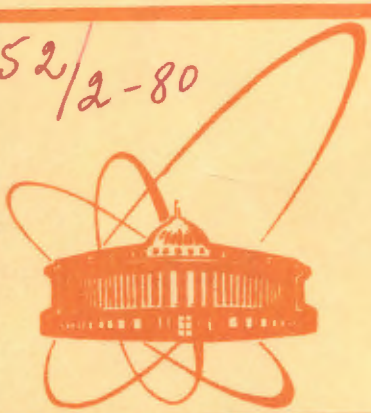


объединенный
институт
ядерных
исследований
дубна

Р11 - 12748

И.Н.Силин, Е.Д.Федюнькин

УНИВЕРСАЛЬНЫЙ АЛГОРИТМ РАЗДЕЛЕНИЯ ВРЕМЕНИ

1979

Силин И.Н., Федюнькин Е.Д.

Р11 - 12748

Универсальный алгоритм разделения времени

Предложен алгоритм разделения времени, пригодный для широкого класса вычислительных систем /в том числе многопроцессорных/. Алгоритм обеспечивает преимущества диалоговым задачам, требующим мало памяти и времени ЭВМ, с сохранением разделения времени между всеми задачами согласно внешним приоритетам при хорошей загрузке процессора /процессоров/. Алгоритм содержит в себе оптимальную процедуру смены задач в оперативной памяти.

Работа выполнена в Лаборатории вычислительной техники и автоматизации ОИЯИ.

Препринт Объединенного института ядерных исследований. Дубна 1979

Silin I.N., Fedyunkin E.D.

P11 - 12748

Timesharing Universal Algorithm

Timesharing system low level scheduling algoruthm is proposed for the wide class of one and multi-processor computer configurations. Dynamical priority is the piece constant function of the channel characteristic and system time quantum. The interactive job quantum has variable length. Characteristic recurrent formula is received. The concept of the background job is introduced. Background job loads processor if high priority jobs are inactive. Background quality function is given on the base of the statistical data received in the timesharing process. Algorithm includes optimal trashing off procedure for the jobs replacements in the memory. Sharing of the system time in proportion to the external priorities is guaranted for the all active enough computing channels (background too). The fast answer is guaranteed for the interactive jobs, which use small time and memory.The external priority control is saved for the high level scheduler. The experience of the algorithm realisation on the BESM-6 computer in JINR is discussed.

Preprint of the Joint Institute for Nuclear Research. Dubna 1979

1. ВВЕДЕНИЕ

Современная мощная вычислительная система рассчитана на массовое обслуживание заявок от потребителей вычислительной мощности. С точки зрения индивидуального пользователя система тем лучше, чем быстрее ее реакция. Ограниченность вычислительной мощности ЭВМ приводит к необходимости классифицировать заявки по приоритету и прерывать обслуживание менее срочных заявок для обслуживания более срочных.

В системах разделения времени организуется циркуляция задач между оперативной памятью и более дешевой памятью высшего уровня - большего объема, но меньшего быстродействия. Может быть несколько последовательных уровней памяти, причем циркуляция информации между нижними уровнями на некоторых машинах осуществляется полностью аппаратно. Планирование перемещений информации с уровня на уровень наталкивается на целый ряд внутренних противоречий, без преодоления которых вычислительная система захлебывается при обслуживании сравнительно малого числа задач. Поиск корректного решения проблемы планирования требует системного, можно сказать "экологического", подхода. Опыт развития и эксплуатации ОС "Дубна" /1-4/ на ЭВМ БЭСМ-6 стимулировал разработку нового алгоритма разделения времени, в главных своих чертах применимого к широкому классу вычислительных систем.

Важность создания оптимального алгоритма разделения времени показывает следующий пример. В ОИЯИ длительное время эксплуатируется двухпроцессорная ЭВМ CDC-6500. Оценки показывают, что замена реализованного в ней алгоритма разделения времени на оптимальный была бы, по сути дела, эквивалентна добавлению в систему еще одного процессора при соответствующем увеличении оперативной памяти.

Ниже мы сформулируем требования к оптимальному алгоритму разделения времени. Подобные требования неоднократно выдвигались /4,5/ в разных комбинациях. Мы хотим найти принципы, допускающие единый подход к указанным требованиям.

В отличие от распространенной практики /6/ мы не исходим из статистической гипотезы о распределении заявок. Попытаемся создать алгоритм,который бы разумно функционировал в любых условиях.

2. ОСНОВНЫЕ ТРЕБОВАНИЯ К ОПТИМАЛЬНОМУ АЛГОРИТМУ РАЗДЕЛЕНИЯ ВРЕМЕНИ

1. Время между задачами должно делиться пропорционально их внешним приоритетам. Внешний приоритет задается оператором или планировщиком высокого уровня. Если задача за некоторый стандартный период не смогла полностью использовать выделенную ей долю времени, то неиспользованное время распределяется между другими задачами.

2. Диалоговые задачи должны обслуживаться при первой возможности. Время ответа должно быть пропорционально использованному на акт диалога времени. Короткие недиалоговые задачи, запущенные с терминала, должны обслуживаться так же, как диалоговые. Запуск задачи в последнем случае нужно рассматривать как акт диалога. Отметим, что диалоговые задачи потребляют в среднем меньше времени, чем их законная согласно внешнему приоритету доля.

3. Должен быть предусмотрен абсолютно высокий приоритет для задач, исполняющих важные системные функции или обеспечивающих сверхбыстрые реакции в режиме ON-LINE. Такие задачи должны, как правило, требовать мало времени и памяти и не участвуют в разделении времени.

4. Должна быть обеспечена максимальная загрузка центрального процессора /или процессоров/. Последнее порождает определенные требования к характеру использования менее быстродействующих компонентов ЭВМ. Заказы к этим компонентам должны распределяться по возможности равномерно во времени, чтобы избежать возникновения очередей.

5. Алгоритм разделения времени не должен требовать чрезмерно больших затрат машинного времени и памяти для своего функционирования. Впрочем, чем мощнее вычислительная система, тем более сложный и изощренный алгоритм может оказаться оправданным.

Эти требования взаимно противоречивы: алгоритм, абсолютно оптимальный с точки зрения какого-то одного пункта, абсолютно неприемлем с точки зрения других пунктов. Например, максимальная загрузка процессора может быть достигнута, если чисто счетные задачи захватывают процессор на все время их активности, что противоречит пунктам 1 и 2. Требование быстрой диалоговой реакции вступает в конфликт с пунктами 1 и 4. Нужен разумный компромисс. Поиск оптимального алгоритма эквивалентен задаче о минимизации функции многих переменных. Компромисс должен определить, какой именно вид имеет указанная

функция. Последняя задача обладает большей сложностью, чем сама задача оптимизации алгоритма, поскольку требует неформального эвристического подхода.

Укажем очевидные следствия из требований 1-5. /Пункты 2.1-2.3 касаются в основном диалоговых задач, а пункты 4.1-4.4 связаны с эффективностью использования процессора/.

2.1. Минимальное время ответа для задач, удаленных из оперативной памяти, определяется временем, необходимым для вкачивания задачи /или ее активных страниц/. Но в соответствии с пунктом 4 диалоговые задачи должны регулярно удаляться из оперативной памяти, чтобы освободить ее для счетных задач. Следовательно, целесообразно давать преимущества задачам, использующим в текущий момент малую память.

2.2. Если диалоговая задача начинает потреблять слишком много счетного времени, она должна временно терять преимущества по отношению к другим задачам, чтобы не нарушать требования 1.

2.3. Аппарат смены задач в памяти должен обеспечить как можно более быстрое вкачивание динамически приоритетной и активной задачи /или ее активных страниц/, не допуская значительных помех со стороны других задач.

4.1. Так как при разделении времени может быть накоплена информация о поведении задач, желательно вместе с динамически приоритетными в данный момент задачами держать в оперативной памяти задачу, хорошо использующую процессор и занимающую мало памяти. Такую задачу будем называть фоновой. Ее функция состоит в том, чтобы захватить процессор, когда более приоритетные задачи не могут этого сделать /например, по причине ввода-вывода или смены задач в памяти/. Фоновые задачи, тем не менее, должны участвовать в общем разделении времени.

4.2. Задачи, активно использующие процессор, должны достаточно редко сменять друг друга в памяти, чтобы сократить расходы на организацию обменов и возможные простои процессора.

4.3. Должен быть предусмотрен абсолютно низкий приоритет для резервных задач. Резервная задача обычно отсутствует в оперативной памяти и должна загружать машину, если другие задачи не способны это сделать. Резервная задача не участвует в разделении времени.

4.4. Задачи, требующие мало процессорного времени и мало оперативной памяти, должны в соответствии с пунктами 1,4

и для полноценного использования внешних устройств чаще, чем другие задачи, получать высокий динамический приоритет.

Не ограничивая общности алгоритма, будем рассматривать в дальнейшем машины с одним процессором и страничной организацией памяти. Особенности, возникающие при использовании алгоритма на ЭВМ других типов, будут специально оговорены.

3. БАЗОВЫЙ АЛГОРИТМ РАЗДЕЛЕНИЯ ВРЕМЕНИ

Будем учитывать время t_i, использованное задачами. Естественный процесс разделения времени состоит в том, чтобы каждый раз выбирать в решение задачу с минимальным t_i/P_i. Здесь P_i - внешний приоритет, i - номер задачи, $P_i \geq 1$. Добившись, чтобы t_i/P_i не зависело от i, мы получим $t_i/t_j = P_i/P_j$ в соответствии с пунктом 1.

Нельзя допускать слишком частых насильственных переключений с задачи на задачу. Отсюда возникает понятие кванта T_i. Скажем сразу, что в соответствии с требованием 4.2 время кванта должно быть не меньше, чем астрономическое время, необходимое для вкачивания в оперативную память активных страниц задачи.

Процесс разделения времени связан с взаимным учетом долгов по счетному времени между задачами. Однако учитывать долги за бесконечное время плохо с точки зрения диалоговых задач, так как задача, которой много задолжали, активизировавшись, может на длительное время захватить процессор и блокировать работу других задач. Возникает понятие еще одного кванта времени - учетного кванта NP_i, где N - величина, общая для всех задач.

Мы увидим далее, что требования, предъявляемые диалоговыми задачами, вполне удовлетворяются, если для каждой задачи ввести квант T_i переменной длительности, зависящий от характера использования задачей времени и памяти. При таком подходе почти исчезает различие между диалоговыми и счетными задачами.

Введем характеристики n_i^j, определяемые по следующему правилу:

$$0 \leq n_i^j < N. \qquad /3.1/$$

В частности, при запуске диспетчера все $n_i^0 = 0$. К каждому n_i^j регулярно прибавляется t_i^j/P_i, где t_i^j - время, потребленное задачей i за элементарный квант с номером j /между двумя временными прерываниями/. Если $n_i^j \geq N$, то из всех n_i^j вычитается N/K, где K - заданное целое число, связанное с величиной максимального кванта, существующего в системе:

$$T_i \leq P_i N/K. \qquad /3.2/$$

Если после вычитания N/K некоторые n_i^j оказываются отрицательными, то они заменяются на $\{n_i^j/r_i\}r_i$. Здесь $r_i = T_i/P_i$, фигурными скобками обозначена дробная часть выражения. Введем обозначения:

$$\delta_i^j = t_i^j/P_i + n_i^j - \frac{N}{K}\max_{\ell}[(t_\ell^j/P_\ell + n_\ell^j)/N], \qquad /3.3/$$

квадратными скобками обозначена целая часть выражения. Имеет место рекуррентная формула:

$$n_i^{j+1} = \delta_i^j - [\min(0, \delta_i^j/r_i)]r_i. \qquad /3.4/$$

Динамическим приоритетом задачи будем называть величину p_i, определяемую ступенчатой функцией

$$p_i = N/r_i - [n_i/r_i]. \qquad /3.5/$$

Численное значение динамического приоритета p_i может измениться лишь в трех случаях: при исчерпании задачей кванта T_i, при его изменении и после обработки ситуации $n_i^j \geq N$. При каждом изменении p_i производится проверка: не следует ли перейти к решению /или инициировать вкачивание/ новой активной задачи с максимальным p_i. При равных p_i выбирается задача с минимальным i. Процедура выбора задачи запускается также при активации или дезактивации любой задачи, в том числе системной задачи, осуществляющей вкачивание.

Алгоритм обеспечивает последовательное подтягивание n_i к верхней границе N. При этом предпочтение отдается задачам с меньшим нормированным квантом r_i. Этим можно пользоваться для удовлетворения требования 2. Гарантируется правильное разделение времени между задачами, у которых n_i перманентно находится вблизи верхней границы N. Действительно, время, использованное такими задачами с момента t_0 до текущего момента t_1, есть

$$\Delta t_i = P_i(n_i^{j_1} - n_i^{j_0} + IN/K), \qquad /3.6/$$

где I - число вычитаний N/K за истекшее время. Пренебрегая членом $P_i(n_i^{j_1} - n_i^{j_0})$, получим $\Delta t_i \sim P_i$, т.е. разделение времени выполняется с точностью до учетного кванта.

Анализ формул /3.3/ и /3.4/ показывает, что для сохранения различий в величинах n_i между задачами, полностью использующими свое время, и задачами, не выбирающими выделенной им доли времени, K должно быть не меньше чем 3. Чисто технически удобно выбрать $K = 4$.

Процедура разделения времени должна иметь иммунитет от нарушений, возникающих из-за того, что задачи, окончившие счет, покидают систему и в решение принимаются новые задачи. Это особенно важно, когда в системе много коротких задач. Иммунитет достигается, если производить разделение времени не между задачами, а между счетными каналами. Практически это означает, что в момент, когда задача кончается и покидает счетный канал или этот канал захватывает новая задача, не следует модифицировать величину n_i. Новую задачу резонно принимать в решение на свободный канал с минимальным n_i.

Тем не менее желательно предоставить коротким задачам возможность быстро кончиться, а запускаемым в счет диалоговым задачам - быстро пройти процесс инициации и войти в диалог, т.е. хотелось бы занулить n_i в момент запуска. Заведем стек перерасходованного времени R. В момент запуска задачи на канал i заполняем стек по правилу $R = R + P_i n_i$, после чего зануляем n_i. Здесь P_i - внешний приоритет задачи, ранее окончившейся на канале i. При обработке ситуации $n_i \geq N$, в тех случаях, когда подвижка $\Delta n_i < N/K$ из-за малости n_i, опустошаем стек: $R = R - P_i (N/K - \Delta n_i)$, т.е. мы учитываем, что допущенный ранее перебор времени компенсируется недобором времени некоторыми задачами. Если $R + P_i n_i \geq R_{max}$, перенос n_i в R при запуске задачи не делается - система перегружена короткими задачами и мы ей ничем не можем помочь. Величина R_{max} определяется характером использования машины, но должна быть не меньше чем $N \sum_i P_i$ по всем каналам при типичных наборах P_i.

4. СИСТЕМНОЕ ВРЕМЯ

До сих пор мы не конкретизировали, какое именно время имеем в виду. Если в качестве использованного времени брать чисто процессорное, могут быть неприятности. А именно, если у нас появится некоторое количество чисто обменных задач, совместно занимающих всю физическую оперативную память, они долго будут иметь высокий динамический приоритет и не допускать в память счетные задачи. Процессор будет простаивать. Поэтому целесообразно в указанных выше формулах вместо

счетного времени использовать "системное время" t_{si}, а именно:

$$t_{si} = t_{ci} + \frac{\alpha}{M} \int \mu_i (t_{ni}) dt_{ni}, \qquad /4.1/$$

где t_{ci} - процессорное время задачи i, μ_i - память, занимаемая этой задачей в текущий момент времени, t_{ni} - ее время неактивности, M - вся доступная для задач оперативная память.

Коэффициент α при интеграле неактивности должен удовлетворять условию $\alpha \leq 1$, чтобы системное время неактивной задачи не шло быстрее, чем системное время задачи, захватившей процессор. Однако в случае, когда неактивная задача занимает всю доступную память и, следовательно, $t_{si} = \alpha \int dt_{ni}$, целесообразно приписывать ей время не меньшее, чем потребляемое чисто счетной задачей. Отсюда $\alpha = 1$, и поэтому

$$t_{si} = t_{ci} + \frac{1}{M} \int \mu_i (t_{ni}) dt_{ni}. \qquad /4.2/$$

Если есть такая возможность, в μ_i формулы /4.2/ можно учитывать только память, реально использовавшуюся на протяжении предыдущего и текущего квантов времени.

5. УПРАВЛЕНИЕ ВЕЛИЧИНОЙ КВАНТА

Диалоговая задача может оказаться в хвосте очереди активизировавшихся счетных задач, у которых n_i вблизи нуля. К тому же если задача, прежде чем войти в диалог, активно использовала процессор, мы можем очень долго ждать затухания ее n_i, если N велико или велико значение $\sum_i P_i$. Для повышения приоритета задач, активизировавшихся после диалога, можно устанавливать для них минимальный квант, не противоречащий условию

$$T_i \geq m_i \Theta. \qquad /5.1/$$

Здесь Θ - среднее время замещения одной страницы, m_i - число активных страниц задачи /берется из статистики за предыдущий и текущий кванты/. Неравенство /5.1/ означает, что квант T_i не меньше, чем время вкачивания активных страниц задачи. Так как задачи с большим внешним приоритетом P_i уже находятся в привилегированном положении, видимо, не следует давать им дополнительных преимуществ. Учитывая формулы /3.5/ и /5.1/,

будем устанавливать после выхода из диалога

$$r_i \equiv \frac{T_i}{P_i} = m_i\Theta . \qquad /5.2/$$

Чтобы удовлетворить пункту 4.2, если задача не входит в диалог за время кванта T_i, квант увеличивается пропорционально времени, набранному задачей с момента активации, но до значения не больше максимально допустимого P_iN/K. Для любой задачи можно на ходу увеличить квант, например вдвое, если до исчерпания очередного кванта окажется, что она использует много страниц, так что $m_i\Theta > r_i$. Маленькие и частые изменения кванта нецелесообразны.

Если задача часто дезактивируется по диалогу, но использует значительное в среднем счетное время, недостаточное, однако, для сильного увеличения ее кванта, то она будет иметь высокий приоритет /3.5/, пока ее характеристика n_i не приблизится к правой границе N. Оставшееся до границы расстояние задача может исчерпать во время случайных коротких дезактиваций приоритетных задач или при смене задач в памяти. Такая задача, следовательно, легко провоцирует ситуацию $n_i \geq N$ с последующим вычитанием N/K у всех характеристик. После этого она снова окажется далеко от N в масштабе своих малых квантов /5.2/ и, получив высокий приоритет, снова приблизится к N и т.д. Иначе говоря, задача будет систематически перерасходовать время. Это неприятно еще потому, что она незаконно задвинет характеристики других задач в область нуля и потеряется информация об использованном времени, что приведет к нарушению разделения времени и между ними. Чтобы ликвидировать такую опасность, будем любой задаче, характеристика которой попадает в критический интервал $N-N/K \leq n_i < N$, увеличивать квант до максимального. Этот прием способствует также правильному разделению времени между фоновыми задачами /раздел 7/.

Предложенный здесь алгоритм управления величиной кванта в сочетании с /3.5/ обеспечивает диалоговой задаче даже при относительно большом n_i преимущество на небольшое время перед задачами с малыми n_i, но большими квантами - недиалоговыми или диалоговыми, потерявшими свои преимущества из-за активного использования процессора либо большой памяти. Диалоговой задаче предлагается кредит, за который она должна, если желает иметь хорошую реакцию, расплачиваться малым потреблением времени между актами диалога. В противном случае в какой-то момент у задачи с большим квантом динамический приоритет окажется выше и она захватит процессор на время большого кванта /либо расстояние ее n_i от N в числе

больших квантов будет больше, чем у диалоговой в числе малых квантов, либо характеристика диалоговой задачи попадет в критический интервал/.

Для задачи, впервые принятой в решение на счетный канал, полезно установить

$$T_i = P_i\, m_0\Theta , \qquad /5.3/$$

где m_0 - минимальное число страниц, которое вообще может иметь задача. Тем самым мы даем возможность очень короткой задаче быстро окончиться.

Иногда полезно устанавливать малый квант /и, следовательно, увеличивать динамический приоритет/ задачам, монопольно захватывающим общие ресурсы, что может вызвать дезактивацию других задач. В некоторых случаях таким задачам следует давать и абсолютный приоритет, когда захватывается популярный ресурс и особенно в том случае, если это делает резервная задача, не участвующая в разделении времени. Естественно, что операционная система не должна допускать захвата важных ресурсов на длительное время.

6. НЕМНОГО О СТАТИСТИКЕ

В предыдущем разделе мы использовали оценку числа активных страниц задачи. В дальнейшем нам потребуются и другие показатели. Естественным интервалом времени для накопления статистики является квант. Однако /см. раздел 5/ квант может быть уменьшен до своего исчерпания. В этом случае мы должны добрать время до цѐлого числа уменьшенных квантов и учесть статистику за время с начала старого кванта.

Так как мы делим системное, а не процессорное время, может иногда получиться, что квант полностью исчерпан вкладом от интеграла неактивности /задача за все время кванта ни разу не активизировалась/. Статистика по такому кванту, как правило, бессмысленна, и нужно использовать статистику по предыдущим квантам.

Для улучшения "взаимопонимания" между пользователем и операционной системой /для развития интеллекта пользователя, как теперь говорят/ необходимо выдавать ему полную информацию о характере потребления задачей вычислительных ресурсов.

7. ВЫБОР ФОНОВОЙ ЗАДАЧИ

В качестве фоновой задачи /см. пункт 4.1/ следует выбирать задачу, требующую меньше памяти и по возможности лучше

использующую процессор. Кроме того, желательно, чтобы ее страницы уже были в оперативной памяти /оптимальный вариант - когда в качестве новой фоновой задачи выбирается подходящая задача, только что исчерпавшая свой квант в качестве приоритетной/. Нужно принять меры, чтобы она участвовала в общем разделении времени и выступала в качестве фоновой за счет своей законной доли системного времени. Для оценки качества задачи в роли фоновой нужно построить функцию пяти переменных. Этими переменными являются: k - коэффициент загрузки процессора,

$$k = \frac{t_c}{t_c + t_n}, \qquad /7.1/$$

m - число активных страниц, μ - число страниц, уже находящихся в оперативной памяти, n - характеристика, P - внешний приоритет.

Попытаемся оценить влияние этих параметров на эффективность использования памяти и процессора. Если задача уже находится в оперативной памяти, то средний коэффициент использования процессора на одну страницу равен k/m. Чем эта величина больше, тем лучше задача использует память и, следовательно, лучше в качестве фоновой. Однако, будучи выбрана в качестве фоновой, она лишь долю k достающегося ей времени может владеть процессором, и поэтому мы выбираем для оценки качества фоновой задачи величину k^2/m.

Если не все активные страницы задачи находятся в оперативной памяти, то их еще нужно вкачать. Время вкачивания: $(m-\mu)\Theta$. Это ухудшает эффективность использования процессора фоновой задачей. Чтобы оценить ухудшение, оценим время, в течение которого задача может выступать в качестве фоновой. Так как мы стараемся не допустить перерасхода фоновой задачей законного времени, то возьмем в качестве оценки время, которое она может использовать, не превосходя по n величину $N-N/2K$, а именно: $P(N-N/2K-n)$. Оценим теперь приведенный коэффициент загрузки процессора k' с учетом простоя при замещении страниц. Астрономическое время работы фоновой задачи можно оценить как $P(N-N/2K-n)+(m-\mu)\Theta$, а время загрузки процессора - как $kP(N-N/2K-n)$, откуда:

$$k' = \frac{kP(N-N/2K-n)}{P(N-N/2K-n)+(m-\mu)\Theta}. \qquad /7.2/$$

Качество фоновой задачи F определяется формулой

$$F = (k')^2/m = \frac{1}{m}\left(\frac{kP(N-N/2K-n)}{P(N-N/2K-n)+(m-\mu)\Theta}\right). \qquad /7.3/$$

Фоновая задача выбирается только из числа задач, у которых $N-N/2K > n$.

Для дальнейшего следует сделать одно замечание о политике подкачки страниц в оперативную память: по возможности не вытесняются из памяти страницы, принадлежащие фоновой задаче /более подробно см. в разделе 8/, чтобы обеспечить фоновой задаче возможность немедленно захватить процессор, если никакая другая из находящихся в памяти задач не может этого сделать. Фоновая задача случайно может оказаться и самой приоритетной среди активных.

Оказывается резонным производить перевыбор по фоновой функции /7.3/ только в момент изменения системы динамических приоритетов или при переходе n_i у фоновой задачи через $N-N/2K$, следовательно, дезактивация фоновой задачи, не связанная с диалогом, не является основанием для выбора новой фоновой задачи.

В экспериментах на БЭСМ-6 наблюдалась перекачка страниц и, следовательно, времени от приоритетной задачи к фоновой в ситуации, когда приоритетная задача не помещалась в памяти вместе с фоновой и достаточно часто, но на короткие промежутки времени прерывалась еще более приоритетной задачей, например системной.

Последний эффект может создать помехи диалоговым задачам. В этой связи мы предлагаем особый режим замещений страниц, когда заявки на подкачку от фоновой задачи удовлетворяются за счет ее собственных страниц, которые к тому же лишаются своих привилегий. Этот режим включается, если фоновая задача затребовала слишком большую память /например, больше половины физической памяти/, и сохраняется до перевыбора фоновой задачи. Процесс разделения времени облегчается, если введены ограничения на предоставляемую одной задаче физическую память.

Итак, задачи выбираются в решение по двум каналам. По первому каналу проходят задачи, требующие большой памяти или плохо использующие процессор, по второму каналу проходят задачи, требующие малой памяти и хорошо использующие процессор. В среднем динамический приоритет задач, проходящих по первому каналу, выше чем динамический приоритет задач, проходящих по второму каналу. Задачи могут переходить из канала в канал, если меняется их качество. Задача вытесняется из второго канала в первый, если она мешает прохождению задач по первому каналу.

8. СМЕНА ЗАДАЧ В ПАМЯТИ

Мы изложим алгоритм смены задач, рассчитанный на ЭВМ со страничной организацией памяти. На машинах с аппаратной поддержкой перемещения программ в физической памяти, вообще говоря, можно применить весьма сходный алгоритм. При этом для получения удовлетворительной реакции в диалоговых программах весьма желательно предусматривать частичное вытеснение больших программ. Например, активизировалась диалоговая задача, требующая мало памяти, а память занята большими программами. Нужно освободить необходимое количество памяти, обслужить диалог, а затем восстановить большую программу, чтобы она доработала свой квант. Тем более это естественно для ЭВМ со страничной организацией памяти.

При организации замещений страниц следует учитывать как приоритет страницы, так и приоритет задач. Приоритеты страниц характеризуют частоту использования страниц в текущий период. На многих типах ЭВМ есть аппаратура, помогающая оценивать активность использования страниц. На БЭСМ-6 такой аппаратуры нет. В ОС "Дубна" на БЭСМ-6 для оценки приоритетов страниц используется аппаратура защиты страниц, предназначенная для прерывания программы при обращении к странице, отсутствующей в оперативной памяти или участвующей в обмене. Интенсивность использования страниц оценивается косвенно из вероятностных соображений. После каждого считывания страницы или выделения страницы /т.е. изменения в характере использования памяти/ защищаются все страницы всех задач. Далее мы обслуживаем прерывания, возникающие при обращении к страницам, изменяем приоритет страницы и снимаем с нее защиту, так что до следующего акта подкачки обращения к данной странице не будут вызывать прерываний. Приоритет страницы вычисляется из следующего соображения: чем чаще используется страница, тем более вероятно, что именно к ней будет первое обращение. Поэтому странице, вызвавшей первое прерывание по защите, присваивается абсолютно максимальный приоритет; странице, вызвавшей второе прерывание, присваивается на единицу меньший приоритет и т.д. У страниц, к которым не было обращений, сохраняется старый относительный приоритет. После следующего акта замещения страниц процедура в точности повтояется.

Таблица приоритетов страниц имеет списковую структуру с прямой и обратной ссылками; в строке, соответствующей данной физической странице, есть номера страниц с приоритетом, на единицу меньшим и на единицу большим. Есть три указателя страниц: наименее приоритетной, наиболее приоритетной и страницы, для которой последний раз вычислялся приоритет.

Для изменения приоритета страницы нужно изменить ссылки максимум в пяти строках таблицы приоритетов. Данный алгоритм был применен одним из авторов в 1969 году в загрузчике мониторной системы "Дубна" [7,8] при программной имитации большой виртуальной памяти и хорошо себя зарекомендовал. В дальнейшем при создании диспетчера DD71 [1] он был использован для замещения страниц в операционной системе. Алгоритм оказался настолько эффективным, что можно предложить именно этот алгоритм вычисления приоритетов страниц для аппаратной реализации.

Однако и программная реализация, по крайней мере на БЭСМ-6, оказалась необременительной. На один акт замещения страниц в среднем приходится обслужить 15 прерываний для оценки приоритетов. Время, затраченное на это, не идет ни в какое сравнение с затратами на организацию замещений страниц, которые, в свою очередь, не превышают 5-7% процессорного времени.

Разработанный для ОС "Дубна" алгоритм замещений страниц основывается на идеологии, аналогичной идеологии рабочих наборов [9], однако мы не фиксируем рабочий набор явным образом, а создаем предпосылки для того, чтобы в соответствии с п.2.3 главы 2 рабочий набор выбираемой в решение задачи автоматически считывался без помех со стороны других задач - во избежание "толкотни" в памяти и для обеспечения быстрой реакции диалоговым задачам. Прогнозирование рабочего набора задачи может приводить к грубым неточностям, и поэтому алгоритм смены задач в памяти должен быть мало чувствителен к таким просчетам. В нашем случае используется прогноз только величины рабочего набора, который сказывается только на величине кванта диалоговой задачи и на фоновой функции.

Нами введена особая дисциплина обслуживания заявок на замещение страниц, а параллельные каналы обмена, имеющиеся в машине, используются не для организации параллельного обслуживания заявок от многих задач, а для ускорения считывания страниц динамически приоритетной и фоновой задач. Однако на многопроцессорной ЭВМ нужно, естественно, обслуживать заявки от большего числа задач.

Для того, чтобы запросы на память обслуживались возможно быстрее, полезно иметь резерв свободной памяти, освобождаемой впрок, до реального ее запроса. В ОС "Дубна" введен такой резерв - две страницы / на машинах с памятью 32 К - одна страница/. Правда, эти страницы неравноправны. Одна из них может быть любой, а вторая всегда расположена в первых 32 К физической памяти, так как аппаратные особенности

БЭСМ-6 вынуждают считывать привилегированные программы и объекты, к которым обращаются по физическому, а не виртуальному адресу, в первые 32 К памяти.

Дисциплина обслуживания заявок следующая. Обслуживаются только заявки от динамически самой приоритетной активной задачи и фоновой задачи. Если есть заявки от других задач, они не обслуживаются до тех пор, пока эти задачи не станут либо динамически приоритетными, либо фоновыми.

Обслуживание заключается в том, что выдается заказ на считывание затребованной страницы на резервную /если страница требуется в первый раз, то страница просто отдается запросившей задаче/. Разыскивается претендент на удаление из памяти, если в этом есть необходимость/может быть, в памяти есть свободная страница, тогда она выбирается в качестве резервной/. Для этого просматривается таблица приоритетов страниц, начиная со строки, соответствующей странице с минимальным приоритетом, и ищется наименее приоритетная страница, которую можно откачать. Поскольку не отбирается страница у динамически приоритетной и фоновой задач, такой страницы может и не оказаться, зато выяснится, есть ли не занятые обменом и незафиксированные страницы у приоритетной и фоновой задач. Если такие страницы есть у фоновой задачи, то вытесняется ее наименее приоритетная страница. Если такие страницы есть только у приоритетной задачи, вытесняется ее страница, но лишь в том случае, если заявка не от фоновой задачи. Заявки от фоновой задачи не обслуживаются также и в том случае, если страниц, которые можно предоставить фоновой задаче, слишком мало - значительно меньше числа ее активных страниц или меньше минимально необходимого числа страниц. Фоновой задаче затребованная страница предоставляется только в том случае, если найден реальный претендент на откачивание. После обслуживания очередной заявки от приоритетной задачи снова может быть сделана попытка обслужить заявку от неудовлетворенной фоновой задачи. Если же после выдачи страницы приоритетной задаче не нашлось страницы, которую можно вытеснить, то задача подкачки ждет появления такой страницы и до ее появления другие заявки на обслуживание не принимаются.

По указанной в разделе 7 причине нужен еще один режим обслуживания заявок от фоновой задачи, когда они удовлетворяются за счет ее же страниц, а при выборе претендента на откачивание после заявки от приоритетной задачи принадлежность страницы фоновой задаче не учитывается.

Процедура выбора претендента на откачивание может быть дополнительно оптимизирована, если в операционной системе

ведется оперативный учет числа страниц, пригодных к вытеснению - суммарно и в каждой из задач.

Если аппаратура обмена с памятью высшего уровня /например, диском/ допускает обмен группой страниц за время, характерное для одного акта обмена, выгодно вытеснять сразу пул наименее приоритетных страниц одной задачи, а когда одна из них снова потребуется,- вкачивать их также группой,повышая приоритет всей группе страниц, чтобы они не вытеснялись при следующем замещении страниц.

Ранее /разделы 5,6/ мы не включали в интеграл неактивности время подкачки страниц задачи. В случае машин с большой виртуальной памятью это правило следует уточнить и включать время вкачивания в интеграл неактивности, если число страниц, захваченных задачей, достигает некоторого предела /например, захватывается вся доступная память/. Таким образом, мы штрафуем задачи, бестолково сканирующие гигантскую виртуальную память и мешающие тем самым другим задачам. Если такие задачи не штрафовать, они могут бесконечно долго инициировать подкачку, сохраняя высокий приоритет, не используя процессор и занимая память.

После выделения задаче максимальной памяти дополнительные запросы на память удовлетворяются за счет вытеснения ее же страниц с начислением системного времени. Можно позволить пользователю менять этот максимум в разрешенных пределах.

Несколько слов о замещении задач на машинах, не имеющих страничной организации памяти. На таких ЭВМ память высшего уровня выделяется блоками, которые можно рассматривать как псевдостраницы. Заявку на вкачивание задачи можно рассматривать как последовательность заявок на считывание страниц.

Если заведомо известно, что приоритетная задача дезактивировалась на короткое время /например, для обмена с барабаном или диском/, можно не отбирать у нее страниц. Именно так и следует действовать на ЭВМ без страничной организации памяти. На ЭВМ со страничной памятью при хорошей оценке приоритетов страниц страницы такой задачи и без того будут отбираться в последнюю очередь. С другой стороны, если задача стала плохо использовать процессор, может быть полезным перераспределить память между задачами в соответствии с их современным состоянием, что и происходит в изложенном ранее основном варианте алгоритма.

9. ОБЩИЕ СТРАНИЦЫ В ПАМЯТИ

В предыдущих разделах мы совершенно не учитывали возможность того, что у разных задач могут быть общие страницы.

На БЭСМ-6, например, такими страницами являются нерезидентные страницы диспетчера. На ЭВМ серии CYBER-70, 170 – это MULTIUSER JOBS. На некоторых современных ЭВМ, ориентированных на реентерабельность, все программы могут использоваться несколькими задачами. Возникает желание по возможности синхронизировать загрузку в память задач, у которых много общих страниц. Мы предлагаем при выборе задачи в счет использовать в качестве модифицированного динамического приоритета $\tilde{p}_i$ величину

$$\tilde{p}_i = p_i + \mu_i(t)\Theta/\tau_i . \qquad /9.1/$$

Здесь, как и раньше, μ_i - число страниц задачи, находящихся в памяти и реально использовавшихся данной задачей на протяжении предыдущего и текущего квантов. $\mu_i(t)\Theta$ - время, необходимое для вкачивания этих страниц, если мы позволим их откачать. В связи с /5.2/ $\tilde{p}_i - p_i \leq 1$. Таким образом, при прочих равных условиях в качестве приоритетной будет выбираться задача, у которой больше страниц в памяти. Этот метод несколько сокращает число подкачек даже в том случае, если у задач нет общих страниц.

В интеграле неактивности /4.2/ в μ_i, вообще говоря, не следует учитывать общие страницы, использованные другими задачами после задачи i.

10. МНОГОПРОЦЕССОРНЫЕ СИСТЕМЫ

Возможны два подхода к проблеме разделения времени в многопроцессорных системах, если одной задаче разрешается захватывать несколько процессоров.

1/ Можно считать каждую ветвь задачи отдельным заданием и делить время между заданиями. Преимущества такого подхода - в простоте реализации, а также в том, что задача может легко устанавливать относительные приоритеты своих ветвей. Однако если степень ветвления высока, то процессы короткие и практически теряет смысл статистика по каждому из них. Кроме того, при слишком большом числе процессов возникает перегрузка планировщика.

2/ Планировщик делит время только между задачами, но каждая задача может захватить много процессоров. В этом случае наиболее приоритетной задаче нужно отдать столько процессоров, сколько она может использовать, остальные - предложить следующей по приоритету задаче и т.д. Такой способ гарантирует оптимальное использование памяти, быстрые реакции

диалоговых задач и сводит к минимуму возможные простои процессоров. Однако затруднено разделение времени между процессами внутри задачи, поскольку задача должна реализовывать это разделение сама, вплоть до деталей, когда это существенно. Например, когда в задаче есть параллельные счетные и диалоговые ветви, она сама должна обеспечивать преимущества диалоговым ветвям и разделение времени между ними.

Можно объединить достоинства обоих подходов, разрешив задаче захватывать несколько счетных каналов, время между которыми делит планировщик. Каждый канал, однако, может использовать много процессоров. Естественно помещать в один канал ветви, использующие в основном общую память, относительный приоритет которых несуществен /например, ветви, обслуживающие один терминал/. Многоканальный подход имеет смысл и для однопроцессорных систем, поскольку облегчает программирование задач со сложной логикой и позволяет уменьшить зависимость стиля программирования от процессорности комплекса. Естественное развитие вычислительных систем приводит примерно к такой идеологии.

Изложенный ранее алгоритм разделения времени применим в общих чертах и к многопроцессорной системе, но требует некоторых уточнений.

При замещениях страниц хотя и следует разрешать подкачку многим приоритетным и фоновым задачам, необходимо придерживаться следующей дисциплины:

A. Приоритетными считаются столько задач /каналов/ с максимальным динамическим приоритетом, сколько достаточно, чтобы захватить все процессоры, если будет выделена соответствующая память.

B. В качестве фоновых выбираются задачи с максимальной фоновой функцией и не в большем количестве, чем необходимо для захвата всех процессоров. Их может быть и меньше, так как приоритетные задачи не обязательно дезактивируются одновременно.

C. Как и в однопроцессорном варианте, страницы по возможности не отбираются у приоритетных и фоновых задач. Если это невозможно, пытаемся отобрать страницы у фоновых задач, начиная с той, у которой худшее качество. Если и это невозможно, то отбираем страницы у самой низкоприоритетной из приоритетных задач. Фоновая задача не имеет права отбирать страницы у приоритетных задач и у фоновых задач лучшего качества. Приоритетная задача не имеет права отбирать страницы у задачи с более высоким приоритетом.

Эта дисциплина позволяет при необходимости посчитать задачу, требующую почти всей физической памяти хотя бы за счет простоя части процессоров.

В связи с многопроцессорностью счетных каналов требуется уточнение некоторых формул. Системное время мы будем записывать в виде

$$t_{si} = t_{ci} + \frac{1}{M}\int \max(0, \mu_i(t) - \nu_i(t)\frac{M}{S})(S - \nu_i(t))\,dt. \qquad /10.1/$$

Здесь S - число процессоров, ν_i - число активных процессов в счетном канале, t_{ci} - суммарное время канала по всем процессорам. Процесс, ждущий замещения страниц, считается активным, если канал не захватил больше памяти, чем это разрешено /например, всю физическую память, а лучше - $\min(\nu_i M/S, M)$/.

Системная добавка к процессорному времени в /10.1/ отлична от нуля в моменты, когда задача занимает больше чем $\min(\nu_i M/S, M)$ памяти. Если $\nu_i \geq S$, то добавка всегда равна нулю за счет первого сомножителя под интегралом.

Нужный для вычисления фоновой функции коэффициент загрузки процессоров k_i следует считать по формуле

$$k_i = t_{ci}/(t_{ci} + \int \max(0, S - \nu_i(t))\,dt. \qquad /10.2/$$

В случае одного процессора формула /10.1/ переходит в формулу /4.2/, а /10.2/ - в формулу /7.1/ /см. разделы 4 и 7/.

Формулу /7.3/ также желательно привести к виду

$$F = \frac{k_i'}{m_i}\min(Sk_i', 1). \qquad /10.3/$$

Здесь k_i определяется формулой /7.2/, k_i'/m_i - эффективность использования памяти, $\min(Sk_i', 1)$ - максимальная загрузка задачей одного процессора. Замена $k_i' \to \min(Sk_i', 1)$ /сравни с формулой /7.3// обеспечивает удовлетворительное качество фоновой задачи и в том случае, когда ей не позволяют использовать более одного процессора. Кроме того, при одинаковой интенсивности использования памяти многопроцессорные задачи ($k_i' > 1/S$) имеют одинаковые фоновые функции независимо от числа их активных процессов.

При подборе фоновых задач имеет смысл выбрать еще одну, если по уже выбранным фоновым задачам $\sum_i k_i \leq 1 - 1/S$. По-видимому, число фоновых задач не должно превышать числа процессоров.

Если процессоры имеют разное быстродействие /таковы, например, некоторые конфигурации ЭВМ фирмы CDC: CDC 6700, CYBER 74-2 /и дезактивировалась задача, занимавшая быстрый процессор, следует немедленно перехватить задачу у медленного процессора и передать ее быстрому, а затем действовать

обычным порядком /10/. В приведенные выше формулы следует ввести коэффициенты, зависящие от скорости процессоров.

11. СИСТЕМЫ С ПРОГРАММНО-УПРАВЛЯЕМОЙ МНОГОСТУПЕНЧАТОЙ ПАМЯТЬЮ

Такого рода системы могут иметь уровни типа: 1/ центральная оперативная память, 2/ расширенная (и сравнительно медленная) оперативная память, 3/ магнитные диски, или: 1/ центральная память, 2/ быстрые магнитные барабаны, 3/ диски.

Объекты, которые в нужный момент желательно иметь в промежуточной памяти, можно разбить на три класса.

1-й класс - это объекты, к которым программы обращаются явным образом /явное обращение к расширенной оперативной памяти/. Для этих объектов можно устраивать точно такую же систему замещений страниц, как между центральной оперативной памятью и промежуточной памятью. Алгоритм вычисления приоритетов страниц или блоков памяти промежуточного уровня может быть абсолютно аналогичным алгоритму, представленному в главе 8, хотя прерывания по защите страниц должны имитироваться программно /если нет их аппаратной реализации/.

2-й класс объектов - это короткие рабочие наборы диалоговых задач и, может быть, рабочие наборы наилучших кандидатов в фоновые задачи. Их нужно по возможности все время держать в промежуточной памяти.

3-й класс - рабочие наборы наиболее приоритетных из оставшихся задач. В последнем случае нужно заблаговременно перемещать в промежуточную память рабочие наборы активных задач, приоритет которых близок к максимальному, и удалять в медленную память рабочие наборы наименее приоритетных задач и страницы, выпавшие из рабочих наборов.

Если, однако, для нас существенно минимизировать количество перемещений магнитных головок на дисковых модулях, нужно распределять объекты третьего класса между промежуточной и медленной памятью, избегая обменов между указанными типами памяти. При этом может уменьшится эффективность использования процессоров.

Для эффективной работы с объектами класса 2 и 3 требуется оценивать рабочие наборы задач в явном виде в соответствии с /9/.

Как именно разделить промежуточную память между объектами разных классов, в сильной мере зависит от характера использования машины. Конкретное решение этого вопроса мы оставляем на усмотрение читателя. В последнее время наблюдается рост интереса к идеологии использования многоступенчатой памяти /см., например, /11/ /.

12. РАЗДЕЛЕНИЕ ВРЕМЕНИ АППАРАТУРЫ ВВОДА-ВЫВОДА

Каналы обмена с внешней памятью - магнитными барабанами, дисками и лентами - обладают ограниченной пропускной способностью, и при возникновении очередей их время нужно делить между задачами. Каждый канал может обслуживать много устройств ввода-вывода, к которым также могут образовываться очереди.

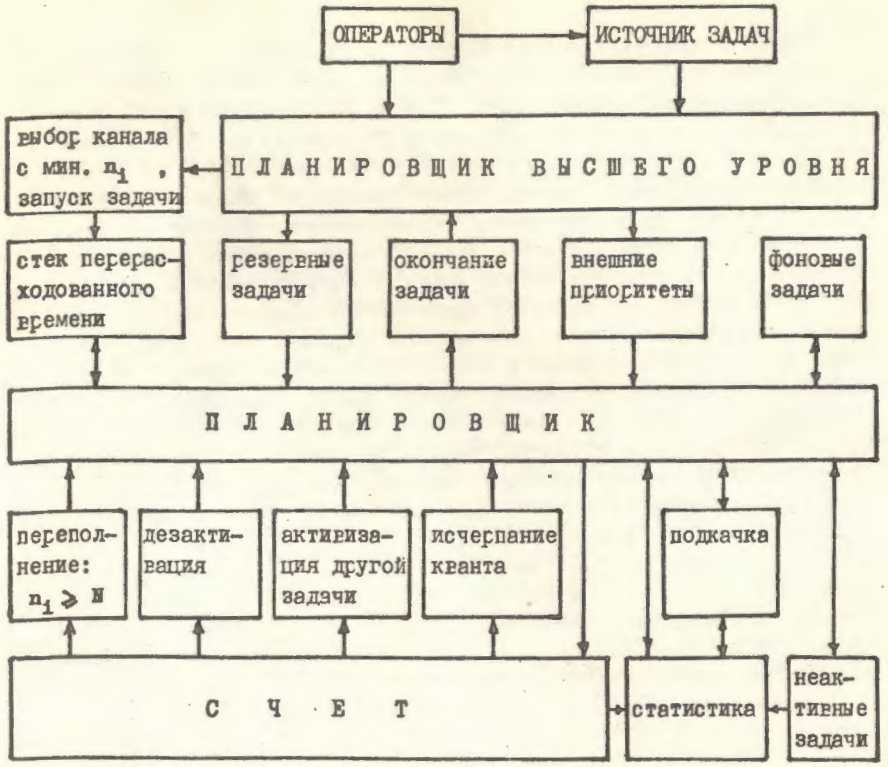
Теоретически каждую из таких очередей можно обслуживать с помощью алгоритма, аналогичного изложенному в разделе 3. Однако при этом мы с большой вероятностью нарушим требование 5 /см. раздел 2/. Обычно можно обойтись значительно более простыми алгоритмами.

В ОС "Дубна", например, к барабанам и дискам больших очередей не возникает. Это достигнуто, во-первых, благодаря дисциплине обслуживания заявок на замещение страниц, при которой в очередь на обмен поступают только заявки от приоритетной и фоновой задач, и, во-вторых, благодаря тому, что задаче, как правило, не разрешается накопить в очереди на обмен более одной-двух заявок. В результате при обслуживании очередей к устройствам прекрасно работает дисциплина FIFO. Тем не менее в драйвере, обеспечивающем работу магнитофонов серии ЕС, пришлось организовать циклическое разделение времени канала между магнитофонами [3], поскольку возможны серийные заявки /поиск нужного рекорда/.

13. ОПЫТ РЕАЛИЗАЦИИ АЛГОРИТМА РАЗДЕЛЕНИЯ ВРЕМЕНИ НА МАШИНЕ БЭСМ-6 В ОС "ДУБНА"

Один из фрагментов алгоритма /замена задач в памяти/ имеет длительную историю и представляет самостоятельный интерес. Опыт его реализации описан в [4] и частично - в разделе 8 настоящей работы. Первая версия ОС "Дубна" развивалась в ОИЯИ при отсутствии магнитных дисков и при ограниченной емкости магнитных барабанов. Основной целью развития ОС было повышение эффективности работы БЭСМ-6 в режиме пакетной обработки заданий. Мультипрограммность и замещения страниц использовались для улучшения загрузки процессора. Параллельно обрабатывалось не более трех задач пользователя и четырех служебных задач с виртуальной памятью.

В настоящее время ОС "Дубна" обеспечивает квазипараллельное исполнение 16 пользовательских задач и 8 служебных в режиме пакетно-диалоговой работы. Неизменным по сравнению со старой версией остался алгоритм вычисления приоритетов страниц. Алгоритм замещений страниц был несколько модифицирован



Структура системы разделения времени.

в связи с введением фоновых задач. Обеспечена возможность удаления из оперативной памяти почти всей информации, имеющей отношение к задаче /в том числе и первичного буфера вывода [12]/, при полном откачивании задачи.

Алгоритм разделения времени заменен на упрощенный вариант предложенного выше универсального алгоритма разделения времени, что позволило сохранить загрузку процессора на уровне 98-99% при затратах на диспетчерские нужды не более 10%. Универсальный алгоритм находится в стадии реализации.

Стандартное значение внешних приоритетов равно единице. Операторам разрешается по своему усмотрению назначить одному из счетных каналов приоритет 4 и еще одному - приоритет 2, чем они обычно пользуются для повышения приоритета пакетных каналов. Особо срочным задачам назначается приоритет независимо от воли операторов.

Нужно отметить, что на БЭСМ-6 разделение времени упрощается из-за ограниченности виртуальной памяти задачи. Одна задача не может использовать более 32 К виртуальной памяти, в то время как физическая память большинства БЭСМ-6 имеет объем 64 К, а у некоторых - 128 К. В частности, за время порядка нескольких секунд можно полностью вкачать в физическую память любую задачу, что позволяет вполне удовлетворительно обеспечить диалог по упрощенной схеме разделения времени с постоянной длиной кванта /нормированного/. Значительное превышение размера физической памяти над памятью одной задачи позволяет добиваться хорошей загрузки процессора за счет размещения в памяти фоновой задачи. Дополнительные упрощения возникают из-за того, что машина однопроцессорная, страница - оптимальная порция для передачи информации на барабаны и диски /нет потребности в передаче сразу нескольких страниц/ и число параллельных каналов обмена невелико /при замещении страниц достаточно организации параллельного считывания и записи малого числа страниц/.

Техническая деталь: выбор величин r_i удобно производить так, чтобы они допускали представление 2^m, где m - целое, а критические значения n_i, при которых исчерпывается квант, были кратными r_i. В этом случае напряженно работающие блоки удается организовать с использованием битовой и масковой логики при сравнительно малой резидентной памяти.

Авторы считают своей приятной обязанностью поблагодарить В.Ю.Веретенова, М.И.Гуревича, А.В.Гусева, Р.З.Залялова, С.Г.Каданцева, О.Н.Ломидзе, А.П.Сапожникова, Б.Б.Сахарова, Г.Л.Семашко, совместно с которыми была создана современная диалогово-пакетная версия ОС "Дубна". На ее базе стала возможной экспериментальная проверка и детальная разработка изложенного здесь алгоритма.

Обкатке алгоритма способствовало также подключение к операционной системе концентратора терминалов, созданного В.В.Галактионовым, Е.Ю.Мазепой, Р.К.Микушкаускасом и В.П.Шириковым [13-15], которым мы также признательны.

ЛИТЕРАТУРА

1. Веретенов В.Ю. и др. ОИЯИ, 11-7059, Дубна, 1973.
2. Веретенов В.Ю. О новых возможностях ОС "Дубна". В сб.: Материалы семинара "Проблемы повышения эффективности БЭСМ-6" /июнь 1975/. Изд-во СЭИ, Иркутск, 1976, с.34-37.

3. Гусев А.В. и др. Новые возможности ОС "Дубна". В сб.: Материалы семинара "Проблемы повышения эффективности БЭСМ-6" /июнь 1975/. Изд-во СЭИ, Иркутск, 1976, с.29-34.
4. Силин И.Н. ОИЯИ, 11-9248, Дубна, 1975.
5. Bertin J., Ritout M., Rougier J.C. L'exploitation partagée des calculateurs. Dunod Paris, 1967. Русский перевод: Бертен Ж., Риту М., Ружие Ж. Работа ЭВМ с разделением времени. "Наука", М., 1970.
6. Конвей Р.В., Максвелл В.Л., Миллер Л.В. Теория расписаний. "Наука", М., 1975.
7. Говорун Н.Н. и др. Мониторная система "Дубна" для ЭВМ БЭСМ-6. Труды II Всесоюзной конф. по программированию. ВЦ СО АН СССР, Новосибирск, 1970, вып. ЖII, с.5-24.
8. Силин И.Н. ОИЯИ, 11-4655, Дубна, 1969, с.43-46.
9. Peter J. Denning. On Modeling Program Behaviour. AFIPS Conf.Proc., Montvale, N.J.,1972,vol.40, p.937.
10. NOS/BE1 System Programmer's Reference Manual. Pub. No. 60494100, Minnesota, 1977.
11. Badel M. Jacques Lerouques. Optimal Multiptogramming: Principles and Implementation. Lect.Notes Compt.Sci., 1978, 65, p.474-503.
12. Ломидзе О.Н., Силин И.Н. "Программирование", 1978, №3, с.55-58.
13. Галактионов В.В., Каданцев С.Г., Шириков В.П. ОИЯИ, 11-11264, Дубна, 1978.
14. Галактионов В.В., Мазепа Е.Ю. ОИЯИ, Р11-12580, Дубна, 1979.
15. Галактионов В.В., Микушкаускас Р.К. ОИЯИ, Р11-12871, Дубна, 1979.

Рукопись поступила в издательский отдел
25 августа 1979 года.