос о распределении задач и заданий по этим исполнительным ресурсам. Возможны следующие варианты решений:

а) статическое распределение задач между процессорами,

б) миграция исполняемых задач (между процессорами),

в) миграция активных заданий.

В случае варианта *а* процессоры работают независимо друг от друга, проблемы выбора дисциплин планирования заданий, проблемы оценки выполнимости решаются отдельно для каждого процессора

Труды СПИИРАН. 2009. Вып. 8. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2009. Issue 8. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

средствами, разработанными для квазипараллельных систем. В частности, при использовании RM–, DM–, EDF–планирования при оценке выполнимости приложений применимы соответственно формулы (1—3).

В а р и а н т *б* сравнительно более гибок: при очередной активизации задачи τ_i для исполнения очередного ее экземпляра $\tau_i^{(j)}$ из имеющихся процессоров выбирается тот, которому будет поручено исполнение вновь порожденного задания. Выбранный процессор обеспечивает исполнение $\tau_i^{(j)}$ на всем интервале его существования (т.е. исключается передача частично выполненного задания $\tau_i^{(j)}$ другому процессору для продолжения или завершения требуемых $\tau_i^{(j)}$ вычислений). Выбор процессора для задания $\tau_i^{(j)}$ должен осуществляться так, чтобы обеспечивалась своевременность исполнения и всех уже порученных ему заданий, и вновь порожденного задания $\tau_i^{(j)}$. То есть каждая входящая в состав приложения задача не привязана жестко к конкретному процессору, в ходе работы системы различные ее экземпляры могут исполняться различными процессорами – в этом смысле в ходе работы системы имеет место миграция задач между процессорами. Дополнительная гибкость в размещении порождаемых заданий может обеспечить более высокую эффективность использования процессорного времени.

Наибольшую гибкость использования процессорного времени обеспечивает в а р и а н т *в*, разрешающий миграцию активного задания в рамках интервала его существования с одного процессора (процессорного ядра) на другой процессор (ядро). Такая миграция может происходить после вытеснения задания в момент его последующего восстановления. Использование варианта реализации системы с миграцией заданий возможно в том случае, если этот вариант должным образом поддерживается соответствующей аппаратной архитектурой. Для того, чтобы задания могли мигрировать между процессорами, эти процессоры должны работать в едином адресном пространстве: коды программ и данных, (а также используемые заданиями внешние устройства), должны быть в равной мере доступны каждому из процессоров, обеспечивающих исполнение программного приложения. Процессоры аппаратного комплекса, отвечающего этому условию, называются симметричными многопроцессорными системами. Многоядерные процессоры обеспечивают аналогичную симметрию своих процессор-

Труды СПИИРАН. 2009. Вып. 8. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2009. Issue 8. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

ных ядер. Методы планирования и оценки выполнимости приложений, разработанные для симметричных многопроцессорных систем, могут с равным успехом использоваться в случае реализации приложений на многоядерных процессорах.

7. Эффект Дала.

Как отмечено выше, дисциплины RM– и DM–планирования при реализации на классических одноядерных процессорах оптимальны в классе $\mathbf{A_2}$, EDF–планирование оптимально в классе $\mathbf{A_1}$. Вместе с тем, при реализации на многоядерных процессорах эти дисциплины планирования не являются оптимальными. При использовании многоядерного процессора применение любой из перечисленных дисциплин планирования для определенной конфигурации приложения может гарантировать выполнимость всех составляющих приложение задач только при незначительной суммарной нагрузке на многоядерный процессор: авторами [3] приведен следующий пример конфигурации многозадачного приложения для многоядерной системы:

$$\begin{aligned}&(C_1 = 2\varepsilon,\ T_1 = 1),\ (C_2 = 2\varepsilon,\ T_2 = 1),...\\ &...,\ (C_m = 2\varepsilon,\ T_m = 1),\ (C_{m+1} = 1,\ T_{m+1} = 1+\varepsilon),\end{aligned} \tag{4}$$

где значение ε незначительно больше нуля.

Нетрудно убедиться, что при использовании RM–планирования (со статическими приоритетами), или EDF–планирования (с динамическими приоритетами) на процессоре с m ядрами такое приложение оказывается невыполнимым, хотя суммарная нагрузка почти в m раз меньше общей мощности m -ядерного процессора.

Отмеченный эффект проявляется в приложениях, содержащих и «легкие», и «тяжелые» задачи (при $C_i \le 1/2$ задачу τ_i называют *легкой*, при $C_i > 1/2$ задачу τ_i называют *тяжелой*).

Из приведенного примера следует, что использование RM–, DM– и EDF–планирования при реализации на многоядерных процессорах требует применения специфических методов оценки выполнимости приложений.

8. UB-тесты для многоядерных процессоров.

В работе [4] приведены методы оценки выполнимости приложений на многоядерных процессорах при использовании RM–, DM– и EDF–планирования. Показано, что при использовании EDF–планирования с m –ядерным процессором выполнимость приложения гарантируется, если суммарная нагрузка от n периодических вытесняемых задач отвечает неравенству

Труды СПИИРАН. 2009. Вып. 8. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2009. Issue 8. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

$$\sum_{i=1}^{n} u_i \le m(1-\lambda)+\lambda, \qquad \lambda = \max\{u_i \mid i = 1,...,n\}. \tag{5}$$

Отметим, что в частном случае, при $m = 1$ (т.е. в случае одноядерного процессора) неравенство (4) превращается в неравенство (3). Отметим также, что из (5) следует невыполнимость приложения с конфигурацией (4) при суммарной нагрузке, почти в m раз меньшей общей мощности m –ядерного процессора.

При использовании RM–планирования с m –ядерным процессором выполнимость приложения n периодических вытесняемых задач без буферизации заданий гарантируется лишь при нагрузке, существенно меньшей, чем в случае EDF–планирования:

$$\sum_{i=1}^{n} u_i \le \frac{m(1-\lambda)}{2}+\lambda; \qquad \lambda = \max\{u_i \mid i = 1,...,n\} \tag{6}$$

Проверка справедливости неравенства (6) играет роль UB–теста для многоядерных систем аналогично тому, как неравенство (1) играет ту же роль для одноядерных систем.

В работе [5] приведен UB-тест для RM-планирования приложений со *сверхлегкими задачами*. Нагрузка от сверхлегкой задачи должна быть ограничена неравенством

$$u_i \le m/(3m-2) \tag{7}$$

В [5] показано, что для приложения из n периодических вытесняемых задач, для каждой из которых справедливо неравенство (7), при RM-планировании без буферизации с m -ядерным процессором в качестве UB-теста можно использовать неравенство

$$u_i \le m^2/(3m-2) \tag{8}$$

9.Адаптация RM-планирования для многоядерных процессоров.

Эффект Дала препятствует непосредственному использованию RM-планирования для систем с многоядерными процессорами. Между тем, в классе $\mathbf{A_2}$ эта дисциплина планирования является наиболее популярной среди разработчиков прикладных программных комплексов для СРВ, реализуемых на классических одноядерных процессорах. Ее принадлежность классу $\mathbf{A_2}$ обеспечивает

– *стабильность* (гарантированно своевременное обслуживание группы наиболее приоритетных задач [6]),

– *предсказуемость* (сохранение выполнимости при досрочном выполнении задач [7]),

Труды СПИИРАН. 2009. Вып. 8. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2009. Issue 8. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

– сравнительно невысокое число переключений контекста между активными задачами,

– приемлемую эффективность (сравнительно высокий уровень загрузки процессора прикладными задачами) [6].

Этим вызвана актуальность поисков такой дисциплины планирования класса $\mathbf{A_2}$, которая обладала бы подобными достоинствами и могла бы использоваться в системах с многоядерными процессорами.

В работе [5] для многоядерных процессоров предлагается метод планирования класса $\mathbf{A_2}$, который представляет собой модификацию RM–планирования и назван авторами RM_US–планированием.

При использовании RM_US–планирования множество $\{\tau_1, \tau_2, \ldots, \tau_n\}$ составляющих приложение задач разбивается на два подмножества:

1) старшие задачи $\{\tau_1, \tau_2, \ldots, \tau_k\}$ — это подмножество состоит из задач с нагрузкой $u_i > m/3m - 2$,

2) младшие задачи $\{\tau_{k+1}, \tau_{k+2}, \ldots, \tau_k\}$ — это подмножество состоит из задач с нагрузкой $u_i \le m/3m - 2$.

Любая из старших задач имеет более высокий приоритет в сравнении с любой из младших задач. Между старшими задачами приоритеты распределяются произвольно. Между младшими задачами приоритеты распределяются по убыванию частоты (как при RM–планировании).

Доказано, что при использовании RM_US–планирования в качестве UB–теста может использоваться неравенство (8). Таким образом, результат, полученный для оценки выполнимости приложений из сверхлегких задач распространяется на приложения, состоящее из задач с произвольными значениями нагрузки $1 \ge u_i > 0$.

10. Время отклика задач для дисциплин класса $\mathbf{A_2}$.

При большом m использование рассмотренных дисциплин планирования класса $\mathbf{A_2}$ с ориентацией на UB–тест типа (8) связано с обеспечением троекратного запаса производительности процессора относительно суммарной нагрузки от составляющих приложение задач. Это связано, в частности, с тем, что выполнение неравенства (8) гарантирует своевременность выполнения приложений с любой конфигурацией параметров задач. Вместе с тем, конкретное приложение может иметь конфигурацию, не требующую такого большого запаса производительности. Уточнение требуемой производительности многоядерного процессора для приложения с конкретной конфигурацией

Труды СПИИРАН. 2009. Вып. 8. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2009. Issue 8. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

может быть выполнено путем вычисления оценок времени отклика для каждой из составляющих приложение задач.

На величину $r_j^{(k)}$ интервала существования задания $\tau_j^{(k)}$ оказывает влияние объем суммарной нагрузки $W(\tau_j^{(k)})$ на процессор от экземпляров задач τ_i на интервале $[(t_{\text{arr}}(\tau_j^{(k)}), t_{\text{end}}(\tau_j^{(k)})]$. При реализации приложений из независимых вытесняемых задач на классических однопроцессорных системах влияние задач τ_1, τ_2, …, τ_{j-1} на величину $r_j^{(k)}$ зависит только от вклада $w_i(\tau_j^{(k)})$ каждой из задач τ_i $(1 \le i \le j)$ в суммарную нагрузку $W(\tau_j^{(k)})$.

При реализации приложений многоядерными процессорами величина $r_j^{(k)}$ зависит еще и от того, как эта нагрузка распределяется в рамках интервала $[(t_{\text{arr}}(\tau_j^{(k)}), t_{\text{end}}(\tau_j^{(k)})]$. Заметим, что на интервале $[(t_{\text{arr}}(\tau_j^{(k)}), t_{\text{end}}(\tau_j^{(k)})]$ задание $\tau_j^{(k)}$ находится в состоянии «ожидание ресурса процессора» в течение $r_j^{(k)} - c_j^{(k)}$ единиц времени, когда все m ядер заняты обслуживанием более приоритетных заданий. То есть, пока $\tau_j^{(k)}$ вытеснено, процессор выполняет для задач τ_1, τ_2, …, τ_{j-1} вычисления объемом $m(r_j^{(k)} - c_j^{(k)})$. В наихудшем (для времени отклика $\tau_j^{(k)}$) случае весь объем вычислений для задач τ_1, τ_2, …, τ_{j-1} приходится на интервалы, когда $\tau_j^{(k)}$ вытеснено. В этом случае $m(r_j^{(k)} - c_j^{(k)}) = W(\tau_j^{(k)})$; в общем случае $m(r_j^{(k)} - c_j^{(k)}) \le W(\tau_j^{(k)})$. Отсюда имеем

$$r_j^{(k)} \le c_j^{(k)} + \frac{W(\tau_j^{(k)})}{m} \le c_j^{(k)} + \frac{\sum_{i=1}^{j-1} w_i(\tau_j^{(k)})}{m} \le C_j + \frac{\sum_{i=1}^{j-1} w_i^*(\tau_j^{(k)})}{m}. \qquad (9)$$

где $w_i^*(\tau_j^{(k)})$ – максимально возможный вклад задачи τ_i в $W(\tau_j^{(k)})$.

Сценарий системных событий, приводящий к наибольшему значению $W(\tau_j^{(k)})$, называют *критическим*. Ключевым моментом для отыскания оценки времени отклика для задачи τ_j является построение

Труды СПИИРАН. 2009. Вып. 8. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2009. Issue 8. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

критического сценария системных событий, связанных с исполнением более приоритетных задач τ_1, τ_2, …, τ_{j-1}.

Вклад τ_i в $W(\tau_j^{(k)})$ максимален, если на интервале $[(t_{\text{arr}}(\tau_j^{(k)}), t_{\text{end}}(\tau_j^{(k)})]$ выполняются следующие условия:

а) моменты порождения соседних экземпляров $\tau_i^{(l)}$ и $\tau_i^{(l+1)}$ задачи τ_i максимально приближены друг к другу, то есть, отстоят ровно на T_i;

б) значения $c_i^{(l)}$ для заданий типа τ_i равны C_i;

в) время отклика первого на интервале $[(t_{\text{arr}}(\tau_j^{(k)}), t_{\text{end}}(\tau_j^{(k)})]$ задания типа τ_i равно R_i, время отклика последнего равно C_i;

г) Фазы φ_i заданий типа τ_i таковы, что момент завершения последнего из них на интервале $[(t_{\text{arr}}(\tau_j^{(k)}), t_{\text{end}}(\tau_j^{(k)})]$ отличается от $t_{\text{end}}(\tau_j^{(k)})$ на пренебрежимо малую величину.

Включение условий *в* и *г* обусловлено необходимостью учитывать возможность внесения вклада в $W(\tau_j^{(k)})$ тем из заданий типа τ_i, которое было порождено до момента $t_{\text{arr}}(\tau_j^{(k)})$. Если этот вклад нулевой, то как и в случае одноядерного процессора,

$$w_i(\tau_j^{(k)}) = C_i \left\lceil r_j^{(k)} \Big/ T_i \right\rceil,$$

но этот вклад будет ненулевым, если

$$(r_j^{(k)} - C_i) - (C_i \left\lceil r_j^{(k)} \Big/ T_i \right\rceil - R_i) > 0, \qquad (10)$$

тогда левая часть неравенства и будет составлять ту добавку, которая привносится в $w_i(\tau_j^{(k)})$ заданием, порожденным до момента $t_{\text{arr}}(\tau_j^{(k)})$. Таким образом, максимально возможный вклад $w_i^*(\tau_j^{(k)})$ задачи τ_i в $W(\tau_j^{(k)})$ равен

Труды СПИИРАН. 2009. Вып. 8. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2009. Issue 8. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

$$w_i^*(\tau_j^{(k)}) = C_i\left\lceil r_j^{(k)} \Big/ T_i \right\rceil + \max\left\{0, (r_j^{(k)} - C_i) - (C_i\left\lceil r_j^{(k)} \Big/ T_i \right\rceil) - R_i\right\}. \quad (11)$$

Учитывая (9), и суммируя (11) по всем более приоритетным задачам, заключаем, что максимальное значение R_i времени отклика задачи τ_j может быть найдено из уравнения

$$R_j = C_i + \frac{\sum_{i=1}^{j-1}\left(C_i\left\lceil r_j^{(k)} \Big/ T_i \right\rceil + \max\left\{0, (R_j - C_i) - (C_i\left\lceil R_j \Big/ T_i \right\rceil) - R_i\right\}\right)}{m}. \quad (12)$$

Подытоживая, отмечаем, что для наиболее приоритетных задач ($i \le m$) время отклика R_i задачи τ_i равно C_i. Для остальных задач ($i > m$) время отклика задачи τ_i выражается рекуррентным соотношением (12). Решение соотношения (12) находится путем вычисления последовательных приближений $R_i^{(0)}$, $R_i^{(1)}$, … с начальным значением $R_i^{(0)} = C_i$. Значения R_i ищутся последовательно для задач τ_{m+1}, τ_{m+2} и так далее в порядке снижения приоритетов задач.

Представленный метод оценки времени отклика задач для приложений на базе многоядерных процессоров ориентирован на класс систем со статическими приоритетами периодических вытесняемых задач, с миграцией исполняемых заданий между ядрами процессора. В рамках систем со статическими приоритетами задач возможность оценивать время отклика задач позволяет проверять выполнимость задач и приложений в целом при использовании любой дисциплины планирования класса $\mathbf{A_2}$.

11. Pfair–планирование.

Как следует из неравенства (3), применение EDF–планирования при использовании классических одноядерных процессоров гарантирует своевременность выполнения задач вплоть до 100%–го использования производительности процессора. Для многоядерных процессоров в классе приоритетно–регулируемых дисциплин планирования такие эффективные дисциплины планирования неизвестны. Но в классе $\mathbf{A_0}\backslash\mathbf{A_1}$ дисциплин планирования с переменными приоритетами заданий найдены дисциплины, обеспечивающие 100%–ое использование

Труды СПИИРАН. 2009. Вып. 8. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2009. Issue 8. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

производительности многоядерного процессора при гарантированной своевременности выполнения задач. Эти дисциплины планирования в различных вариантах опираются на идею Pfair–планирования — пропорционально–равномерного планирования, иллюстрируемую на рис. 1, *а*.

На графике рис. 1, *а* представлены варианты расходования процессорного времени на интервале существования задания $\tau_i^{(j)}$ от момента $t_{\text{arr}}(\tau_i^{(j)})$ порождения $\tau_i^{(j)}$ до момента $d_i^{(j)} = (t_{\text{arr}}(\tau_i^{(j)}) + D_i$. Оси ординат соответствует объем $e(t)$ процессорного времени, израсходованного заданием к моменту времени t.

В числе возможных вариантов порядка расходования процессорного времени при условии успешного (своевременного) выполнения задания $\tau_i^{(j)}$ возможны два крайних случая:

1) максимально быстрое завершение $\tau_i^{(j)}$,

2) максимально возможная отсрочка начала исполнения задания.

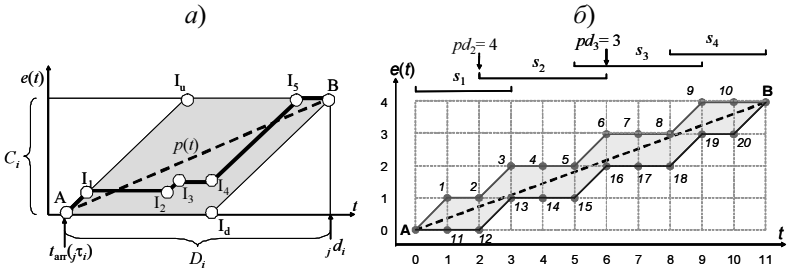


Рис. 1. Pfair-графики исполнения заданий.

а – идеальный Pfair-график,

б – варианты реального Pfair-графика.

Если задание имеет наивысший приоритет на протяжении интервала своего существования, то соответствующий ему график расходования процессорного времени пройдет через точки A, I_u и B — т.е. $\tau_i^{(j)}$ будет выполнено максимально быстро (верхний путь расходования процессорного времени).

Диаметрально противоположный сценарий обслуживания задания $\tau_i^{(j)}$ процессором соответствует максимально возможной задержке

Труды СПИИРАН. 2009. Вып. 8. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2009. Issue 8. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

начала исполнения $\tau_i^{(j)}$. В этом случае соответствующий путь на графике расходования процессорного времени проходит через точки A, I_d и B (нижний путь расходования процессорного времени).

Между верхним и нижним путями заключен параллелограмм, в рамках которого лежат все возможные пути, соответствующие успешному выполнению задания $\tau_i^{(j)}$. Высота параллелограмма равна объему C_i процессорного времени, требуемого для выполнения $\tau_i^{(j)}$, а его длина равна относительному предельному сроку D_i выполнения $\tau_i^{(j)}$.

Каждый из возможных путей имеет вид ломаной линии с горизонтальными участками (они соответствуют тем интервалам времени, когда процессор занят исполнением других заданий) и наклонными участками (они идут под углом 45° и соответствует тем интервалам времени, когда $\tau_i^{(j)}$ является текущим заданием). В качестве примера на рис. 1, *а* жирной линией представлен график исполнения задания, которое начинает исполняться в момент своего порождения, дважды вытесняется (точки I_1 и I_3), дважды возобновляется (точки I_2 и I_4) и досрочно завершается в точке I_5.

Диагональ параллелограмма (пунктирная линия на рис. 1, *а*) соответствует некому идеальному пути $p(t)$, на котором исполнение задания движется равномерно от точки A к точке B.

При $D_i = C_i$ параллелограмм вырождается в прямую линию, соответствующую единственно возможному пути своевременного завершения задания $\tau_i^{(j)}$.

При $D_i > C_i$ диагональ $p(t)$ представляет собой абстракцию пути исполнения задания, пути нереального, поскольку наклонные участки реального пути должны идти строго под углом 45°.

Пропорционально–равномерное планирование (Pfair–планирование), отличается тем, что графики, соответствующие каждому из составляющих приложение заданий, идут в определенной мере близко к диагонали $p(t)$ параллелограмма. То есть в каждый момент времени t из интервала существования задания $\tau_i^{(j)}$ объем израсходованного для $\tau_i^{(j)}$ процессорного времени примерно пропорционален длине пройденного участка интервала существования $\tau_i^{(j)}$ с коэф-

Труды СПИИРАН. 2009. Вып. 8. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2009. Issue 8. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

фициентом, равным C_i/D_i. Допустимые рамки отклонения реального графика от $p(t)$ различаются для рассматриваемых ниже методов Pfair-планирования с квантованием и без квантования времени.

Пропорционально–равномерное планирование «справедливо» (fair) в том смысле, что, с одной стороны, каждое задание получает процессорное время без чрезмерного запоздания, но с другой — и без чрезмерного опережения в сравнении с $p(t)$. Чрезмерное запоздание чревато несвоевременным завершением $\tau_i^{(j)}$, а чрезмерное опережение приводит к задержке исполнения какого-то из вытесненных кооперативных заданий (что чревато несвоевременностью исполнения вытесненного задания) [8]. Основная идея, которая привела к появлению методов Pfair–планирования, состоит в том, что при таком «справедливом» распределении процессорного времени реализуется наиболее полное использование мощности процессора при обеспечении своевременного выполнения всех заданий [9].

12. Pfair–планирование с квантованием времени.

Квантование времени означает, что процессорное время выделяется активным заданиям порциями (квантами) равной длины. Принятая дисциплина планирования задает порядок приоритетов активных заданий, в соответствии с ним определяется, каким из активных заданий следует выделить очередной квант процессорного времени.

Таким образом дисциплины планирования с квантованием времени относятся к классу $\mathbf{A_0/A_1}$ дисциплин планирования с переменными приоритетами заданий. Как уже отмечалось, этому классу принадлежит RR–планирование, при котором вновь порожденное задание и задание, только что израсходовавшее предоставленный квант процессорного времени, становятся наименее приоритетными, а наивысшим приоритетом наделяется задание, дольше других ожидавшее предоставления ресурса процессора.

В случае Pfair–планирования кванты процессорного времени выделяются заданиям в таком порядке, который обеспечивает для каждого задания $\tau_i^{(j)}$ в любой точке его интервала существования выполнение неравенства

$$-\Delta < e_i^{(j)}(t) - p_i(t) < \Delta \qquad (13)$$

где $e_i^{(j)}(t)$ — реальный график выделения процессорного времени заданию $\tau_i^{(j)}$, $p(t)$ — соответствующий идеальный график, Δ — длительность кванта процессорного времени.

Труды СПИИРАН. 2009. Вып. 8. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2009. Issue 8. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

Далее рассматриваются выполняющиеся на m –ядерном процессоре приложения из n задач $\{\tau_1, \tau_2, \ldots, \tau_n\}$ со значениями абсолютных предельных сроков, совпадающими со значениями периодов ($D_i = T_i$) и с суммарной нагрузкой, равной числу ядер процессора $U = m$. Длительность Δ кванта процессорного времени для Pfair–планирования определяется выполнением следующих четырех шагов:

1) единица измерения времени выбирается так, чтобы для каждой из задач значения T_i периодов и значения C_i требуемого объема процессорного времени выражались целыми числами. Это означает, что для требуемой точности представления модельных параметров T_i и C_i единица измерения времени выбирается более или менее мелкой. Ясно, что в любом случае ее нет смысла выбирать мельче, чем продолжительность такта процессора;

2) определяется значение Δ_t наибольшего общего делителя продолжительности периодов T_i для каждой из задач;

3) определяется значение Δ_c наибольшего общего делителя значений C_i требуемого объема процессорного времени для каждой из задач;

4) значение Δ определяется как наибольший общий делитель для Δ_t и Δ_c. При рассмотрении вычислительных моделей выбранную таким образом величину Δ удобно использовать в качестве единицы измерения времени. Тогда неравенства (13) переписываются в виде

$$-1 < e_i^{(j)}(t) - p_i(t) < 1. \qquad (14)$$

Для алгоритмов с квантованием возможные пути на графике расходования процессорного времени проходят по узлам целочисленной решетки типа приведенной на рис. 1, *б* (каждый отрезок реального графика $e_i^{(j)}(t)$ начинается и заканчивается в точках с целочисленными координатами). Для Pfair–графиков задач со значениями модельных параметров $C_i = 4$, $D_i = 11$ верхний путь проходит по узлам 1—10, нижний путь — по узлам 11—20 (см. рис. 1, *б*). Пути промежуточных вариантов Pfair–графиков (насчитывается 109 таких путей) могут переходить с верхнего пути на нижний и обратно.

13. Дисциплина Pfair-планирования PD[2].

Квантование процессорного времени при Pfair–планировании означает разбиение каждой задачи τ_i на C_i *подзадач* (по числу кван-

Труды СПИИРАН. 2009. Вып. 8. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2009. Issue 8. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

тов процессорного времени, требуемых для выполнения τ_i). Например, задача с такими модельными параметрами, как на рис.1, *б*, разбивается на четыре подзадачи s_1, s_2, s_3 и s_4. Из требования (14) следует, что для подзадачи s_1 верхний путь проходит через точки 1 и 2, нижний путь проходит через точки 11 и 12 (единственный промежуточный путь проходит через точки 1 и 12). Для подзадачи s_2 верхний путь проходит через точки 3—5, нижний путь через точки 13—15; (существуют два промежуточных пути).

Для каждой из четырех подзадач (см. рис. 1, *б*) соответствующий верхний и нижний пути обрамляют параллелограмм, заключающий все возможные пути расходования подзадачей процессорного времени. Для подзадачи s_1 этот параллелограмм заключен в интервале времени [0, 3), называемым *окном* подзадачи s_1. Окнами подзадач s_2, s_3 и s_4 являются соответственно интервалы [2, 6), [5, 9) и [8, 11).

График рис.1, *б* соответствует легкой задаче. Отметим, что окна соседних подзадач этой легкой задачи пересекаются. На рис. 2 приведено размещение окон задач, составляющих приложение $\{\tau_1, \tau_2, \tau_3, \tau_4\}$ со следующим набором модельных параметров:

$$(C_1 = 2,\ T_1 = 3), (C_2 = 2,\ T_2 = 4),$$
$$(C_3 = 4,\ T_3 = 6), (C_4 = 2,\ T_4 = 12) \qquad (15)$$

Приложение содержит две легкие (τ_2 и τ_4) и две тяжелые задачи (τ_1 и τ_3). У легкой задачи τ_2 (как и у τ_4) окна составляющих ее подзадач не пересекаются. Тяжелая задача τ_1 состоит из двух подзадач, окна ее подзадач $s_{1,1}$ и $s_{1,2}$ пересекаются. Тяжелая задача τ_3 состоит из четырех подзадач; окно $s_{3,1}$ пересекается с окном $s_{3,2}$ окно $s_{3,3}$ — с окном $s_{3,4}$.

Расстановка относительных приоритетов между активными подзадачами, реализуемая дисциплиной Pfair–планирования PD^2, зависит от значений двух параметров планирования:

1) статического параметра $b(s_{i,k})$,

2) динамического $pd(s_{i,k}, t)$.

В точке планирования *t* дисциплиной PD^2 учитываются значения параметров $b(s_{i,k})$ и $pd(s_{i,k}, t)$ активных подзадач. Подзадача $s_{i,k}$

Труды СПИИРАН. 2009. Вып. 8. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2009. Issue 8. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

является активной в том случае, если момент t приходится на начало или на внутреннюю точку ее окна. При $k>1$ подзадача $s_{i,k}$ является активной в момент t случае, если имеют место два обстоятельства:

1) t приходится на начало или на внутреннюю точку ее окна,

2) исполнение предыдущей подзадачи $s_{i,k-1}$ уже завершено.

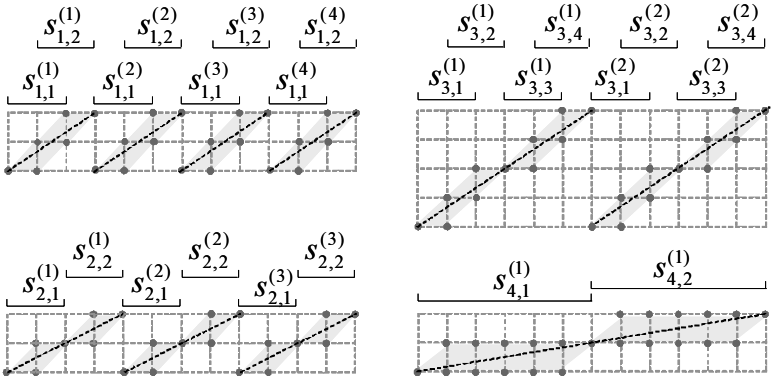


Рис. 2. Конфигурация приложения из двух тяжелых и двух легких задач с суммарной нагрузкой $U=2$.

Значение параметра $b(s_{i,k})$ равно единице, если окно подзадачи $s_{i,k}$ пересекается с окном следующей за $s_{i,k}$ подзадачи той же задачи τ_i, в противном случае значение $b(s_{i,k})$ равно нулю. Роль следующей за $s_{i,k}$ подзадачи играет подзадача $s_{i,k+1}$, если таковая имеется. Если же подзадача является последней из подзадач задачи τ_i, то значение $b(s_{i,k})$ равно нулю. Значения $b=1$ соответствуют подзадачам $s_{1,1}$, $s_{3,1}$ и $s_{3,3}$, приложения с конфигурацией (15); остальным подзадачам этого приложения соответствуют значения $b=0$.

Обозначение динамического параметра pd является сокращением от словосочетания *pseudo–deadline*, введенного авторами работы [10]. Значение $pd(s_{i,k},t)$ в очередной точке планирования t равно времени, остающемуся в момент t до завершения окна, отведенного активному экземпляру подзадачи $s_{i,k}$ т.е. значение $pd(s_{i,k},t)$ для ак-

Труды СПИИРАН. 2009. Вып. 8. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2009. Issue 8. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

тивного экземпляра подзадачи $s_{i,k}$ с окном, завершающимся к моменту времени $d(s_{i,k})$ равно разности $d(s_{i,k}) - t$.

В очередной точке перепланирования (перед каждым квантом времени работы процессора) дисциплина Pfair-планирования должна определять отношение порядка $s_{i,k} \succ s_{j,l}$, задающее относительные приоритеты активных подзадач. В работе [10] определена дисциплина Pfair–планирования PD^2, в соответствии с которой приоритеты подзадач назначаются так, чтобы соблюдались следующие условия:

1) если $pd(s_{i,k},t) < pd(s_{j,l},t)$ то $s_{i,k} \succ s_{j,l}$,

2) если $pd(s_{i,k},t) = pd(s_{j,l},t)$ и $b(s_{i,k}) > b(s_{j,l})$ то $s_{i,k} \succ s_{j,l}$,

3) если $pd(s_{i,k},t) = pd(s_{j,l},t)$, $b(s_{i,k}) = b(s_{j,l}) = 1$ и $s_{i,k+1} \succ s_{j,l+1}$ то $s_{i,k} \succ s_{j,l}$.

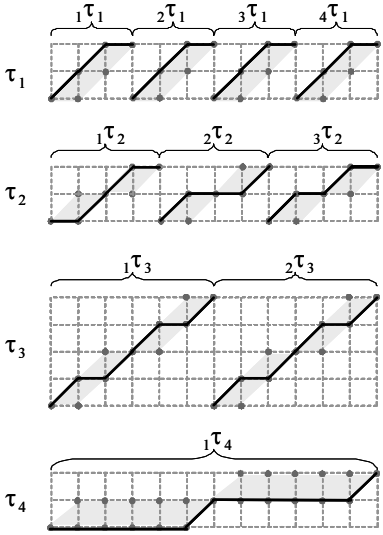
Если для пары подзадач $s_{i,k}$ и $s_{j,l}$ условия 1–3 не выполняются (т.е. если $pd(s_{i,k},t) = pd(s_{j,l},t)$, $b(s_{i,k}) = b(s_{j,l}) = 0$ или $b(s_{i,k}) = b(s_{j,l}) = 1$, но отношение приоритетов подзадач $s_{i,k+1}$ и $s_{j,l+1}$ не определено), то отношение приоритетов подзадач $s_{i,k}$ и $s_{j,l}$ остается неопределенным (выбирается произвольно).

Дисциплина Pfair–планирования PD^2 обеспечивает выполнимость вытесняемых периодических задач для приложений из n задач τ_1, τ_2, …, τ_n со значениями абсолютных предельных сроков, совпадающими с длительностью периодов $D_i = T_i$ и с суммарной нагрузкой, не превышающей числа ядер процессора m. Другими словами, для рассматриваемого класса приложений дисциплина Pfair–планирования PD^2 обеспечивает выполнимость всех задач при 100%–ой загрузке мощности m–ядерного процессора.

Результат применения дисциплины Pfair–планирования PD^2 к приложению с конфигурацией (15) приведен на рис.3. Длина T_{SYS} полного цикла функционирования приложения равна 12 квантам (12 единицам измерения времени) — наименьшему общему кратному периодов задач. Из 12–ти точек перепланирования в 2–х точках (перед 6–и квантами 6 и 12) выполняется по 2 переключения контекста, в остальных точках — по одному. В общем случае плотность пере-

Труды СПИИРАН. 2009. Вып. 8. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2009. Issue 8. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

ключений контекста (среднее число переключений контекста на единицу времени) не может превышать числа m ядер процессора, поскольку в каждой точке перепланирования возможно не более m переключений контекста.



1	$d(\tau_1) = d(\tau_2) = d(\tau_3) < d(\tau_4)$ $b(\tau_1) = b(\tau_3) = 1$	$p(\tau_1) = p(\tau_3)$
2	$d(\tau_2) < d(\tau_1) = d(\tau_3) < d(\tau_4)$	$p(\tau_2) > p(\tau_1) = p(\tau_3)$
3	$d(\tau_3) < d(\tau_2) < d(\tau_4)$	$p(\tau_3) > p(\tau_2) > p(\tau_4)$
4	$d(\tau_1) = d(\tau_3) < d(\tau_4)$ $b(\tau_1) = b(\tau_3) = 1$	$p(\tau_1) = p(\tau_3)$
5	$d(\tau_1) = d(\tau_2) = d(\tau_3) = d(\tau_4)$	$p(\tau_i) = p(\tau_j)$
6	$d(\tau_3) = d(\tau_4)$	$p(\tau_3) = p(\tau_4)$
7	$d(\tau_1) = d(\tau_2) = d(\tau_3) < d(\tau_4)$ $b(\tau_1) = b(\tau_3) = 1$	$p(\tau_1) = p(\tau_3)$
8	$d(\tau_2) < d(\tau_1) = d(\tau_3) < d(\tau_4)$	$p(\tau_2) > p(\tau_1) = p(\tau_3)$
9	$d(\tau_3) < d(\tau_2) < d(\tau_4)$	$p(\tau_3) > p(\tau_2) > p(\tau_4)$
10	$d(\tau_1) = d(\tau_3) < d(\tau_4)$ $b(\tau_1) = b(\tau_3) = 1$	$p(\tau_1) = p(\tau_3)$
11	$d(\tau_1) = d(\tau_2) = d(\tau_3) = d(\tau_4)$	$p(\tau_i) = p(\tau_j)$
12	$d(\tau_3) = d(\tau_4)$	$p(\tau_3) = p(\tau_4)$

Рис. 3. Pfair–планирование PD^2 для приложения с конфигурацией (15).

Аналогичная оценка максимального числа переключений контекста для приложения, состоящего из легких задач, получается из других соображений. Рассмотрим максимально возможный вклад легкой задачи τ_i в суммарную плотность переключений контекста. В рамках одного периода задачи максимально возможное число консервации ее контекста происходит в случае, если для τ_i реализуется нижний, как у задачи τ_4 из рис. 3 (либо верхний, как у задания $\tau_2^{(3)}$), путь расходования процессорного времени. При таком порядке выделения процессорного времени активные кванты (на которых задача τ_i использует процессорное время) максимально равномерно разносятся в рамках периода задачи τ_i и каждый активный квант завершается консервацией контекста задачи. То есть в рамках каждого периода исполнения задачи ее контекст консервируется C_i раз. В таком случае вклад

Труды СПИИРАН. 2009. Вып. 8. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2009. Issue 8. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

τ_i в суммарную плотность переключений контекста равен C_i/T_i, т.е. равен нагрузке u_i на процессор от задачи τ_i. А это означает, что максимальная общая плотность переключений контекста в приложении, состоящем из легких задач, равна суммарной нагрузке U, т.е. числу процессоров m.

Максимально возможная плотность переключений контекста для тяжелой задачи τ_i достигается при максимально равномерном распределении ее пассивных квантов в рамках периода исполнения τ_i. В этом случае каждый из пассивных квантов начинается консервацией контекста τ_i. Тогда плотность переключений контекста τ_i равна $T_i - C_i$, следовательно, вклад тяжелой задачи в суммарную плотность переключений контекста равен $1 - u_i$. Максимальная общая плотность переключений контекста в приложении, состоящем из тяжелых задач, равна $m - U$. В общем случае для приложений, состоящих из k тяжелых задач τ_1, τ_2, …, τ_k и $n-k$ легких задач τ_{k+1}, τ_{k+2}, …, τ_n верхняя граница $S_{\max}$ числа переключений контекста в секунду задается выражением

$$S_{\max} = \frac{\sum_{i=1}^{k}(1-u_i) + \sum_{i=k+1}^{n} u_i}{\Delta},$$

где Δ — длина кванта процессорного времени, выраженная в секундах.

Снижение плотности переключений контекста для легкой задачи возможно путем слияния активных квантов соседних подзадач: подзадача с четным номером объединяется с соседней подзадачей с нечетным номером, активные кванты объединенных подзадач следуют непосредственно друг за другом. При таком распределении активных квантов легкой задачи ее вклад в плотность переключений контекста снижается вдвое.

Снижение плотности переключений контекста для тяжелой задачи возможно путем аналогичного слияния ее пассивных квантов. При этом вклад задачи в плотность переключений контекста также снижается вдвое. Это означает, что плотность переключений контекста при Pfair-планировании с квантованием времени не может быть ниже чем

$$S_{\min} = S_{\max}/2.$$

Труды СПИИРАН. 2009. Вып. 8. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2009. Issue 8. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

14. Pfair–планирование без квантования времени.

В работе [11] представлен вариант пропорционального планирования с другим подходом к определению равномерности распределения процессорного времени между составляющими приложение периодическими задачами τ_1, τ_2, …, τ_n. Порождения ряда заданий $\tau_i^{(1)}$, $\tau_i^{(2)}$, …, $\tau_i^{(j)}$, … для каждой из задач является системными событиями, возникающим в моменты времени 0, T_i, $2T_i$, …, jT_i, … (в общем случае при ненулевых значениях фазы задания порождаются в моменты $jT_i + \varphi$). Абсолютный предельный срок $d_i^{(j)}$ каждого из заданий $\tau_i^{(j)}$ совпадает с моментом $(j+1)T_i$ порождения следующего однотипного задания $\tau_i^{(j+1)}$. Этими системными событиями ход времени исполнения приложения разбивается на сегменты: границы сегментов соответствуют системным событиям (порождениям каких-то заданий); внутри сегмента события задания не порождаются.

Специфика подхода к определению равномерности распределения процессорного времени, предложенная в [11], состоит в том, что пропорциональность израсходованного задачами процессорного времени соблюдается только на стыке сегментов. Единственное требование к распределению процессорного времени внутри сегмента состоит лишь в том, чтобы к моменту завершения очередного сегмента каждое из активных заданий израсходовало объем процессорного времени, равный $u_i L$, где L — длина сегмента.

15. LLRET–планирование.

В работе [11] определена дисциплина внутрисегментного планирования LLRET (Largest Local Remaining Execution Time), обеспечивающая выполнение этого требования. При LLRET–планировании приоритет задачи тем выше, чем больше объем недовыполненных (в рамках сегмента) вычислений для этой задачи. При равенстве объемов недовыполненных вычислений относительные приоритеты задач назначаются произвольно.

В рамках сегмента размещаются до $n+1$ точек планирования (где n — число задач). Точки планирования разбивают сегмент на участки. Первая точка планирования размещается в начале сегмента; в этой точке назначаются приоритеты активных заданий для первого участка сегмента. В начале сегмента объем оставшихся до завершения сегмента вычислений для задачи τ_i равен $u_i L$. Таким образом, в пер-

Труды СПИИРАН. 2009. Вып. 8. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2009. Issue 8. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

вой точке планирования процессорное время предоставляется задачам с наибольшим значением нагрузки u_i.

Необходимость переназначения приоритетов заданий появляется в результате возникновения внутрисегментных событий. Различаются две разновидности внутрисегментных событий:

1) *освобождение* ядра процессора текущей задачей τ_i. Происходит в момент завершения необходимого в рамках сегмента для τ_i объема вычислений $u_i L$ (график выполнения τ_i достигает верхнего ребра параллелограмма допустимых путей). Освободившееся ядро предоставляется наиболее приоритетной из задач, ожидающих предоставления ресурса процессорного времени;

2) *захват* ядра процессора задачей τ_i. Происходит в тот момент, когда объем недовыполненных ею вычислений становится равным интервалу времени, оставшемуся до конца сегмента (график его выполнения достигает правого ребра параллелограмма допустимых путей). Для реализации захвата вытесняется наименее приоритетная из текущих задач.

Рис. 4 иллюстрирует LLRET–планирование на примере приложения из четырех задач τ_1, τ_2, τ_3, τ_4 с набором модельных параметров (15). Приложение исполняется на двухядерном процессоре.

На первом участке сегмента рис. 4 процессорное время выделяется задачам τ_1 и τ_3, с наиболее высокими значениями нагрузки $u_1 = u_3 = 2/3$. На этом участке графики использования процессорного времени задачами τ_1 и τ_3 идут по левому ребру параллелограмма допустимых путей, задачи τ_2 и τ_4 не используют процессор, их графики идут по нижнему ребру.

Первый участок сегмента на рис. 4 заканчивается захватом ядра процессора задачей τ_2. Обе текущие задачи τ_1 и τ_3, имеют равные приоритеты (к концу первого участка у каждой из них остается по $L/6$ недовыполненных вычислений). Вытесняется задача τ_3.

Второй участок (второй кадр рис. 4) заканчивается освобождением ядра процессора задачей τ_1. У каждой из задач τ_3 и τ_4, ожидающих предоставления ресурса процессорного времени, остается по

Труды СПИИРАН. 2009. Вып. 8. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2009. Issue 8. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

$L/6$ недовыполненных вычислений. Освободившееся ядро передается задаче τ_3.

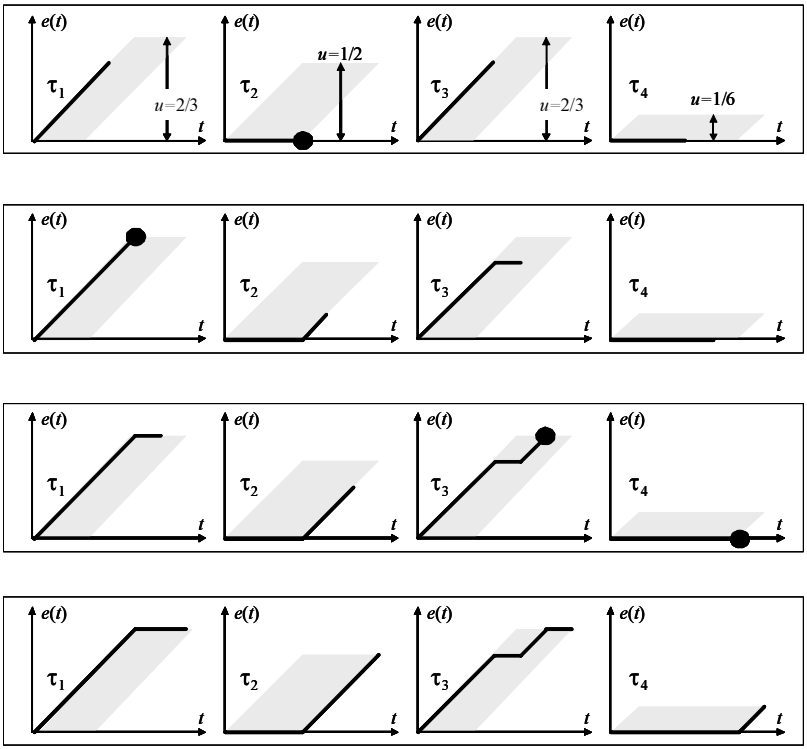


Рис. 4. Внутрисегментное планирование:

Третий участок (третий кадр рис. 4) завершается одновременным возникновением двух внутрисегментных системных событий: освобождение ядра процессора задачей τ_3 и захват ядра процессора задаче τ_4. Последний кадр рис. 4 иллюстрирует успешное завершение вычислений задачами τ_2 и τ_4 на последнем участке сегмента.

В работе [11] доказана оптимальность LLRET-планирования в классе $\mathbf{A_0}$ (для каждой из задач приложения обеспечивается завершение требуемого в рамках сегмента объема вычислений при сумме нагрузок от всех задач приложения, не превышающей числа ядер процессора).

Труды СПИИРАН. 2009. Вып. 8. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2009. Issue 8. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

Каждая из задач вызывает в рамках сегмента по одному из внутрисегментных событий — либо захват ядра, либо освобождение ядра. Других причин возникновения внутрисегментных событий нет. Следовательно, каждый сегмент содержит ровно n внутрисегментных событий (по числу задач). Если внутрисегментные события не сливаются (как в конце 3-го участка на рис. 4), то внутри каждого сегмента имеется n точек перепланирования, в каждой из них выполняется переключение контекста. Кроме того, начало сегмента также является точкой перепланирования и в этой точке может выполняться до m переключений контекста.

Число сегментов на единицу времени не может быть больше, чем $nT_{\min}$, где $T_{\min}$ — период наиболее часто активизируемой задачи. Отсюда следует, что при Pfair-планировании без квантования времени средняя частота следования точек перепланирования не превышает величины

$$P_{\max} = \frac{n(1+n)}{T_{\min}},$$

а средняя частота переключений контекста не превышает величины

$$S_{\max} = \frac{n(m+n)}{T_{\min}}$$

В конкретных частных случаях границы для средней частоты перепланирования и переключений контекста может быть в разы меньшей. Так, если периоды всех задач кратны $T_{\min}$, то число сегментов на единицу времени в точности равно $1/T_{\min}$ и оценки для $P_{\max}$ и $S_{\max}$ уменьшаются, соответственно, в n раз.

Заключение.

В СРВ, реализуемых на одноядерных процессорах, RM– и EDF–дисциплины планирования оптимальны в классах дисциплин, соответственно, со статическими и с динамическими приоритетами задач. В частности, EDF–планирование гарантирует своевременность выполнения всех задач при стопроцентном использовании процессорного времени. Для симметричных многопроцессорных СРВ и для СРВ на многоядерных процессорах (системах, допускающих миграцию заданий) дисциплины RM– и EDF–планирования не являются оптимальными. Более того, их использование для приложений со специфической конфигурацией модельных параметров (сочетание тяжелых и легких задач) может привести к тому, что значительная часть ядер процессора

Труды СПИИРАН. 2009. Вып. 8. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2009. Issue 8. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

простаивает, несмотря на то, что своевременность выполнения некоторых задач не обеспечивается (эффект Дала).

В последние годы в области исследований, связанных с поиском эффективных методов построения СРВ с миграцией заданий получен ряд существенных результатов в части разработки специальных методов планирования для таких систем, а также в части оценки выполнимости приложений. Особый интерес представляют работы, ориентированные на поиск эффективных методов планирования со статическими приоритетами, поскольку эти методы обеспечивают стабильность и предсказуемость выполнения многозадачных приложений, сравнительно невысокие накладные расходы на реализацию планирования и переключение контекстов. Разработанный в начале текущего десятилетия метод RM_US-планирования, адаптирующий RM-планирование к использованию в системах с миграцией заданий [5] обеспечивает приемлемую эффективность использования многоядерных процессоров — благодаря различию подходов к назначению приоритетов легким и тяжелым задачам авторам метода удалось обойти проблемы, связанные с эффектом Дала.

Метод RM_US-планирования ориентирован на приложения, в которых предельные сроки выполнения задач совпадают с их периодами. Если в системе с постоянными приоритетами такое совпадение не имеет места, то для обеспечения проверки выполнимости приложения необходимо располагать методом оценки времени отклика задач. Такой метод представлен в настоящей статье.

Для классических однопроцессорных систем известна дисциплина планирования (EDF-планирование), обеспечивающая высокую (вплоть до стопроцентной) эффективность использования вычислительных ресурсов при сохранении гарантий своевременности выполнения всех задач. Разработки, направленные на поиск методов планирования, обладающих аналогичной эффективностью для систем с миграцией заданий, привели к отысканию пропорционально-равномерных дисциплин планирования (Pfair-планирования). Были предложены методы Pfair-планирования с квантованием времени и без квантования времени. Характерной чертой этих методов является непостоянство приоритетов заданий, что приводит к сравнительно высокой плотности точек перепланирования и высокой частоте переключения контекстов. Как показано в заключительном разделе настоящей статьи, частота переключений контекста при Pfair-планировании с квантованием времени определяется соотношением между суммарными потребностями процессорного времени между тяжелыми и легкими

Труды СПИИРАН. 2009. Вып. 8. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2009. Issue 8. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

задачами. При Pfair-планировании без квантования времени имеет место квадратичная зависимость частоты следования точек перепланирования и частоты переключения контекстов от числа составляющих приложение задач.

Литература

1. *C.Liu, J.Layland.* Scheduling Algorithms for Multiprocessing in a Hard Real-Time Environment // Journal of the ACM, v.20, n.1, 1973. P. 46–61.
2. *Х.Гома.* UML. Проектирование систем реального времени, параллельных и распределенных приложений UML. Проектирование систем реального времени, параллельных и распределенных приложений. М.: ДМК Пресс, 2002. 480 с.
3. *S.K.Dhall, C.L.Liu.* On a Real-Time Scheduling Problem // Operating Research, v.26, n.1, 1978. P. 127–140.
4. *T.Baker.* Multiprocessors EDF and Deadline Monotonic Schedulability Analysis // Proceedings of 24 IEEE Real-Time Systems Symposium, 2003. P. 120–129.
5. *B.Anderson, S.Daruah, J.Jonson.* Static-Priority Scheduling on Microprocessors // Proceedings of 22 IEEE Real-Time Systems Symposium, 2001. P. 193-202.
6. *A.D.Ferrari.* Real-Time Sceduling Algorithms // Dr.Dobb's Journal. 1994, n.12. P. 60–66
7. *R.Ha, J.W.S.Liu.* Validating timing constrains in multihrocessjr and distributed systems // Proceedings of the IEEE international conference on disteibuted systems, 1994, P.162–171.
8. *S.K.Baruah.* Fairness in Periodic Real-Time Sceduling Algorithms // Proceedings of 16 IEEE Real-Time Systems Symposium, 1995. P. 200–209.
9. *A.Srinivasan, J.Anderson.* Early-Reliase Fair Scheduling // Proceedings of the 12-th Euromicro Conference on Real-Time Systems. 2000. P. 35–43.
10. *A.Srinivasan, J.Anderson.* Optimal rate-based scheduling on multiprocessors // Proceedings of the ACM Symposium on Theory of Computing, 2002. P. 189–198.
11. *H.Cho, B.Ravindran, D.Jensen.* An Optimal Real-Time Scheduling Algorithm for Multiprocessors // Proceedings of 27 IEEE Real-Time Systems Symposium, 2006. P. 101–110.

Никифоров Виктор Викентьевич — д–р техн.наук, проф., ведущий науч. сотр. лаборатории технологий и систем программирования Санкт-Петербургского института информатики и автоматизации РАН. Область научных интересов: программирование систем реального времени, встроенных систем, операционные системы. Число научных публикаций — 80. nik@iias.spb.su; СПИИРАН, 14-я линия В.О., д. 39, г. Санкт-Петербург, 199178, РФ; р.т. +7(848)328-0887.

Nikiforov Victor — Dr.Sci. (Tech.), Prof., Senior researcher, Laboratory for Software Technology and Systems, St. Petersburg Institute for Informatics and Automation of the Russian Academy of Sciences (SPIIRAS). Research interests: methods for real-time software development, embedded systems, operating systems. The number of publications — 80. nik@iias.spb.su; SPIIRAS, 39, 14-th Line V.O., St. Petersburg, 199178, Russia; office phone +7(848)328-0887.

Труды СПИИРАН. 2009. Вып. 8. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2009. Issue 8. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru