УДК 681.32

МЕТОДИКА ВЕРОЯТНОСТНОГО АНАЛИЗА НАБОРОВ ЗАДАЧ В ОДНОПРОЦЕССОРНЫХ СИСТЕМАХ РЕАЛЬНОГО ВРЕМЕНИ С ФИКСИРОВАННЫМИ ПРИОРИТЕТАМИ

В. П. Дашевский,
аспирант
Санкт-Петербургский институт
информатики и автоматизации РАН (СПИИРАН)

Предложена методика вероятностного анализа выполнимости периодических задач в системе реального времени. Методика позволяет гарантировать качество обслуживания каждой задачи в системе, где максимально возможная загрузка процессора превышает единицу, а средняя остается меньше единицы. Представленный алгоритм численного анализа является эффективным как в отношении машинного времени, так и в отношении требуемой оперативной памяти.

This paper describes probabilistic analysis of periodic real-time tasks with varying execution times. The analysis provides performance guaranties for the task sets with total maximum utilization higher than one while the average utilization remains less than one. Each task may me pre-empted by higher priority task and has deadline not grater than its period. Proposed algorithm is memory and time-efficient.

Введение

Системами реального времени (СРВ) принято считать такие системы, в которых наряду с логической правильностью вычислительного процесса необходимо обеспечивать также своевременность его протекания. При разработке и тестировании этих систем приходится учитывать не только функциональный, но и временной аспект.

В большинстве случаев процесс проектирования вычислительной системы реального времени начинается с абстрактного описания ее функциональных возможностей, указания ресурсных и временных ограничений. Затем весь спектр функциональных возможностей представляется в виде набора более простых задач, для выполнения которых выбираются процессорные элементы. В завершение для каждого процессорного элемента назначается дисциплина планирования, в соответствии с которой проводится анализ временных требований задач.

Во многих приложениях СРВ нарушение сроков выполнения задач может приводить к катастрофическим последствиям. По этой причине основные методики анализа систем реального времени, развитые в 1970–1980 годы, были ориентированы на наихудшее время выполнения программ (НВВП). Недостаток такого подхода заключается в том, что время выполнения программ сравнительно редко приближается к наихудшему. В большинстве случаев оно остается существенно меньше, поэтому процессоры оказываются слабо загруженными. Кроме того, методы повышения производительности вычислительных систем, такие как кэширование данных, предсказание переходов, конвейерная обработка с неявным запретом прерываний, использование контроллеров прямого доступа в память, лишь увеличивают разрыв между средним и наихудшим временем выполнения. Таким образом, консервативные методики анализа с использованием лишь НВВП накладывают все большие ограничения на эффективность использования аппаратных средств.

В последние годы появилось много новых приложений СРВ, в которых нарушение сроков задач не приводит к заметным последствиям, пока вероятность таких нарушений остается меньше некоторого порога. Таковы, например, различные приложения мультимедийных и телекоммуникационных систем. Некоторые авторы отмечают, что анализ устойчивости в системах «чистого» управления также допускает редкое нарушение сроков без заметной потери управляемости. Указанные особенности создают предпосылки для развития методик вероятностного анализа СРВ, с помощью которых можно повысить эффективность использования

аппаратуры, а значит снизить затраты на оборудование.

Методы вероятностного анализа, развитые ранее, опираются на простые модельные законы распределения времени выполнения, такие как равномерный или экспоненциальный закон распределения. Более точные модели используют произвольные распределения, однако одной из основных трудностей при работе с ними является быстро растущая с числом задач сложность анализа.

В данной работе рассматривается методика вероятностного анализа набора задач для одного процессора, адаптированная для широко применяемой дисциплины планирования с фиксированными приоритетами задач. Законы распределения времени выполнения задач могут быть выбраны произвольно, но при некоторых предположениях анализ может быть еще более упрощен без существенной потери точности. Анализ является эффективным как в отношении машинного времени, так и в отношении требуемой оперативной памяти.

Практически во всех исследованиях, посвященных СРВ, каждая из задач решается системой многократно. В первую очередь это связано со спецификой применения СРВ в качестве подсистем сбора информации и управления. Для описания задач традиционно используются несколько параметров: период повторения запросов на решение задачи, относительный срок выполнения, время выполнения, время блокировки из-за доступа к общим данным. Разными авторами допускаются различные предположения относительно значений этих параметров.

Методики анализа систем с детерминированными параметрами времени выполнения, рассмотренные в работах [2, 5–10, 13–15, 17], основываются на предположении, что время выполнения задачи постоянно. В системах с жесткими требованиями на выполнение задач в срок приходится рассчитывать систему исходя из НВВП для каждой задачи. Чаще всего это приводит к выводу о перегрузке системы и невозможности ее работы в реальном времени, однако, если производительности процессора хватает для наихудшего случая, то система оказывается слабо загруженной в среднем, что приводит к выводу о плохой эффективности ее использования.

Авторы [16] предлагают пять различных алгоритмов планирования для апериодических задач. Используемая ими модель задач предполагает случайные времена возникновения запросов на выполнения задач и постоянные времена выполнения, оцененные исходя из НВВП.

В работе [1] рассмотрена система, в которой времена выполнения задач могут быть случайными. На основе работы [7] авторами разработана методика вероятностного анализа набора задач, с помощью которой можно оценить вероятность нарушения срока для каждой задачи. Однако вычислительная сложность предложенной методики оказывается довольно высокой, поэтому авторы прибегают к приближенным вычислениям, используя центральную предельную теорему. Использование приближенных вычислений может приводить к завышенным оценкам вероятности выполнения задачи в срок, что делает проблематичным использование подобной методики для систем с жесткими требованиями. Авторами предложен также другой метод организации вычислений, при котором задачи с большим разбросом времени выполнения разбиваются на две подзадачи, первая из которых выполняется за время, не превышающее среднего времени выполнения исходной задачи, а вторая представляет собой случайно возникающий остаток ее выполнения. Первая подзадача планируется аналогично другим задачам с детерминированными характеристиками, а вторая ставится в очередь к спорадическому серверу. Показано, что с помощью подобных преобразований удается достигнуть более высокой эффективности использования процессоров, однако разделение далеко не всегда удается произвести на практике, что ограничивает применение этого метода.

В работе [11] авторы обобщают результаты классической работы [2] на случай задач со случайными временами выполнения. Каждая задача характеризуется периодом повторения запросов, функцией распределения времени выполнения, а также качеством обслуживания, представляющим собой необходимую вероятность выполнения задачи в срок. Помимо планирования задач система осуществляет контроль над их принятием в расписание, с помощью которого, во-первых, ни одна задача не использует больше процессорного времени, чем ей гарантировано, во-вторых, ни одна задача не принимается в расписание, если ее выполнение не может быть гарантировано. Подобный контроль обеспечивает защиту качества обслуживания задач с большим периодом (т. е. имеющих меньший приоритет в соответствии с RMS) от вытеснения задачами с меньшим периодом. Авторами предложен алгоритм анализа выполнимости набора задач при условии, что времена выполнения всех задач распределены равномерно. К сожалению, методика анализа не может быть легко обобщена на случай произвольного распределения времени выполнения задач.

Альтернативная методика анализа набора задач со случайным временем выполнения предложена в работе [3]. В ее основе также лежит анализ системы с детерминированными параметрами задач, развитый в работе [7]. Подобно работе [1] целью анализа является оценка нижней границы для вероятности выполнения задач в срок. Предлагаемая методика учитывает также влияние необходимости завершения предыдущих запросов на обработку текущего запроса, которое становится особенно заметным, когда средняя загрузка СРВ приближается к единице, а максимальная превосходит ее. Однако приводимые автором данные о сравнении результатов развитой методики с численным моделированием того же набора задач показывают, что она дает слишком заниженную оценку на вероятность выполнения задач в срок. Чем выше становится перегрузка системы, тем сильнее отличаются результаты анализа от результатов

моделирования. Из результатов моделирования, полученных автором, видно, что при стохастическом времени выполнения программ искомая вероятность выполнения в срок очень слабо зависит от постоянства фазовых соотношений между запросами на решение различных задач. Автором предложено обобщение методики анализа на случаи задач, в алгоритмах которых присутствуют непрерываемые критические секции, а также на случаи распределенных систем.

Работа [12] является развитием [1]. Однако авторы рассматривают вероятность гарантирования не отдельных задач, а всего набора в целом. Для этого им приходится рассматривать все возможные сочетания времен выполнения задач, в результате чего сложность анализа не позволяет исследовать большое число задач.

В работе [4] показано, что редкое нарушение сроков в задачах «чистого» управления не означает потери устойчивости в управлении, хотя традиционно многие задачи управления (в особенности опасными или дорогостоящими объектами) относят к задачам жесткого реального времени. Этот результат показывает, что надежность системы управления в реальном времени связана с жесткостью требований реального времени лишь косвенно, и вероятностный анализ может быть применен к системам с жесткими требованиями.

Анализ, приводимый в данной работе, является развитием методики, изложенной в работе [1]. По сравнению с [1] был улучшен алгоритм проверки контрольных точек: уменьшено число проверяемых точек, в которых приходится вычислять свертки, а также уменьшена сложность вычисления самих сверток. Эти упрощения позволяют проводить анализ всей системы задач, не прибегая к центральной предельной теореме, благодаря которой авторам [1] удается сократить объем вычислений, однако при этом теряются гарантии того, что полученная в результате анализа оценка вероятности является оценкой снизу.

Формулировка задачи вероятностного анализа

Для простоты изложения ограничимся рассмотрением одного процессора. Методику можно применить и в случае многопроцессорной системы, если предварительно задачи распределить по процессорам, а связи между ними специфицировать таким образом, что можно провести анализ для каждого процессора отдельно.

Предположения относительно системы задач. В качестве модели задач используется традиционная для большинства коммерческих операционных систем реального времени модель периодических задач с фиксированными приоритетами, удовлетворяющих следующим предположениям:

1) запросы на решение задач поступают в систему периодически; относительный срок, за который запросы должны быть обработаны, не превосходит периода следования запросов;

2) время выполнения задачи является случайной величиной, распределенной по некоторому, заранее выбранному закону; при этом максимальное время выполнения не превосходит периода повторения запросов;

3) времена выполнения различных задач являются статистически независимыми случайными величинами (в отличие от времен отклика);

4) обработка запроса на выполнение задачи прекращается принудительным образом, если она не укладывается в назначенный срок;

5) задача с более высоким приоритетом вытесняет менее приоритетную, не дожидаясь завершения ее выполнения;

6) время переключения контекста процессора между задачами пренебрежимо мало по сравнению с характерными временами их выполнения;

7) для каждой задачи множество задач, способных ее вытеснить, задается разработчиком системы при проектировании и не изменяется во время функционирования системы;

8) средняя загрузка процессора всеми задачами не превосходит единицы. Однако в силу случайного характера, а также большого разброса значений времени выполнения максимальная загрузка может быть больше единицы.

Поскольку в каждом периоде повторения время вычисления задачи меняется случайным образом, то часть параметров каждой задачи будет зависеть также от номера периода. В дальнейшем будем описывать j-й запрос $W_{i,j}$ на решение i-й задачи W_i следующими параметрами:

T_i — период повторения запросов на решение задачи;

D_i — относительный срок выполнения;

ϕ_i — приоритет задачи;

$r_{i,j}$ — момент поступления запроса на решение задачи;

$c_{i,j}$ — время выполнения запроса (без вытеснения другими задачами);

$d_{i,j}$ — срок завершения выполнения запроса;

$R_{i,j}$ — момент завершения выполнения запроса;

$\overline{c}_i$ — среднее время выполнения запроса;

c_i^+ — максимальное время выполнения запроса;

c_i^- — минимальное время выполнения запроса;

$\overline{u}_i$ — средняя загрузка процессора данной задачей;

u_i^+ — максимальная загрузка процессора данной задачей;

u_i^- — минимальная загрузка процессора данной задачей;

$f_i(x)$ — функция плотности распределения времени выполнения;

p_i — вероятность выполнения запроса в срок.

Приведенные выше условия в принятых обозначениях можно записать следующим образом:

$$c_{i,j} \le D_i \le T_i;$$

$$r_{i,j} = r_{i,1} + (j-1)T_i;$$

$$d_{i,j} = r_{i,j} + D_i;$$

$$\bar{c}_i = \lim_{j\to\infty} \frac{1}{j} \sum_{k=1}^{j} c_{i,k};$$

$$c_i^+ = \max c_{i,j};$$

$$c_i^- = \min c_{i,j};$$

$$\bar{u}_i = \frac{\bar{c}_i}{T_i};$$

$$u_i^- = \frac{c_i^-}{T_i};$$

$$u_i^+ = \frac{c_i^+}{T_i};$$

$$\bar{U} = \sum_{i=1}^{N} \bar{u}_i < 1;$$

$$U^+ = \sum_{i=1}^{N} u_i^+ > 1;$$

$$p_i = P(R_{i,j} \le d_{i,j}).$$

В добавление к принятым обозначениям будем также считать, что задачи пронумерованы в соответствии с их приоритетами:

$$i < j \Rightarrow \phi_i > \phi_j. \qquad (1)$$

Строго говоря, можно не накладывать ограничения на время выполнения задачи в предположении 2, однако естественно предположить, что каждая отдельно взятая задача не приводит к перегрузке процессора. Таким образом, целью анализа, в первую очередь, является определение частоты нарушения сроков при совместном выполнении задач, т. е. при условии, когда только случайное увеличение времени выполнения сразу нескольких задач в совокупности способно привести к перегрузке процессора.

Формулировка задачи вероятностного анализа СРВ. Задача формулируется следующим образом:

Для заданного набора независимых задач W_i, $i = 1, \ldots, N$ *с указанными вероятностными характеристиками, задаваемыми с помощью функций плотности распределения времени выполнения, требуется определить вероятность* p_i *завершения задачи* W_i *до назначенного ей срока* D_i. Если полученная вероятность окажется не меньше некоторого заранее заданного значения, то будем считать, что система способна справляться с решением данной задачи в реальном времени.

Сложность проведения анализа подобного рода заключается, как и в случае с детерминированными временами выполнения, в том, что каждая задача может вытесняться в процессе решения более приоритетными задачами, что приводит к увеличению времени отклика. Чем больше становится время отклика, тем большее число задач с более высоким приоритетом могут вытеснить текущую задачу, что приводит к еще большему увеличению времени отклика. Влияние этой положительной связи становится еще более трудно прогнозируемым, если время выполнения задачи может изменяться в широких пределах. В такой ситуации можно ожидать, что вероятностный анализ способен дать лучшие результаты, поскольку наиболее вероятное время отклика существенно отличается от максимально возможного.

Вероятностный анализ времени выполнения задач

Профили времени выполнения и операции с ними. Случайный характер времени выполнения задачи может быть описан с помощью функции распределения времени выполнения. По определению функция распределения времени выполнения задачи в точке x есть вероятность того, что время выполнения c не превзойдет значения x:

$$F(x) = P(c < x).$$

Функцию плотности распределения

$$f(x) = F'(x)$$

для краткости называют также *профилем времени выполнения* задачи [18]. Поскольку время выполнения всегда неотрицательно, то функции распределения определены на положительной полуоси.

В общем случае каждую задачу можно представить совокупностью некоторых элементарных выполняемых блоков, каждый из которых описывается своим профилем времени выполнения. Однако в данной статье ограничимся рассмотрением всей задачи как одного неделимого блока.

Для анализа набора задач необходимо уметь находить профиль суммарного времени выполнения двух и более задач. В общем случае задача является довольно трудной, поскольку суммарное время выполнения нескольких задач зависит от их порядка, количества вытеснений более приоритетными задачами и использования общих данных. Тем не менее в практике достаточно часто встречаются примеры слабо зависимых задач. В данном случае будем исходить из предположения 3 (см. ранее).

Основной операцией над профилями времени выполнения, применяемой для анализа совокупностей независимых задач, является операция свертки

$$h(x) = f_1(x) \otimes f_2(x) = \int_0^x f_1(y) f_2(x-y)\, dy, \qquad (2)$$

с помощью которой можно получить профиль суммы времен выполнения двух задач. Важными свойствами сверток являются:

- коммутативность — $f_1 \otimes f_2 = f_2 \otimes f_1$;
- ассоциативность — $(f_1 \otimes f_2) \otimes f_3 = f_1 \otimes (f_2 \otimes f_3)$;
- дистрибутивность — $(f_1 + f_2) \otimes f_3 = (f_1 \otimes f_3) + (f_2 \otimes f_3)$;

Для дальнейшего изложения полезен также профиль нулевого времени выполнения, или нулевой профиль выполнения, который будем обозначать

$\delta(x)$. Основное свойство нулевого профиля выполнения состоит в том, что для любого профиля $f(x)$ $f(x) \otimes \delta(x) = f(x)$.

В машинном представлении профиль времени выполнения описывается дискретным образом с помощью массива чисел:

$$a_k = P(h(k-1) < c \le hk), \quad k = 1, \ldots, M,$$

где h — шаг дискретизации; M — общее количество дискрет, выбираемое таким образом, чтобы покрыть исследуемый участок временной оси для каждой задачи.

Вероятность выполнения задачи за время x с дискретно заданным профилем a_k можно оценить снизу как

$$F(x) \ge \sum_{k=0}^{\left\lfloor \frac{x}{h} \right\rfloor} a_k. \qquad (3)$$

Из выражения (3), в частности, следует, что для оценки снизу вероятности выполнения задачи W_i достаточно массива размером $\left\lfloor \frac{D_i}{h} \right\rfloor + 1$.

Для вычисления свертки дискретно заданных профилей $f_{1,k}$ и $f_{2,k}$ интеграл в выражении (2) заменяется суммой:

$$f_{3,k} = \sum_{l=0}^{k} f_{1,l} \cdot f_{2,k-l}.$$

Нулевой профиль выполнения в дискретном случае определяется как

$$\delta_k = \begin{cases} 1, & k = 0 \\ 0, & k > 0, \end{cases}$$

так что для любого дискретного профиля f_k их свертка

$$(\delta \otimes f)_k = \sum_{l=0}^{k} \delta_l f_{k-l} = \delta_0 f_k + \sum_{l=1}^{k} \delta_l f_{k-l} = f_k$$

удовлетворяет определению нулевого профиля выполнения.

Задание профиля времени выполнения для всех задач требует значительного количества памяти, а также приводит к большому времени вычисления сверток [1]. Однако подобное усложнение вычислений не приводит к существенному повышению их точности. Точное измерение функции плотности распределения на практике представляет собой довольно сложную задачу, хотя в большинстве случаев разработчика интересует не столько точное значение, сколько порядок величины вероятности превышения срока.

Чтобы упростить задачу вычисления сверток над дискретными профилями, используем следующие два предположения.

1. Профили многих реальных алгоритмов имеют явно выраженный линейчатый характер, благодаря чему среди чисел, задающих дискретный профиль, сравнительно небольшое количество оказываются отличными от нуля. Естественно в такой ситуации задавать профиль не массивом чисел, а массивом пар вида (k_i, p_i), где k_i — номер дискрета, соответствующего времени выполнения hk_i, а p_i — вероятность наблюдения этого времени выполнения. Количество таких пар соответствует количеству различных ветвлений программы.

2. Профили времени выполнения, для которых максимальное время выполнения существенно отличается от среднего, должны обладать быстро убывающей функцией $f(x)$. В таком случае без существенной потери точности ее можно заменить дискретным профилем, который также описывается набором пар вида (k_i, p_i). Величина p_i представляет собой вероятность $P(hk_{i-1} < x \le hk_i)$. В этом случае количество пар также может быть сделано существенно меньше, чем количество дискрет, необходимое для описания произвольного профиля.

С помощью этих упрощений можно добиться сокращения объема памяти, необходимого для хранения профиля времени выполнения каждой задачи W_i. Вычисление свертки в таком случае представляет собой операцию над двумя профилями, первый из которых задан дискретным образом в виде массива чисел a_m, а второй задан парами чисел вида (k_i, p_i), $i = 1, \ldots, n$:

$$a'_m = \sum_{i=1}^{n} a_{m-k_i} p_i.$$

Таким образом, вычисление свертки в данном случае требует меньшего количества операций, чем над двумя дискретными профилями. Для анализа системы задач достаточно одного массива чисел, описывающего один дискретный профиль, к которому поочередно добавляются профили отдельных задач. Упрощенные профили позволяют сохранить соотношение (3) для оценки вероятности времени выполнения задачи, вытесняемой другими задачами.

Алгоритм вероятностного анализа задачи. Итак, задача анализа состоит в том, чтобы оценить снизу вероятность выполнения запроса $W_{i,j}$ в срок. Используемая для этого методика базируется на расчете необходимого времени, развитом в работе [7]. Для оценки вероятности снизу мы будем рассматривать критический участок, понятие которого введено в работе [2]. Предполагается, что критический участок начинается тогда, когда запросы на все задачи приходят одновременно. Без потери общности можно считать, что начало критического участка совпадает с началом отсчета на временной оси.

Для оценки вероятности удобно воспользоваться вспомогательной функцией $w_i(t)$, которая ограничивает время отклика системы сверху:

$$w_i(t) = c_{i,j} + \sum_{k=1}^{i-1} \sum_{l=1}^{s_k(t)} c_{k,l}, \qquad (4)$$

где

$$s_k(t) = \left\lceil \frac{t}{T_k} \right\rceil$$

представляет собой наихудшую оценку количества запросов задачи W_k, которые будут приняты в системе за время $[r_{i,j}, r_{i,j}+t)$ при любом $t>0$. Поскольку время выполнения каждого запроса является случайной величиной, то приходится явно суммировать $s_k(t)$ случайных величин $c_{k,l}$. Первая сумма в выражении (4), в соответствии с предположением (1), представляет собой суммирование времен всех более приоритетных задач, а вторая учитывает многократные запросы на выполнение одной задачи.

Завершение выполнения запроса $W_{i,j}$ происходит тогда, когда время, прошедшее с момента его поступления, становится не меньше необходимого времени

$$w_i(t) \le t$$

для некоторого t из отрезка $[r_{i,j}, r_{i,j}+D_i]$. Поскольку в зависимости от t количество слагаемых в выражении (4) меняется, то естественно разбить этот отрезок на участки, в которых функции $s_k(t)$, $k \in \{1, \ldots, i\}$ сохраняют постоянное значение. Границами этих участков являются, очевидно, точки временной оси, принимающие значения вида

$$l \cdot T_k, \quad l = 0, 1, 2, \ldots, \left\lfloor \frac{D_i}{T_k} \right\rfloor; \quad k = 0, 1, \ldots, i. \qquad (5)$$

Обозначим множество всех участков с постоянными $s_k(t)$ через $\boldsymbol{C}_i$. Множество их границ, описываемое выражением (5), с включенной точкой D_i обозначим $\boldsymbol{B}_i$:

$$\boldsymbol{B}_i = D_i \cup \{l \cdot T_k : l = 0, 1, 2, \ldots, \left\lfloor \frac{D_i}{T_k} \right\rfloor; \; k = 0, 1, \ldots, i\}.$$

Зная функции распределения отдельных задач и их количество на каждом участке, можно построить функцию распределения величины $w_i(t)$, вычислив их свертку.

Обозначим через $E(x)$ вероятность того, что $w_i(t) < x$. На каждом участке $h \in \boldsymbol{C}_i$ наибольшее значение $E(x)$ достигается для наибольшего $x \in h$, которое соответствует завершению участка постоянства $s_k(t)$ в момент прихода очередного запроса на решение одной из задач с более высоким приоритетом. Дискретный характер изменения количества запросов во времени приводит к разрыву первого рода для $E(x)$ на границах участков. Поскольку начало участка h связано с появлением нового запроса, то значение $E(x)$ справа от разрыва всегда оказывается меньше, чем слева. В общем случае значение $E(x)$ на правом конце участка h может быть как больше, так и меньше значения $E(x)$ на правом конце предыдущего участка (примыкающего к h слева). Это связано с тем, в каком соотношении находится длина участка h и среднее время выполнения одного или, возможно, нескольких запросов, добавившихся в его начале. Если среднее время обработки нового запроса оказывается меньше, чем длина участка h, то, в среднем, у системы появляется дополнительное время для выполнения запроса задачи W_i, и тогда вероятность выполнения задачи W_i к концу участка h становится больше. В противном случае появление новых запросов только уменьшает время, которое остается системе для выполнения задачи W_i, и тогда изменение $E(x)$ оказывается отрицательным.

Для того чтобы обработка запроса $W_{i,j}$ на критическом участке была завершена в срок, достаточно, чтобы она завершилась на любом из участков $h \in \boldsymbol{C}_i$. Поэтому можно оценить вероятность выполнения запроса снизу:

$$p_i \ge \max_{x \in h,\, h \in \boldsymbol{C}_i} E(x). \qquad (6)$$

Поскольку наибольшее значение $E(x)$, $x \in h$ достигается на правой границе участка h, то выражение (6) можно записать в виде

$$p_i \ge \max_{x \in \boldsymbol{B}_i} E(x-0). \qquad (7)$$

Чтобы упростить вычислительную сложность анализа, перебор значений множества $\boldsymbol{B}_i$ удобно вести в порядке возрастания. В этом случае вычисление сверток для функции распределения $E(x)$ можно производить последовательно, выполняя по одной операции свертки на каждую точку из $\boldsymbol{B}_i$. Алгоритм вероятностного анализа представлен на

Входные данные
Набор задач $S = \{W_i\}$, их периоды T_i, относительные сроки D_i, функции плотности распределения $f_i(x)$ и приоритеты ϕ_i, $i = 1, 2, \ldots, N$.

Выходные данные
Нижняя оценка p_i вероятности того, что запрос на решение задачи W_i будет выполнен в срок, в соответствии с формулой (7)

Алгоритм

for each W_i, $1 \le i \le N$ do
  $g_i(x) = \delta(x)$
  $p_i = 0$
  $X = 0$
  $t_k = 0$, $1 \le k \le i$
  while $X \le D_i$ do
    $p = \int_0^X g_i(x)\,dx$
    $p_i = \max\{p_i, p\}$
    $L = \{m : t_m = \min_{1 \le k \le i} t_k\}$
    $l : \phi_l = \max_{m \in L} \phi_m$
    $X = t_l$
    $g_i(x) = g_i(x) \otimes f_l(x)$
    $t_l = t_l + T_l$
  end while
end for

■ ***Рис. 1.*** *Алгоритм вероятностного анализа набора задач*

рис. 1. Общее количество вычислений сверток можно оценить по формуле

$$|\boldsymbol{B}_i| = \sum_{k=1}^{i} \left\lceil \frac{D_i}{T_k} \right\rceil. \qquad (8)$$

Оценка сложности алгоритма. Основной вклад в сложность алгоритма вносит вычисление сверток и вычисление вероятности $P(c < x)$ для дискретного профиля, полученного в очередной контрольной точке из множества $\boldsymbol{B}_i$. Общее количество контрольных точек определяется формулой (8).

В общем случае вычислительная сложность алгоритма оценивается

$$C = \sum_{x \in B_i} C_x,$$

где C_x — сложность анализа в контрольной точке x. В каждой точке x приходится вычислять вероятность p_i, а также очередную свертку для $g_i(x)$. Значение C можно оценить сверху так:

$$C \le |\boldsymbol{B}_i| \cdot \max_{x \in B_i} C_x \le |\boldsymbol{B}_i| (C_{pr} + C_{conv}),$$

где C_{pr}, C_{conv} — оценки сверху для сложности вычисления вероятности и свертки, соответственно.

Сложность вычисления вероятности можно оценить сверху величиной

$$C_{pr} = n_i \rho_{add}.$$

Сложность вычисления свертки можно оценить сверху величиной

$$C_{conv} = M_i n_i (\rho_{add} + \rho_{mul}).$$

В приведенных формулах M_i обозначено количество пар чисел, задающих упрощенный профиль для задачи W_i; ρ_{add} — относительный вес сложности операции сложения; ρ_{mul} — относительный вес сложности операции умножения; $n_i = \left\lceil \frac{D_i}{h} \right\rceil$ — количество дискрет в профиле выполнения задачи W_i; h — шаг дискретизации профиля выполнения в машинном представлении.

Таким образом, общая сложность вероятностного анализа для задачи W_i оценивается сверху выражением

$$n_i \left(M_i \rho_{mul} + (M_i + 1)\rho_{add}\right) \sum_{k=1}^{i} \left\lceil \frac{D_i}{T_k} \right\rceil.$$

В данном случае сложность линейно зависит от количества дискрет в профиле $g_i(x)$. В случае, когда для задания профиля задачи W_i используется полный дискретный профиль, задаваемый массивом, сложность свертки возрастает до

$$C_{conv} = \frac{1}{2} n_i^2 (\rho_{mul} + \rho_{add}),$$

и общая сложность анализа оценивается сверху формулой

$$\left(\frac{1}{2} n_i^2 (\rho_{mul} + \rho_{add}) + n_i \rho_{add}\right) \sum_{k=1}^{i} \left\lceil \frac{D_i}{T_k} \right\rceil.$$

В таком случае сложность зависит от количества дискрет в профиле $g_i(x)$ квадратичным образом.

Объем памяти, необходимый для проведения анализа, можно оценить следующим образом. Для вычисления сверток требуется память для хранения одного исходного дискретного профиля и одного результирующего дискретного профиля, который впоследствии заменяет исходный. Для хранения упрощенных профилей каждой задачи требуется M_i пар чисел, первое из которых целое, а второе — вещественное. Таким образом, объем памяти, необходимой для анализа N задач, равен

$$2 z_{real} \max_{i=1, \ldots, N} n_i + \sum_{i=1}^{N} M_i (z_{integer} + z_{real}),$$

где $z_{integer}$, z_{real} — объем памяти, необходимый для хранения целого и вещественного чисел соответственно.

В случае, когда для задания профиля задачи W_i используется полный дискретный профиль, задаваемый массивом, необходимый объем памяти возрастает до

$$2 z_{real} \max_{i=1, \ldots, N} n_i + \sum_{i=1}^{N} n_i z_{real}.$$

Сравнение результатов вероятностного анализа с результатами численного моделирования

С помощью методики вероятностного анализа можно получить оценку снизу для вероятности выполнения задачи в срок. Поскольку методика основана на рассмотрении критического участка, как это принято при анализе СРВ с детерминированными параметрами задач, то вероятность p_i для каждой задачи рассчитывается исходя из наихудшего числа ее вытеснений более приоритетными задачами. В действительности же наихудшее число вытеснений реализуется не для каждого запроса на решение задачи, поэтому доля запросов, обработанных в срок, оказывается больше значения p_i. Тем не менее можно ожидать, что дополнения этих значений до единицы окажутся одного порядка.

Для оценки точности методики вероятностного анализа была реализована программа прямого моделирования набора задач, которая в течение длительного времени набирает статистику обработки в срок периодически поступающих в систему запросов. В качестве примера для сравнения был использован набор задач, характеристики которого приведены в табл. 1. Все задачи в этом наборе обладали функциями распределения вида

■ ***Таблица 1.*** *Набор задач для сравнения*

i	T_i	D_i	c_i^-	$\bar{c}_i$	c_i^+	u_i^-	$\bar{u}_i$	u_i^+
1	100,0	100,0	10,0	19,9889	100,0	0,1000	0,1999	1,0000
2	150,0	150,0	12,0	23,9867	120,0	0,0800	0,1599	0,8000
3	200,0	200,0	12,0	23,9867	120,0	0,0600	0,1199	0,6000
4	600,0	600,0	20,0	39,9778	200,0	0,0333	0,0666	0,3333
Всего:						0,2733	0,5464	2,7333

■ ***Таблица 2.*** *Результаты численного эксперимента*

	Вероятностный анализ	Прямое моделирование
i	p_i	Доля запросов, выполненных в срок
1	1,0000000	1,00000000 ± 0,0000000
2	0,9989125	0,99959300 ± 0,0000090
3	0,9954908	0,99898873 ± 0,0000164
4	0,9999913	0,99999588 ± 0,0000015

$$F(x) = \begin{cases} 0, & x < c^-, \\ \dfrac{1-\exp\left(-\dfrac{x-c}{T}\right)}{1-\exp\left(-\dfrac{c^+-c^-}{T}\right)}, & c^- \le x \le c^+, \\ 1, & x > c^+. \end{cases}$$

Результаты обеих методик, представленные в табл. 2, позволяют сделать следующие выводы:

- вероятность, полученная с помощью рассматриваемой методики, действительно не больше значения, полученного в результате численного эксперимента;
- вероятность, полученная с помощью рассматриваемой методики, по порядку величины совпадает с долей успешно обработанных запросов, полученной с помощью численного моделирования; наблюдаемые отличия (в особенности у задач 2 и 3) связаны с тем, что анализируется лишь критический участок, из-за чего вероятность занижается;
- вероятность решения в срок задачи с меньшим приоритетом может быть больше вероятности решения в срок задачи с большим приоритетом, как в случае задач 3 и 4; методика вероятностного анализа позволяет замечать подобные особенности.

Таким образом, методика вероятностного анализа действительно позволяет с приемлемой точностью оценить вероятность выполнения задач.

Заключение

В статье рассмотрена методика вероятностного анализа совокупности периодических задач, выполняемых на одном процессоре в СРВ. С ее помощью можно получить нижнюю оценку для вероятности обработки в срок каждой задачи. Вероятностное описание позволяет расширить область применения традиционных дисциплин планирования на случай, когда максимальная загрузка СРВ становится больше единицы, и методики анализа, развитые ранее для систем с детерминированными параметрами, становятся неприменимы.

Наряду с изложением алгоритма анализа набора задач получена оценка его сложности, а также оценка объема оперативной памяти, требуемой для хранения данных. Сравнение результатов работы алгоритма с численным моделированием показывает, что оценка вероятности оказывается близкой к численному эксперименту.

Литература

1. **Tia T.-S., Deng Z., Shankar M., Storch M., Sun J., Wu L.-C., Lui J.W.-S.** Probabilistic Performance Guarantee for Real-Time Tasks with Varying Computation Times // Proceedings of the Real-Time Technology and Applications Symposium. — Chicago, Illinois. — May 1995. — IEEE. — P. 164–173.
2. **Lui C.L. and Layland J. W.** Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment // Journal of the ACM. — 1973. — P. 46–61.
3. **Gardner M.** Probabilistic Analysis and Scheduling of Critical Soft Real-Time Systems // PhD Dissertation. — University of Illinois at Urbana-Champain. — 1999.
4. **Cervin A.** Analyzing the Effects of Missed Deadlines in Control Systems // Real-Time Graduate Student Conference. — Lund, March 8-9, 2001. http: //www.artes.uu.se/events/gsconf01/ papers/cervin.pdf.
5. **Audsley N. C.** Deadline Monotonic Scheduling. YCS.146 // Computer Science, Department, University of York, September 1990.
6. **Audsley N. C., Burns A., Richardson M. F., Wellings A. J.** Hard Real-Time Scheduling: The Deadline-Monotonic Approach Proceedings 8th IEEE Workshop on Real-Time Operating Systems and Software. — Atlanta, USA, 15–17 May 1991.
7. **Lehoczky J., Sha L. and Ding Y.** The Rate-Monotonic Scheduling Algorithm: Exact Characterization and Average Case Behavior // Proceedings IEEE Real-Time Systems Symposium, Santa Monica, California. IEEE Computer Society Press. — 5–7 December 1989. — P. 166–171.
8. **Burns A.** Preemptive Priority Based Scheduling: an Appropriate Engineering Approach // Advances in Real-Time Systems, Prentice Hall. — 1995. — P. 225–248.
9. **Sha L., Rajkumar R. and Lehoczky J.** Priority Inheritance Protocols: An Approach to Real-Time Synchronization // IEEE Transactions on Computers. — 39(9): 1175–1185. — September 1990.
10. **Sha L., Rajkumar R., Lehoczky J., Ramamritham K.** Mode Change Protocols for Priority-Driven Preemptive Scheduling // CMU/SEI-88-TR-34. — Software Engineering Institute. — November 1988.
11. **Atlas A. K. and Bestavros A.** Statistical rate monotonic scheduling // Proceedings of the 19th Real-Time System Symposium. — December 1998. — P. 123–132.
12. **Hu Z. S., Zhou T. and Sha E. H.-M.** Estimating probabilistic timing performance for real-time embedded systems // IEEE Transactions on Very Large Scale Integration (VLSI) Systems. — 9(6): 833–844. — December 2001.
13. **Audsley N. C., Burns A., Richardson M. F., Tindell K. W. and Wellings A. J.** Applying New Scheduling Theory to Static Priority Pre-emptive Scheduling // Software Engineering Journal. — 8(5): 284–292. — September 1993.
14. **Audsley N. C., Burns A., Davies R. I., Tindell K. W. and Wellings A. J.** Fixed Priority Pre-emptive Scheduling: An Historical Perspective // Journal of Real-Time Systems. — Vol. 8. — 1995. — P. 173–198.
15. **Stankovic J. A., Spuri M., Di Natale M., Butazzo G.** Implications of Classical Scheduling Results for Real-Time Systems // IEEE Computer. — June 1995. — P. 16–25.
16. **Spuri M., Butazzo G.** Scheduling Aperiodic Tasks in Dynamic Priority Systems // Journal of Real-Time Systems. — Vol. 10. — N 2. — 1996. — P. 1979–2012.
17. **Joseph M. and Pandya P.** Finding response times in a real-time system // The Computer Journal — British Computer Society. — 29(5): 390–395. — October 1986.
18. **Bernat G., Colin A., Petters S.** WCET Analysis of Probabilistic Hard Real-Time Systems // The 23rd IEEE International Real-Time Systems Symposium. — December 3–5. — 2002. — Austin, Texas (USA).