Автор перевода: Середа О.Э.

АЛГОРИТМЫ ПЛАНИРОВАНИЯ МНОГОЗАДАЧНОСТИ В ЖЕСТКИХ СИСТЕМАХ РЕАЛЬНОГО ВРЕМЕНИ.

C. L. LIU
Project MAC, Massachusetts Institute of Technology
AND
JAMES W. LAYLAND
Jet Propulsion Laboratory, California Institute of Technology

КРАТКОЕ СОДЕРЖАНИЕ. Проблема многозадачного планирования на одном процессоре изучается с точки зрения характеристик, свойственных программным функциям, которые должны гарантировать обслуживание. Они показывают, что оптимальный фиксированный приоритет планировщика имеет верхний предел для используемого процессора, который может составлять как меньше, так и 70 процентов для больших наборов задач. Это также показывает, что полная загрузка процессора может быть достигнута путем динамического присвоения приоритетов на основе их текущих значений. Сочетание этих двух методов планирования также обсуждается.

Ключевые слова и фразы: многозадачные системы реального времени, планирование, многозадачное планирования, динамическое планирования, задание приоритетов, используемый процессор, deadline driven scheduling.

1. ВВЕДЕНИЕ.

Использование компьютеров для управления и контроля производственными процессами в последние годы значительно расширилось и, вероятно, еще более резко расширится в ближайшем будущем. Часто компьютер используется в таких приложениях, которые разделяются между определенным количеством: срочных проверок, функциями мониторинга и не срочной работой пакетной обработки потока. В других объектах, однако, не без срочных работ, эффективное использование компьютера может быть достигнуто только путем тщательного планирования срочных проверок и функций самомониторинга. Эта последняя группа может быть названа «чистое управление технологическими процессами" и обеспечивает основу для комбинаторного анализа планирования, представленного в этой статье. Изучалось два алгоритма планирования для этого типа программирования; оба приоритетно управляемые и упреждающие; это означает, что обработка любой задачи прерывается по запросу другой более приоритетной задачи. Первый алгоритм изучал использование фиксированного назначения

приоритета и может добиться загрузки процессора порядка 70 процентов или более. Второй алгоритм планирования может добиться полного использования процессора, посредством динамического назначения приоритетов. Сочетание этих двух алгоритмов также обсуждается.

2. ПРЕДПОСЫЛКИ.

Компьютер, управляющий технологическими процессами, выполняет одну или несколько функций управления и мониторинга. Наведение антенны для отслеживания космических аппаратов на орбите является одной из примеров таких функций. Каждая функция, которая будет выполняться, имеет связь с набором из одной или нескольких задач. Некоторые из этих задач выполняются в ответ на события в оборудовании, контролируемом или проверяемом с помощью компьютера. Остальные выполняются в ответ на события в других задачах. Ни одна из задач может быть выполнено до события, которое этого требует. Каждая из задач должна быть завершена до некоторого фиксированного времени, после подачи запроса на ее выполнение. Службы в этот промежуток времени должны гарантировать классификацию среды разработки в "режиме жесткого реального времени» [1], в отличие от "режима мягкого реального времени", где статистическое распределение времени отклика является приемлемым.

Большая часть имеющейся литературы по мультипрограммированию занимается статистическим анализом коммерческих систем разделения времени ([2] содержится обширная библиография). Другая часть работает с подмножеством более интересных аспектов планирования пакетной обработки объекта или смешанной пакетной обработки с разделением времени, обычно в многопроцессорной конфигурации [3-8].Несколько работ непосредственно затрагивают проблемы программирования "режима жесткого реального времени". Манакер [1] выводит алгоритм для генерации планирования задач в среде жесткого реального времени, но его результаты будут ограничены некоторой нереальной ситуацией – только один запрос во времени для всех задач, хотя несколько сроков завершения учитываются. Лампсон обсуждает проблему программного обеспечения по планированию в общих чертах и представляет свой набор Algol процедур по мультипрограммированию, которые могут быть реализованы программно или разработаны в планировщике для специальных целей. Для распределения ресурсов и назначение приоритетов и временных интервалов, он предлагает программу, которая вычисляет оценочное время отклика дистрибутивов, основываясь на информации синхронизации, предоставленной по необходимости программам гарантированного обслуживания. Однако, он не описывает алгоритмы, которые должны использовать программы.

В тексте Мартин [10] описывает ряд систем, которые считаются системами «реального времени» и обсуждает в определенном порядке, проблемы с которыми сталкиваются в их программировании. Описание Мартина тесного управления инженерным контролем, которое должна поддерживаться в течение реального времени разработка программного обеспечения точно отражается в работе Jirauch [11] по автоматическому программному обеспечению системы контроля. Эти работы служат, чтобы подчеркнуть необходимость более системного подхода к разработке программного обеспечения, чем тот, который используется в настоящее время.

3. СРЕДА РАЗРАБОТКИ.

Для получения любых аналитических результатов о поведении программы в среде разработки режима жесткого реального времени, некоторые допущения должны быть сделаны исходя из того, что это среда разработки. Не все из этих допущений являются абсолютно необходимыми, и последствия их строгости будут обсуждаться в следующем разделе.

(A1) запросы для всех задач, для которых существующие жесткие сроки являются периодическими с постоянным интервалом между запросами.

(A2) сроки состоят из возможности выполнения только ограничений - то есть, каждая задача должна завершиться до следующего запроса, который придет.

(A3) задачи являются независимыми в том, что запросы на определенные задачи не зависят от начала или завершения запросов для других задач.

(A4) время выполнения для каждой задачи является постоянным по отношению к этой задаче и не зависит от времени. Время выполнения здесь относится ко времени, которое требуется процессору для выполнения задача без её прерывания.

(A5) Любые непериодические задачи в системе – специальные; они – это процедуры инициализации или восстановления после отказа; они вытесняют периодические задачи в то время как они сами будут запущены.

Допущение (A1) сопоставляется с мнением Мартина [2], но кажется правильным для чистого управления технологическими процессами. Допущение (A2) устраняет проблемы очередей для отдельных задач. По условию (A2)для каждой периферийной функции должно быть значительное количество оборудования буферизации. Любые контрольные циклы, заложенные в компьютере, должны быть разработаны, чтобы позволить, по крайней мере, дополнительный блок задержки. Отметим, что условие (A3) не исключает ситуации, в которой появление задачи τ_J, возможно только определенное (фиксированное) число раз, например N, вхождений задачи τ_i. Такая ситуация может быть смоделирована путем выбора периодов задач τ_i и

τ_J, так что период τ_J составляет N периодов τ_i и N–й запрос τ_i совпадает с 1-м запросом для τ_J и так далее. Время выполнения в условии (A4) можно интерпретировать, как максимальное время обработки задачи. Таким образом, учет времени, необходимого для запроса «наследника» и расходы прерываний могут быть приняты во внимание. Из-за наличия большой основной памяти, из которой программы выполняются и передач, перекрывающих друг друга, между основным и вспомогательным хранилищем и выполнение программы в современных компьютерных системах, условие (A4) должно быть хорошо апроктимировано, даже если оно не является точным. Эти условия позволяют полностью охарактеризовать задачи двумя числами: период запроса и время выполнения. Если не указано иное, в этой работе мы будем использовать $\tau_1, \tau_2, \ldots\ldots\ldots, \tau_m$ для обозначения m периодических задач, с периодом запроса T1, T2, ... , Tm и их время выполнения будет C1, C2, ... , Cm, соответственно. Частота запроса задачи определяется как величина обратная периоду запроса.

Алгоритм планирования представляет собой набор правил, определяющих задачи, которые должны быть выполнены в данный момент. Алгоритмы планирования изученные в данном документе, являются упреждающими и приоритетозависимыми. Это означает, что, когда есть запрос на задачу, которая имеет более высокий приоритет, чем та, которая в настоящее время выполняется, выполняемая задача немедленно прерывается, и запускается запрос на новую задачу(с более высоким приоритетом). Таким образом, спецификации таких алгоритмов представляют собой спецификацию метода назначение приоритетов задач. Алгоритм планирования называется статическим, если приоритеты назначаются задаче, раз и навсегда. Статический алгоритм планирования называется фиксированным алгоритмом приоритетного планирования. Алгоритм планирования называют динамическим, если приоритеты задачи могут меняться от запроса к запросу. Алгоритм планирования называется смешанным алгоритмом планирования, если приоритеты некоторых задач фиксированы, а приоритеты оставшихся задач изменяются от запроса к запросу.

4. АЛГОРИСМЫ ФИКСИРОВАННЫХ ПРИРОРИТЕТОВ.

В этом разделе мы выведем правило назначения приоритетов, которое дает оптимальный статический алгоритм планирования. Важным понятием в определении этого правила является критический момент для выполнения задачи. Крайний срок запроса на задачу определяется как время следующего запроса для той же задачи. Для планирования ряда задач, предусмотренных в соответствии с некоторым алгоритмом планирования, мы говорим,

что происходит переполнение во время t, если t является срок невыполнения запроса. Для данного набора задач, алгоритм планирования является возможным, если задачи планируется так, чтобы переполнение никогда не происходит. Мы определяем время отклика запроса на определенную задачу, как промежуток времени между запросом и ответом на него. Критический момент для задачи определяется как момент времени , когда запрос на выполнение данной задачи будет иметь самое долгое время ответа. Критической зоной времени для задачи является временной интервал между критическим моментом и ответом на соответствующий запрос задачи. Мы имеем следующую теорему.

Теорема 1. Критический момент для любой задачи возникает, когда задача посылает запрос одновременно с запросами всех задач высших приоритетов.

Доказательство. Пусть $\tau_1, \tau_2, \ldots\ldots\ldots, \tau_m$ обозначают ряд одноприоритетных задач, в котором τ_m - задача с более низким приоритетом. Рассмотрим конкретный запрос для τ_m ,который приходит во время t1. Предположим, что между t1 и t1 + Tm, время, когда придет следующий запрос τ_m , запросы задачи τ_i, i < m ,встречаются в t2, t2 + Ti , t2 +2Ti, … … … , t2+kTi, как показано на рисунке 1(Fig. 1). Очевидно, что преимущество τ_m над τ_i приведет к определенным суммам задержки по завершении запроса τ_m , что и произошло во время t1, если запрос τ_m завершен до t2. Кроме того, на рисунке 1 мы видим сразу что выполнение запроса во время t2 не ускорит завершение τ_m. Время выполнения τ_m остается неизменным, либо задерживается на время выполнения приоритетных действий. Следовательно, задержка в завершении τ_m увеличивается при совпадении t2 с t1.

Повторяя рассуждение для всех τ_i , i = 2, … … … , m - 1, мы получаем доказательство теоремы.

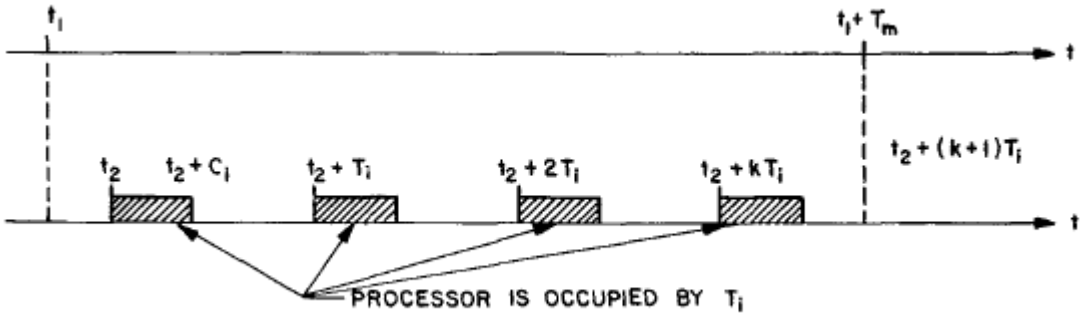

FIG. 1. Execution of τ_i between requests for τ_m

Одно из значений этого результата это то, что простой прямой расчет может определить так или иначе назначенный приоритет, эту возможность даст алгоритм планирования. В частности, если запросы для всех задач в их критические моменты будут выполнены до их соответствующих сроков, то

алгоритм планирования возможен. В качестве примера, рассмотрим набор из двух задач τ_1 *и* τ_2 с T1 = 2, T2 = 5, и C1 = 1, C2 = 1. Если мы позволим τ_1 быть высокоприоритетной задачей, то на рисунке 2 (а) мы видим, что такое приоритетное назначение возможно. Кроме того, значение C2 может быть увеличено не более чем на 2, но не больше, чем показано на рисунке 2 (б). С другой стороны, если мы позволим τ_2 быть более приоритетной задачей, то ни одно из значений C 1 и C2 не может быть увеличено более чем на 1, как показано на рисунке 2 (с).

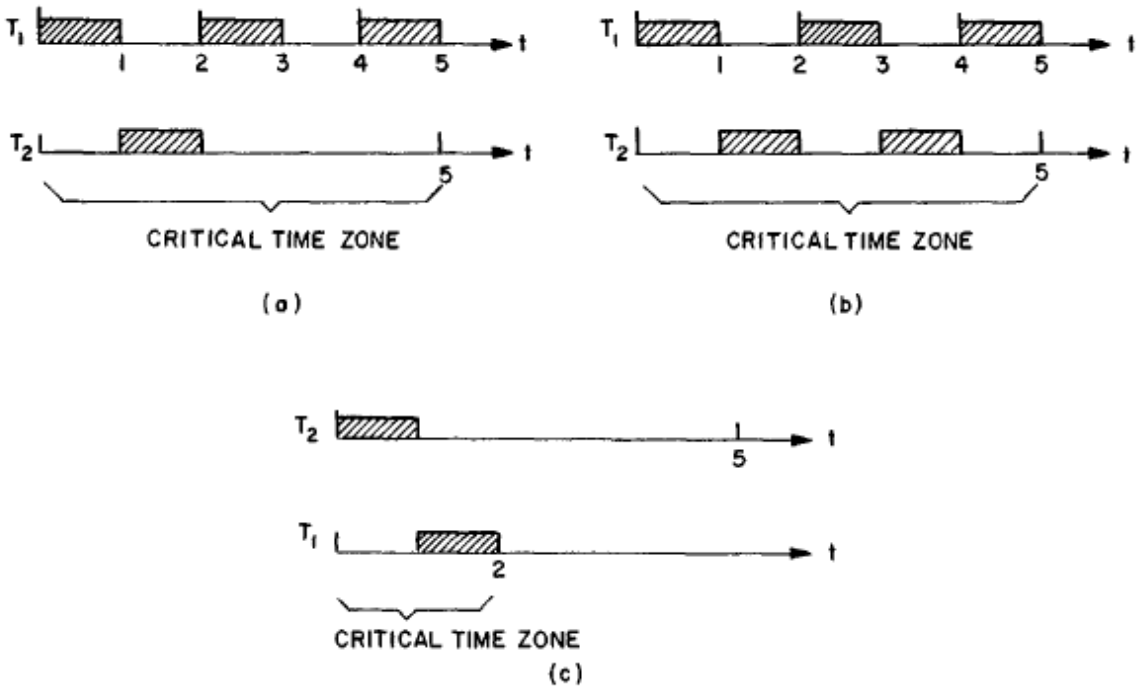

Fig. 2. Schedules for two tasks

В результате в теореме 1 также предполагается приоритетное задание, которое является оптимальным в мысли, выраженной в теореме 2. Пусть нас мотивирует общий результат от рассмотрения при планирования двух задач τ_1 и τ_2. Пусть T1 и T2 периоды запросов задач,при этом T1 <T2. Если мы позволим τ_1 быть более приоритетной то, в соответствии с теоремой 1, неравенство должно быть выполнено: [1] [2]

$$\lfloor T_2/T_1 \rfloor \; C_1 + C_2 \leq T_2 . \qquad (1)$$

Если мы позволим τ_2быть более приоритетной задачей, тогда неравенство должно быть выполнено:

$$C_1 + C_2 \leq T_1 . \qquad (2)$$

Следовательно:

---

[1] Следует отметить, что (1) является необходимым, но недостаточным для обеспечения возможности задания приоритетов.

[2] $\lfloor X \rfloor$ обозначает наибольшее целое число, меньшее или равное x. $\lceil X \rceil$ обозначает наименьшее целое больше или равна x.

$$\lfloor T_2/T_1 \rfloor \; C_1 + \lfloor T_2/T_1 \rfloor \; C_2 \leq \lfloor T_2/T_1 \rfloor \; T_1 \leq T_2 ,$$

(2) следует из (1). Иными словами, всякий раз, когда T1 <T2 и C1, C2 такие, что расписание задачи возможно когда τ_2 приоритетнее чем τ_1, это также возможно когда τ_1приоритетнее τ_2 , но обратное утверждение не верно. Таким образом, мы должны дать более высокий приоритет τ_1 и более низкий приоритет τ_2.

Таким образом, в целом, кажется, что "разумные" правила приоритетного задания назначают приоритеты задачам в соответствии с частотой их запросов, независимо от их выполнения. В частности, задачи с более высокой частотой запросов будут иметь более высокий приоритет. Такое назначение приоритетов известно как частотномонотонное задание приоритетов. Как выяснилось, такое назначение приоритетов является оптимальным в том смысле, что никаких других фиксированных правил назначения приоритета не может запланировать набор задач, которые не могут быть назначены с помощью частотномонотонного назначения приоритетов.

Теорема 2. Если возможно назначение приоритетов существует некоторому множеству задач, то частотномонотонное назначение приоритетов также возможно для решения данного множества задач.

Доказательство. Пусть $\tau_1 , \tau_2 , \ldots\ldots\ldots , \tau_m$ - множество задач с некоторыми назначенными приоритетами. Пусть $\tau_i \, и \, \tau_J$ - две задачи со смежными приоритетами, в этом случае τ_i является более высокоприоритетной. Предположим, что $T_i > T_J$. Давайте поменяем приоритеты между $\tau_i \, и \, \tau_J$.Не трудно заметить, что конечное назначение приоритетов по-прежнему возможно. Так как частотномонотонное назначение приоритетов может быть получено из любой последовательности посредством по-парного обмена приоритетов с высокоприоритетными задачами, мы получаем доказательство теоремы.

5. ПОЛНОЦЕННОЕ ИСПОЛЬЗОВАНИЕ ПРОЦЕССОРА.

На данный момент инструменты позволяют определить верхнюю границу для коэффициента загруженности процессора в системах статического назначения приоритетов. Определим коэффициент загрузки(процессора), как часть процессорного времени, затраченного на выполнение поставленной задачи. Иными словами, коэффициент загрузки равен единице минус время простоя процессора. Если C_i/T_i процессорное время, отведенное на выполнение задачи τ_i, то для выполнения m задач, коэффициент загрузки будет равен:

$$U = \sum_{i=1}^{m} (C_i/T_i).$$

Хотя коэффициент загрузки процессора может быть улучшен за счет увеличения значения C_i или уменьшением значения T_i все равно существует верхняя граница , т.к. все задачи должны выполняться в срок, особенно на критических участках(зонах) времени(см. теорему 1). Понятно, что не целесообразно задавать вопрос, как может существовать маленький коэффициент загрузки процессора . Тем не менее, есть смысл спрашивать, на сколько большим может быть достигнут коэффициент загрузки . Давайте уточним, что мы имеем в виду. В соответствии с заданными приоритетами, говорят, что множество задач в полной мере использует процессор, если назначенные приоритеты являются допустимыми для множества и если увеличение времени выполнения любой из задач множества сделает назначение приоритета невозможным. Для описанного алгоритма статического приоритетного планирования, верхней границей коэффициента использования является минимум среди коэффициентов использования всех множеств задач, которые в полной мере используют процессор. Для всех множеств задач, коэффициент загрузки процессора которых ниже этой границы, существует статическое назначение приоритетов. С другой стороны, загрузка выше этой границы может быть достигнута только, если Т, соответствующих задач связаны между собой.

Так как частотномонотонное назначение приоритетов является оптимальным, коэффициент загрузки, достигнутый частотномонотонным заданием приоритетов для данного множества задач больше или равен коэффициенту загрузки для любого другого способа назначения приоритетов данному множеству. Таким образом, верхняя граница определяется, как наименьшее значение среди коэффициентов загрузки, соответствующих частотномонотонному заданию приоритетов по всем возможным периодам запросов и времени выполнения задач. Граница сначала определяется для двух задач, а затем расширяется на произвольное число.

Теорема 3. Для множества из двух задач со статически назначенными приоритетами, верхняя граница связана с коэффициентом загрузки процессора, как $\boldsymbol{U = 2 * (2^{\frac{1}{2}} - 1)}$ ***или*** $\boldsymbol{U = 2 * (\sqrt{2} - 1)}$**.**

Доказательство Пусть τ_1 *и* τ_2 - две задачи с периодами Т1 и Т2, и временем выполнения С1 и С2, соответственно. Предположим, что Т2> Т1. В соответствии с частотномонотонным назначением приоритетов, τ_1 имеет более высокий приоритет, чем τ_2. В критический момент(в критической зоне) времени τ_2, определяется как $\lceil T2/T1 \rceil$ запросов τ_1 . Теперь сделаем так, чтобы С2 в полной мере использовало доступное время процессора в критической зоне времени. Могут быть 2 случая:

Случай 1. Время выполнения С1 мало на столько, что все запросы τ_1 в критической зоне времени Т2 будут завершены до начала второго запроса τ_2. То есть,

$$C_1 \leq T_2 - T_1 \lfloor T_2/T_1 \rfloor .$$

Таким образом максимально возможное значение C2 будет:

$$C_2 = T_2 - C_1 \lceil T_2/T_1 \rceil .$$

Соответствующий коэффициент загрузки процессора будет равен:

$$U = 1 + C_1[(1/T_1) - (1/T_2) \lceil T_2/T_1 \rceil].$$

В этом случае, коэффициент загрузки процессора монотонно убывает во времени C1.

Случай 2. Время выполнения N-го(N= $\lceil T2/T1 \rceil$) запроса на τ_1 превосходит время второго запроса на τ_2. В этом случае

$$C_1 \geq T_2 - T_1 \lfloor T_2/T_1 \rfloor .$$

Отсюда следует что наибольшее возможное значение C2 равно:

$$C_2 = -C_1 \lfloor T_2/T_1 \rfloor + T_1 \lfloor T_2/T_1 \rfloor \text{ и}$$

и соответствующий коэффициент загрузки равен

$$U = (T_1/T_2) \lfloor T_2/T_1 \rfloor + C_1[(1/T_1) - (1/T_2) \lfloor T_2/T_1 \rfloor].$$

В этом случае, U монотонно возрастает во времени C1.

Минимум U четко проходит на границе между этими двумя случаями. Таким образом

$$C_1 = T_2 - T_1 \lfloor T_2/T_1 \rfloor$$

мы имеем

$$U = 1 - (T_1/T_2)[\lceil T_2/T_1 \rceil - (T_2/T_1)][(T_2/T_1) - \lfloor T_2/T_1 \rfloor]. \qquad (3)$$

Для удобства обозначений[3] примем I= $\lfloor T2/T1 \rfloor$ и f={$T2/T1$}. В таком случае равенство (3) может быть записано как:

$$U = 1 - f(1 - f)/(I + f).$$

Так как U монотонно возрастает на I, минимальное U достигается при минимально возможное значение I, а именно, I = 1. Минимизируя U на f, мы определяем, что на f = 2 ½ - 1, U достигает своего минимального значения, которое равно

$$U = 2(2^{\frac{1}{2}} - 1) \simeq 0.83.$$

Что и требовалось доказать.

Следует отметить, что коэффициент загрузки становится равным 1, если f = 0, т.е. если период запроса задачи более низкого приоритета вмещает периоды запросов других задач. Получим соответствующие оценки для произвольного числа задач. В этот момент, давайте ограничимся

[3] {T2/T1} означает (T2/T1) - $\lfloor T2/T1 \rfloor$, т.е. дробную часть T2/T1.

рассмотрением случая, в котором соотношение между любыми двумя периодами запроса составляет менее 2.

Теорема 4. Для множества задач m с фиксированным порядком приоритетов, и ограничением, на соотношение между любыми двумя периодами запроса меньше, чем 2, верхняя граница коэффициента загрузки процессора равна U = m*($2^{1/m}$ - 1).

Доказательство. Пусть $\tau_1, \tau_2, \ldots\ldots\ldots, \tau_m$ обозначает множество из m задач. Пусть $C_1, C_2, \ldots\ldots\ldots, C_m$ будет время выполнения задач, которые в полной мере используют процессор и минимизируют коэффициент загрузки процессора. Предположим, что $T_m > T_{m-1} > \cdots\cdots\cdots > T_2 > T_1$. Обозначим через U коэффициент загрузки процессора. Мы хотим показать, что

$$C_1 = T_2 - T_1 .$$

Предположим что

$$C_1 = T_2 - T_1 + \Delta, \qquad \Delta > 0.$$

Пусть

$$\begin{aligned} C_1' &= T_2 - T_1 \\ C_2' &= C_2 + \Delta \\ C_3' &= C_3 \\ &\vdots \\ C_{m-1}' &= C_{m-1} \\ C_m' &= C_m . \end{aligned}$$

Очевидно, $C_1', C_2', \cdots, C_{m-1}', C_m'$ также в полной мере используют процессор. Пусть U ' обозначает соответствующий коэффициент загрузки. Мы имеем

$$U - U' = (\Delta/T_1) - (\Delta/T_2) > 0.$$

С другой стороны, предположим что

$$C_1 = T_2 - T_1 - \Delta, \qquad \Delta > 0.$$

Пусть

$$\begin{aligned} C_1'' &= T_2 - T_1 \\ C_2'' &= C_2 - 2\Delta \\ C_3'' &= C_3 \\ &\vdots \\ C_{m-1}'' &= C_{m-1} \\ C_m'' &= C_m . \end{aligned}$$

И снова, $C_1'', C_2'', \cdots, C_{m-1}'', C_m''$ также в полной мере используют процессор. U" обозначает соответствующий коэффициент загрузки. Мы имеем

$$U - U'' = -(\Delta/T_1) + (2\Delta/T_2) > 0.$$

Поэтому, если U действительно минимальный коэффициент загрузки, то

$$C_1 = T_2 - T_1$$

Аналогично, мы можем показать что

$$\begin{aligned} C_2 &= T_3 - T_2 \\ C_3 &= T_4 - T_3 \\ &\vdots \\ C_{m-1} &= T_m - T_{m-1}\,. \end{aligned}$$

Следовательно,

$$C_m = T_m - 2(C_1 + C_2 + \cdots + C_{m-1}).$$

Для упрощения обозначений, пусть

$$g_i = (T_m - T_i)/T_i\,, \qquad i = 1, 2, \cdots, m.$$

Тогда

$$C_i = T_{i+1} - T_i = g_iT_i - g_{i+1}T_{i+1}\,, \qquad i = 1, 2, \cdots, m-1$$

и

$$C_m = T_m - 2g_1T_1$$

И наконец,

$$\begin{aligned} U &= \sum_{i=1}^{m} (C_i/T_i) = \sum_{i=1}^{m-1} [g_i - g_{i+1}(T_{i+1}/T_i)] + 1 - 2g_1(T_1/T_m) \\ &= \sum_{i=1}^{m-1} [g_i - g_{i+1}(g_i + 1)/(g_{i+1} + 1)] + 1 - 2[g_1/(g_1 + 1)] \\ &= 1 + g_1[(g_1 - 1)/(g_1 + 1)] + \sum_{i=2}^{m-1} g_i[(g_i - g_{i-1})/(g_i + 1)]. \end{aligned} \qquad (4)$$

Как и в двухзадачном случае загрузка принимает значение 1, если g_i = 0, для всех i.

Чтобы найти верхнюю границу коэффициента загрузки, рав. (4) должны быть сведены к минимуму по g_J . Это можно сделать, установив первую производную U по отношению к каждому g_J-х равными нулю, и решения результирующей разности уравнений:

$$\partial U/\partial g_j = (g_j^2 + 2g_j - g_{j-1})/(g_j + 1)^2 - (g_{j+1})/(g_{j+1} + 1) = 0, \qquad j = 1, 2, \cdots, m-1. \qquad (5)$$

Допущение $g_0 = 1$ было принято для удобства.

Общее решение уравнения (5) может быть записано

$$g_j = 2^{(m-j)/m} - 1, \qquad j = 0, 1, \cdots, m-1. \qquad (6)$$

Из этого следует

$$U = m(2^{1/m} - 1),$$

что и требовалось доказать.

Для m=2, уравнение (6) сводится к ранее решенному для множества из двух задач без ограничений на периоды запросов. Для m=3, уравнение (6) сводится к

$$U = 3(2^{1/3}) - 1) \simeq 0.78$$

Для большего количества m, U≈ln2.

Ограничение, по которому наибольшее соотношение между периодами запросов менее 2 теорема 4 действительно может быть удалена, в связи со следующим утверждением:

Теорема 5. Для множества задач m с фиксированным порядком приоритета, верхнюю границу для коэффициента загрузки процессора можно определить, как U=m*($2^{1/m}$-1).

Доказательство. Пусть $\tau_1, \tau_2, \ldots, \tau_i, \ldots, \tau_m$ будет множество из m задач, которые в полной мере используют процессор. Обозначим через U соответствующий коэффициент загрузки. Предположим, что для некоторого i, $\lfloor Tm/Ti \rfloor > 1$. Для наглядности, пусть Tm= qTi + r, q> 1 и r ≥ 0. Давайте заменим задачу c на задачу τ'_i ,такую , что T'_i= qTi и $C'_i = C_i$, и увеличим Cm на величину, необходимую чтобы снова в полной мере использовать процессор. Это увеличение больше C_i(q-1), времени в критической зоне времени τ_m, которое занимает τ_i, а не τ'_i. Пусть U 'обозначает коэффициент загрузки такого множества задач. Мы имеем

$$U' < U + [(q-1)C_i/T_m] + (C_i/T'_i) - (C_i/T_i)$$

Или

$$U' \leq U + C_i(q-1)[1/(qT_i + r) - (1/qT_i)].$$

Где q-1>0 и [1/(qTi+r)] – (1/qTi) ≤ 0, U'≤U. Поэтому мы заключаем, , что при определении верхняей границы коэффициента загрузки процессора, мы должны рассматривать только множества задач, в которых соотношение между любыми двумя периодами запроса меньше, чем 2.Следствие из теоремы 4.

6. УМЕНЬШЕНИЕ ГРАНИЦЫ ЗАГРУЗКИ.

В предыдущем разделе было показано, что верхняя граница загрузки процессора обусловленная требованием к гарантированному обслуживанию в режиме реального времени может подойти в неравенство (2) и для больших множеств задач. Желательно, найти способы, чтобы улучшить эту ситуацию, т.к практическую стоимость переключения между задачами, еще нужно подсчитать. Одним из самых простых способов сделать гнаицу загрузки равной 1, это сделать, чтобы $\{Tm/Ti\}$=0 для i=1,2, … , m-1. Так как это может быть сделано не всегда, альтернативное решение является буферизировать задачу τ_m и, возможно, несколько низкоприоритетных задач и уменьшить их жесткие сроки. Предположим, что поставленная задача, имеющая конечный период и буферизированная выполняются несколькими способами - например, первый пришел первый обработан - то необходимые максимальное время задержки и объема буфера могут быть вычислены, как предложено в этой статье.

Лучшим решением является назначение приоритетов задач динамически. Остальные разделы этой статьи посвящен одному

определенному методом динамического задания приоритетов. Этот метод является оптимальным в том смысле, что если множеству задач можно назначить некоторые приоритеты(условия описаны в предыдущих главах), оно также может быть запланировано по этому метод. Другими словами, верхняя граница коэффициента загрузки процессора равна 100 процентов.

7. ДИНАМИЧЕСКИЙ АЛГОРИТМ ПЛАНИРОВКИ.

Перейдем теперь к изучению динамических алгоритмов планирования, которые мы называем дедлайн [4] управляющие алгоритмы планирования. Используя этот алгоритм, приоритеты выдаются задачам в зависимости от дедлайнов их текущих запросов. Задаче будет назначен высокий приоритет, если дедлайн ее текущего запроса является ближайшим, и будет присвоен низкий приоритет, если дедлайн ее текущего запроса наступит не скоро. В любой момент, задача с наивысшим приоритетом имеющая невыполненные запросы будет выполнена. Такой метод назначения приоритетов для задач называется динамическим, в отличие от статического назначения, в котором приоритеты задачи не меняются во времени. Сейчас мы попытаемся создать необходимое и достаточное условие работы дедлайн управляющего алгоритма планирования.

[4] Дедлайн – конечный момент (дата и\или время) к которому должна быть выполнена задача.