Math-Net.Ru

Общероссийский математический портал

Э. Гирлих, М. М. Ковалев, В. М. Котов, Алгоритм для задачи распределения работ, выполняемых в реальном времени при наличии дополнительной информации, *Дискрет. матем.*, 2003, том 15, выпуск 4, 133–140

DOI: <https://doi.org/10.4213/dm222>

Использование Общероссийского математического портала Math-Net.Ru подразумевает, что вы прочитали и согласны с пользовательским соглашением
<http://www.mathnet.ru/rus/agreement>

Параметры загрузки:

IP: 3.91.87.62

26 апреля 2023 г., 15:04:20

Алгоритм для задачи распределения работ, выполняемых в реальном времени при наличии дополнительной информации

© 2003 г. Э. Гирлих, М. М. Ковалев, В. М. Котов

Задача распределения работ в системе с m идентичными параллельными процессорами, при котором минимизируется загрузка максимально загруженного процессора, является NP-трудной проблемой. Для этой задачи разработано много приближенных алгоритмов, но для версии этой задачи, в которой работы поступают и распределяются в реальном времени, нельзя разработать приближенный алгоритм, гарантированная оценка которого меньше $1 + 1/\sqrt{2}$ при $m \geq 4$ и стремится к 1,837 при $m \rightarrow \infty$.

В статье исследуется версия задачи, в которой работы поступают одна за другой и решение о их распределении принимается в реальном времени при дополнительном условии, что суммарная длительность работ известна. Для такой версии задачи предлагается алгоритм с гарантированной оценкой, равной $5/3$.

1. Введение

Задача распределения работ в системе с m идентичными параллельными процессорами, при котором минимизируется загрузка максимально загруженного процессора, является NP-трудной проблемой [1]. Рассматривается n независимых работ с длительностями $p_1, \dots, p_n$ их выполнения на процессоре. Работы должны быть распределены на m идентичных параллельных процессорах так, чтобы минимизировать загрузку максимально загруженного процессора (время до завершения последней выполняемой работы). Наиболее известными приближенными алгоритмами для этой задачи являются алгоритмы, изложенные в [1–4]. Понятно, что рассматриваемая задача может быть сформулирована как задача о разбиении, состоящая в том, чтобы разбить множество из n чисел на m непересекающихся подмножеств так, чтобы максимальная из сумм чисел в подмножествах была минимальной.

Впервые версия задачи, в которой работы поступают по очереди и каждая поступившая работа должна быть сразу направлена на один из m процессоров, была рассмотрена в 1966 г. В этой задаче нет никакой информации о количестве и длительности следующих работ, а сделанные уже назначения не могут быть изменены. В [1] показано, что если алгоритм назначает очередную работу на наименее загруженный процессор, то гарантированная оценка такого алгоритма равна $2 - 1/m$ (то есть время выполнения работ при использовании этого алгоритма не более чем в $2 - 1/m$ раза хуже результата в сформулированной выше задаче об оптимальном разбиении для этих работ). В [6] доказана нижняя

оценка для такого алгоритма, равная $2 - 1/m$ для $m = 2, 3$ и $1 + 1/\sqrt{2}$ для $m \geq 4$. В [5] эта оценка была доведена до 1,837 для достаточно больших m . Начиная с 1991 г., было предложено несколько алгоритмов для задачи, в которой работы поступают и распределяются последовательно, лучший из них [7] имеет оценку 1,923 для всех m .

Однако на практике встречаются задачи, представляющие собой что-то среднее между версией, в которой почти ничего неизвестно, и версией, в которой сразу известна вся информация о работах. Это значит, что в этих задачах имеется некая дополнительная общая информация. Такой информацией может быть

- общее время выполнения работ,
- информация о возможном законе распределения времени выполнения или верхние и нижние границы возможного времени выполнения работ,
- возможность получать заранее информацию о времени выполнения фиксированного числа очередных работ или наличие буфера фиксированной длины.

Будем называть такие задачи задачами с последовательно поступающими работами при наличии дополнительной информации.

До сегодняшнего для исследовано очень малое число алгоритмов для таких задач дискретной оптимизации (см. [8–10]). В данной работе предлагается алгоритм для задачи с последовательно поступающими работами в случае, когда суммарная длительность выполнения этих работ известна заранее. Доказано, что гарантированная оценка для предлагаемого алгоритма равна $5/3$.

Приведем описание алгоритма.

Предполагается, что работы поступают по очереди, и пусть t — суммарная длительность выполнения всех работ. Не умаляя общности будем считать, что все длительности нормированы так, что $t/m = 1$.

Через C^* будем обозначать время выполнения работ для оптимального решения задачи.

Для описания алгоритма введем следующую классификацию работ и процессоров.

Работа называется малой, если ее длительность (выполнения) не превосходит $1/3$, средней, если ее длительность лежит в интервале $(1/3, 2/3]$, не очень большой, если ее длительность лежит в интервале $(2/3, 5/6]$, большой, если ее длительность лежит в интервале $(5/6, 1]$, и наконец, работа называется очень большой, если ее длительность превосходит 1.

Процессор называется свободным, если его загруженность равна 0, малозагруженным, если его загруженность не превосходит $1/3$, среднезагруженным, если его загруженность лежит в интервале $(1/3, 2/3]$, сильнозагруженным, если его загруженность лежит в интервале $(2/3, 1]$, и наконец, перегруженным, если его загруженность больше 1.

Опишем алгоритм в терминах правила выбора процессора для текущей работы. Это правило учитывает класс текущей работы. Алгоритм состоит из двух стадий и выглядит следующим образом.

Первая стадия (пока существует хотя бы один свободный процессор).

- (1) Поступившая работа A очень большая. Она распределяется на процессор из числа свободных.

- (2) Поступившая работа A большая. Она распределяется на процессор из числа среднезагруженных, если такой процессор существует. Если класс среднезагруженных процессоров пуст, то работа распределяется на процессор из числа малозагруженных, если такой существует. Если класс малозагруженных процессоров пуст, то работа A распределяется на процессор из класса свободных.
- (3) Поступившая работа A не очень большая. Ищется процессор из класса сильнозагруженных, на который распределена единственная работа, причем эта работа относится к числу не очень больших, и на него распределяется работа A . Если такого процессора не находится, то работа распределяется на процессор из числа свободных.
- (4) Поступившая работа A средняя. Ищется процессор из класса сильнозагруженных, на который распределена единственная работа, причем эта работа из числа больших, и на него распределяется работа A . Если такого процессора не находится, то работа A распределяется на процессор из числа свободных.
- (5) Поступившая работа A малая. Ищется процессор из класса сильнозагруженных, на который распределена единственная работа из числа больших, и на него распределяется работа A . Если такого процессора не находится, то работа A распределяется на процессор из числа малозагруженных или, если этот класс пуст, на процессор из числа свободных.

Вторая стадия (свободных процессоров нет).

- (1) Поступившая работа A очень большая. Работа распределяется на наименее загруженный процессор.
- (2) Поступившая работа A не очень большая или большая. Ищется процессор из числа сильнозагруженных такой, что его загруженность после распределение на него работы A не превысит величины $5/3$. Если такого процессора нет, то ищется процессор из числа среднезагруженных, затем из числа малозагруженных и, наконец, наименее загруженный процессор (если предыдущие классы процессоров пусты), и на него распределяется работа A .
- (3) Поступившая работа A средняя. Работа A распределяется на наиболее загруженный процессор из числа сильнозагруженных. Если этот класс пуст, то ищется процессор из числа слабозагруженных или наиболее загруженный процессор из числа среднезагруженных (если предыдущие классы пусты), и на него распределяется работа A .
- (4) Поступившая работа A малая. Работа A назначается на процессор из числа слабозагруженных. Если этот класс пуст, то ищется наиболее загруженный процессор из числа сильнозагруженных или из числа среднезагруженных (если предыдущие классы пусты), и на него распределяется работа A .

2. Анализ точности алгоритма

Приведем некоторые свойства классов процессоров.

Свойство 1. Из условия $t/m = 1$ следует, что перед любым шагом алгоритма существует процессор, нагрузка которого не превосходит 1.

Опишем свойства, которыми обладают классы процессоров после первой стадии алгоритма.

Свойство 2. (1) Каждый процессор из класса перегруженных удовлетворяет следующим условиям:

- на него распределена только одна работа из числа очень больших;
- на него распределено ровно две работы, причем обе не очень больших;
- на него распределена одна большая работа, а также одна или несколько средних и малых работ.

(2) На каждый процессор из класса сильнозагруженных распределена одна не очень большая или большая работа.

(3) Существует не более одного процессора из класса сильнозагруженных, на который распределена одна не очень большая работа.

(4) Мощность класса малозагруженных процессоров не превышает единицы.

Свойство 3. Пусть после первой стадии алгоритма имеется k , $k \geq 4$, перегруженных процессоров. Тогда выполняется одно из следующих условий.

(1) Имеется не менее $k - 2$ среднезагруженных и малозагруженных процессоров и не более одного сильнозагруженного процессора, причем на него распределена одна работа и эта работа из числа не очень больших.

(2) Имеется не менее $k - 1$ сильно загруженных процессоров, на которые распределены большие работы.

Лемма 1. Загрузка любого из перегруженных процессоров не превышает величины $5C^*/3$.

Доказательство. Утверждение очевидно в случае, когда на первой стадии не поступило очень больших работ. В случае, если такая работа A поступила, причем $A \geq 5/3$, то утверждение леммы следует из неравенства $C^* \geq A$.

Докажем теперь некоторые свойства, которыми обладают классы процессоров после каждого шага второй стадии.

Лемма 2. Пусть после очередного шага второй стадии алгоритма имеется k , $k \geq 4$, перегруженных процессоров. Тогда выполняется одно из следующих условий.

(1) Имеется не менее $k - 2$ среднезагруженных или малозагруженных процессоров и не более одного сильнозагруженного процессора.

(2) Имеется не менее $k - 1$ сильнозагруженных процессоров, на которые распределены большие работы, и не более одного сильнозагруженного процессора, на которые распределена одна работа, и эта работа не очень большая.

Доказательство. Доказательство проведем по индукции. После первой стадии алгоритма утверждение справедливо. Предположим, что оно справедливо перед очередным шагом второй стадии алгоритма и покажем, что оно останется справедливым и после этого шага. Рассмотрим две возможности А и Б.

(А) Имеется хотя бы один среднезагруженный процессор.

Если поступила не очень большая, большая или очень большая работа, то один из процессоров становится перегруженным.

Если поступила средняя работа, то возможны следующие ситуации.

- (1) Имеется хотя бы один сильнозагруженный процессор. Тогда этот процессор после распределения в него текущей работы переходит в класс перегруженных.
- (2) Имеется малозагруженный процессор. Тогда этот процессор после распределения в него текущей работы становится либо среднезагруженным, либо сильнозагруженным.
- (3) Имеются только среднезагруженные процессоры. Тогда наиболее загруженный из них процессор после распределения в него текущей работы переходит в класс сильнозагруженных или перегруженных процессоров.

Если поступила малая работа, то возможны следующие ситуации.

- (1) Имеется малозагруженный процессор. Тогда этот процессор после распределения в него текущей работы переходит в класс среднезагруженных или остается слабозагруженным.
- (2) Имеется сильнозагруженный процессор. Тогда этот процессор после распределения в него текущей работы переходит в класс перегруженных процессоров или остается сильнозагруженным.
- (3) Имеются только среднезагруженные процессоры. Тогда наиболее загруженный среди них процессор после распределения в него текущей работы переходит в класс сильнозагруженных процессоров или остается среднезагруженным.

Следовательно, утверждение остается справедливым и после выполнения очередного шага, так как число сильнозагруженных процессоров по-прежнему не превосходит единицы. Действительно, это число могло увеличиться только тогда, когда класс сильнозагруженных процессоров был пуст.

(Б) Класс среднезагруженных процессоров пуст.

Если поступила средняя работа, то после ее назначения на наиболее загруженный процессор из класса сильнозагруженных процессоров (а его загрузка не меньше $5/6$), он переходит в класс перегруженных процессоров.

Если поступила малая работа, то после ее назначения на наиболее загруженный процессор из класса сильнозагруженных, он переходит в класс перегруженных процессоров или остается сильнозагруженным.

Если поступила не очень большая или большая работа A , то возможны следующие ситуации.

- (1) Имеется сильнозагруженный процессор B такой, что

$$|A| + |B| \leq 5/3$$

(здесь и далее $|A|$ означает длительность выполнения работы A , а $|B|$ — загрузка процессора B). Тогда текущая работа распределяется на B , и процессор переходит в класс перегруженных процессоров.

- (2) Имеется малозагруженный процессор. После распределения текущей работы на этот процессор он переходит в класс перегруженных или сильнозагруженных.
- (3) Для всех имеющихся сильнозагруженных процессоров B выполняется неравенство

$$|A| + |B| > 5/3.$$

Тогда текущая работа назначается на наименее загруженный процессор, и он переходит в класс перегруженных процессоров.

Если поступила очень большая работа, то она распределяется на наименее загруженный процессор, и он переходит в класс перегруженных процессоров.

Следовательно, в этом случае после выполнения шага класс среднезагруженных процессоров остается пустым.

Таким образом, структура классов остается прежней.

Лемма 3. *Предположим, что на некотором шаге алгоритма имеется не более трех процессоров, не являющихся перегруженными. Тогда для очередной работы A найдется такой процессор B , что загруженность его после распределения на него очередной работы не превысит $5C^*/3$.*

Доказательство. Пусть вначале имеется три процессора B_1, B_2, B_3 , не являющихся перегруженными. Если очередная работа A является малой или средней, то утверждение очевидно. Пусть A является не очень большой, большой или очень большой.

- (1) Предположим, что A — не очень большая или большая работа и утверждение леммы не верно, то есть

$$|B_i| + |A| > 5/3, \quad i = 1, 2, 3.$$

Отсюда следует, что $|B_i| > 2/3, i = 1, 2, 3$, следовательно,

$$|B_1| + |B_2| + |B_3| > 5/3 + 2/3 + 2/3 \geq 3.$$

Получаем противоречие.

- (2) Предположим, что A — очень большая работа и утверждение леммы не верно. Из условия леммы следует, что существует процессор B с загрузкой $|B| < 2/3$. Понятно, что $C^* \geq |A|$. Следовательно,

$$(|B| + |A|)/|A| = |B|/|A| + 1 \leq 2/3 + 1 \leq 5/3.$$

Поэтому

$$|B| + |A| \leq 5C^*/3.$$

Доказательство в случае, когда имеется два процессора, не являющихся перегруженными, проводится аналогично.

Теорема 1. *Описанный выше алгоритм имеет гарантированную оценку $5/3$.*

Доказательство. Покажем, что для каждого перегруженного процессора P , полученного на текущем шаге, выполняется неравенство

$$|P| \leq 5C^*/3.$$

Это очевидно, когда поступила малая или средняя работа.

Рассмотрим ситуацию, когда поступила не очень большая, большая или очень большая работа, а класс среднезагруженных процессоров пуст (в противном случае см. лемму 1). Это может быть только ситуация, когда после первой стадии работы алгоритма каждый процессор либо сильнозагруженный, либо перегруженный. Это значит, что на каждый процессор распределена либо одна очень большая работа, либо одна большая, либо две не очень больших работы.

Если текущая работа A не очень большая или большая, то в оптимальном решении должен быть процессор, содержащий две не очень большие работы. Поэтому

$$C^* \geq 4/3,$$

тогда как загрузка полученного перегруженного процессора не превосходит 2. Следовательно, для процессора P общая загруженность не превышает $5C^*/3$.

Если текущая работа из числа очень больших, то в оптимальном решении должен быть процессор, на который назначена не очень большая и большая работы. Поэтому

$$C^* \geq \max(3/2, |A|),$$

тогда как загрузка полученного перегруженного процессора не превосходит $1 + |A|$. Пусть

$$(1 + |A|)/(3/2) \geq 5/3.$$

Тогда

$$|A| \geq 3/2,$$

и следовательно,

$$(1 + |A|)/|A| \leq 1/|A| + 1 \leq 2/3 + 1 \leq 5/3.$$

Список литературы

1. Graham R. L., Bounds for certain multiprocessor anomalies. *Bell System Tech.* (1966) **45**, 1563–1581.
2. Graham R. L., Bounds on multiprocessing timing anomalies. *SIAM J. Appl. Math.* (1969) **17**, 263–269.
3. Coffman E. G., Garey M. R., Johnson D. S., An application of bin-packing to multiprocessor scheduling. *SIAM J. Comput.* (1978) **7**, 1–17.
4. Hochbaum D. S., Shmoys D., Using dual approximation algorithms for scheduling problems: theoretical and practical results. *J. ACM* (1987) **34**, 144–162.
5. Bartal Y., Karloff H., Rabany Y., A better lower bound for on-line scheduling. *Inform. Process. Lett.* (1994) **50**, 113–116.

6. Faigle U., Kern W., Turán G., On the performance of on-line algorithms for particular problems. *Acta Cyclic.* (1989) **9**, 107–119.
7. Albers S., Better bounds for online scheduling. *Proc. 29th Annual ACM Symp. Theory Computing*, 1997, 130–139.
8. Grove E. F., Online binpacking problem with lookahead. *Proc. 6th Annual ACM–SIAM Symp. Discrete Algorithms*, 1995, 430–436.
9. Mao W., Kincaid R. K., A look-ahead heuristic for scheduling jobs with release dates on a single machine. *Comp. Operat. Res.* (1994) **10**, 1041–1050.
10. Kellerer H., Kotov V., Speranza M. G., Tuza Z., Semi on-line algorithms for the partition problem. *Operat. Res. Lett.* (1997) **21**, 235–242.

Статья поступила 18.01.2001.