# О построении расписаний выполнения параллельных задач на группах кластеров с различной производительностью

*Жук С.Н.,*
*Princeton University (szhuk@princeton.edu)*

**Аннотация.** Предложен онлайновый алгоритм распределения параллельных задач на группе кластеров с различными производительностями процессоров и показано, что он гарантирует для любого потока задач построение расписания, отличающегося от оптимального не более чем в 2e раз.

**Ключевые слова:** параллельные вычисления, балансировка нагрузки, расписания выполнения задач для кластеров с различной производительностью.

## *1. Введение*

Рассмотрена задача построения расписаний выполнения параллельных задач на группах кластеров с различной производительностью процессоров. Имеется группа кластеров с различным числом процессоров и различной производительностью процессоров (у разных кластеров) и некоторое число многопроцессорных задач, каждая из которых характеризуется требуемым числом процессоров и требуемым нормализованным временем выполнения (временем выполнения на кластере единичной производительности). Необходимо составить расписание распределения задач по кластерам с целью минимизации общего времени выполнения. Предполагается, что выполнение задачи не может быть прервано во время своего выполнения.

Данная задача является NP-трудной (даже для частных случаев). Соответственно, неизвестны эффективные алгоритмы ее точного решения. В настоящей статье предложен приближенный алгоритм, позволяющий быстро находить приближенное решение и получены оценки его точности.

Задача составления расписания для мультипроцессорных задач на группе кластеров тесно связана с задачей упаковки прямоугольников в несколько полос. Кластеры можно рассматривать как полубесконечные полосы с шириной, соответствующей общему числу процессоров, а задачи – как прямоугольники с шириной, соответствующей требуемому числу



процессоров, и высотой, соответствующей требуемому времени. Единственным существенным отличием является то, что в том случае, когда рассматриваются прямоугольники, соответствующая задача должна занимать последовательные процессоры, в то время как в действительности задача может занимать произвольное множество процессоров. Принципиальным для данной работы является тот факт, что кластеры могут иметь различную производительность. Это существенно, поскольку даже для задачи построения расписаний для однопроцессорных машин с различными производительностями результаты далеки от окончательных [1,2].

В данной работе предложен онлайновый алгоритм распределения параллельных задач на группе кластеров с различными производительностями процессоров и показано, что он гарантирует для любого потока задач построение расписания, отличающегося от оптимального не более, чем в 2e раз. Этот результат является обобщением результатов, полученных для распределения параллельных задач на группе кластеров с одинаковыми процессорами [3].

## *2. Обозначения и полученные результаты*

Для того чтобы определить, насколько хорошим является предложенный приближенный алгоритм, используем следующие понятия и обозначения.

Пусть $T$ – некоторая последовательность мультипроцессорных задач, $C$ – множество кластеров: $Opt(T,C)$, и $A(T,C)$ – оптимальное время выполнения и время выполнения для расписания (соответственно), полученного при помощи алгоритма $A$. Далее предположим, что требуемое нормализованное время ограничено сверху некоторой константой $h_{max}$. Тогда абсолютная точность может быть вычислена по следующей формуле:

$$R_A = \sup_{T,C} \left\{ A(T,C) \Big/ Opt(T,C) \right\}$$

а асимптотическая точность определяется следующим образом:

$$R_A^{\infty} = \lim_{k \to \infty} \sup_{T,C} \left\{ A(T,C) \Big/ Opt(T,C) \,\middle|\, Opt(T,C) \geq k \right\}$$

Особый интерес представляют так называемые онлайновые алгоритмы (on-line algorithms). Такие алгоритмы, получая задачи одну за другой, каждую задачу распределяют сразу, и не меняют в дальнейшем ее распределение. В онлайновых алгоритмах предполагается, что заранее не существует никакой

информации об остальных задачах и последовательности их поступления. Обычно онлайновые алгоритмы более удобны для использования на практике, так как далеко не всегда с самого начала известны все размещаемые объекты. Однако достаточно сложно получить онлайновый алгоритм с хорошим качеством аппроксимации.

В данной работе получены следующие результаты:

- Показано (Теорема 1), что любой онлайновый алгоритм, распределяющий многопроцессорные задания на группе кластеров с произвольными производительностями процессоров не может иметь асимптотическую точность, меньшую чем *e* (здесь *e* – основание натурального логарифма).
- Предложен полиномиальный онлайновый алгоритм с асимптотической точностью близкой к *2e* (Теорема 3).

Введем некоторые обозначения. Пусть $T=\{T_1,\ldots,T_n\}$ - конечная последовательность задач, $h(T_j)$ и $w(T_j)$ - требуемое нормализованное время и требуемое число процессоров для *j*-й задачи. Требуемое нормализованное время ограничено сверху величиной $h_{max}$.

Пусть $C=\{C_1,\ldots,C_m\}$ - множество кластеров, $w_i$ - число процессоров в кластере, $\alpha_i$ - производительность кластера (это означает, что кластер может обработать задачу с требуемым нормализованным временем *h* за $\frac{h}{\alpha_i}$ ).

Предположим, что производительности ограничены снизу некоторой константой $\alpha_{\min}$ .

**Задача**: Найти распределение для заданных мультипроцессорных задач *T* на заданном множестве кластеров *C*, минимизируя общее время выполнения.

Ниже будем предполагать, что $w_1\geq w_2\geq\ldots\geq w_m$ .

Обозначим $\tilde{w}_i = w_i \cdot \alpha_i$ (*i*=1,…,*m*) общую производительность некоторого кластера.

Предположим, что существуют две последовательности задач $T^1=\{T_1^1,\ldots,T_n^1\}$ и $T^2=\{T_1^2,\ldots,T_n^2\}$ . Обозначим через $T^1+T^2$ объединенную последовательность:

$$T^1+T^2=\{T_1^1,\ldots,T_n^1,T_1^2,\ldots,T_n^2\}$$

Предположим, R – некоторая многопроцессорная задача. Обозначим:



$$last(R)=\max k \ : \ w_k\geq w(R)$$

Теперь можно разбить все задачи по классам $M_1,\ldots,M_m$ следующим образом:

$$R\in M_i \Leftrightarrow last(R)=i$$

Предположим, что $S(R)=h(r)\cdot w(R)$ - общий объем вычислений, требуемый для некоторой задачи *R*. Если *Q* – некоторое множество задач, тогда через $S(Q)$ обозначим общий объем вычислений, требуемый для всех задач из множества *Q*.

Предположим, что *T* – последовательность задач, тогда обозначим

$$S_i(T)=S(\{R\in T \mid R\in M\})$$

Предположим, что имеется некоторый алгоритм *A*, который осуществляет распределение задач по кластерам. Обозначим через *A(T)* вектор, содержащий *m* значений $y=(y_1,\ldots y_m)$ , где $y_k$ - общий объем требуемых вычислений для задач из последовательности *T*, которые распределены алгоритмом *A* на кластер с номером *k*.

Введем следующие обозначения:

$$d_i = \sum_{j=1}^{i} \tilde{w}_i,$$

$$D_i(T) = \sum_{j=1}^{i} S_j(T),$$

$$h(T) = \max_i \frac{D_i(T)}{d_i}.$$

Когда добавляется новая задача, $D_i(T)$ может только возрасти. Следовательно, $h(T)$ также может только возрасти. Приведем сейчас достаточно простую нижнюю оценку.

**Лемма 1.** Для любого множества кластеров *C* и любой последовательности многопроцессорных задач *T*

$$Opt(T,C)\geq h(T)$$

Достаточно простым обобщением нижней оценки из [3] является следующая

**Теорема 1.** Для любого онлайнового алгоритма *A* выполняется следующее неравенство

$$R_A^{\infty} \geq e$$

Однако основная трудность состоит в получении верхней оценки асимптотической точности для некоторого онлайнового алгоритма. Эта задача будет рассмотрена в следующем разделе.

## *3. Алгоритм с асимптотической точностью 2e.*

Для задачи составления расписания внутри одного кластера используем так называемый «шельфовый» алгоритм. Он был подробно представлен в работах по упаковке прямоугольников в полубесконечную полосу.

Предположим, имеется некоторая константа $r \in (0,1)$ Шельфом будем называть конкретный интервал времени, в течение которого задача может быть выполнена на данном кластере. Когда приходит новая задача, алгоритм определяет целое число *k*, такое что $r^{k+1} < h(R) \leq r^k$ и далее эта задача помещается в шельф длиной $r^k$ (под длиной шельфа понимается длина интервала, соответствующего данному шельфу в терминах нормализованного времени). Задача может быть помещена в один из существующих шельфов (если в нем существует достаточное число свободных процессоров) или может быть создан новый шельф. Внутри каждого шельфа задачи располагаются последовательно горизонтально (каждая новая задача занимает требуемое число процессоров на весь период времени шельфа).

Таким образом распределение многопроцессорных задач, которые попадают в один класс (каждой из этих задач соответствует одно значение *k*) выполняется при помощи некоторой эвристики, которая представляет собой одномерную упаковку в контейнеры (one-dimensional bin-packing). Будем использовать эвристику первого подходящего (First Fit), которая помещает задачу в первый шельф, который подходит или создает новый шельф, если подходящего шельфа не оказалось.

Описанный выше алгоритм будем называть *Shelf(r)*.

### 3.1. Алгоритм $A_r$.

Алгоритм распределяет многопроцессорные задачи в режиме онлайн. Предположим, что последовательность задач *T* уже распределена и предположим, что

$$A_r(T) = y = (y_1, \ldots . y_k)$$



Каждая новая задача *R* обрабатывается алгоритмом следующим образом:

1. Вычисляется $h = h(T + |R|)$.
2. Определяется номер кластера

$$k = \max_{w(R) < w_i} i \text{ и } \frac{y_i}{\tilde{w}_i} \leq eh$$

Если такое число *k* существует, то задача помещается на кластер *k*, в противном случае алгоритм останавливается. В последнем случае остается некоторое число нераспределенных задач (что означает, что алгоритм работает некорректно).

3. Для распределения задач внутри кластеров используется алгоритм *Shelf(r)*.

**Теорема 2**. Для любого множества кластеров *C* и любой последовательности многопроцессорных задач *T* алгоритм $A_r$ распределяет все задания из *T*.

**Теорема 3**. Для алгоритма $A_r$ выполняется следующее неравенство

$$R_{A_r}^{\infty} \leq 2e/r$$

Для доказательства данной теоремы необходимо следующее свойство шельфового алгоритма.

**Лемма 2**. Рассмотрим некоторый кластер. Предположим, *S* – общий объем вычислений требуемый для выполнения всех задач, за исключением последнего, которое помещено в кластер. Пусть *w* – число процессоров, $\alpha$ - производительность кластера, $h_{max}$ – наибольшее из возможных нормализованное требуемое время для некоторой задачи. Тогда для времени выполнения на этом кластере, полученном в соответствии с алгоритмом распределения *Shelf(r)*, выполняется следующее неравенство:

$$Shelf(r) \leq \frac{2}{r} \cdot \frac{S}{w\alpha} + \frac{h_{\max}}{\alpha} \left( \frac{1}{r(1-r)} + 1 \right)$$

**Лемма 3**. Для любого множества кластеров *C* и любой последовательности многопроцессорных задач *T* выполняется следующее неравенство:

$$A_r(T,C) \leq \frac{2e}{r} \cdot Opt(T,C) + \left( \frac{1}{r(1-r)} + 1 \right) \cdot \frac{h_{\max}}{\alpha_{\min}}$$

**Доказательство.** Предположим, максимальное время выполнения достигается на кластере с номером *j*. Пусть *R* – последняя задача, распределенная на этот кластер, и предположим, что общий объем вычислений, требуемых для всех остальных задач - *S*. Поскольку задача распределена на данный кластер

$$\frac{S}{\hat{w}_j} \le eh(T) \le e \cdot Opt(T,C)$$

Используя лемму 2 для кластера с номером *j*, получаем:

$$A_r(T,C) \le \frac{2}{r} \cdot \frac{S}{\alpha_j w_j} + \left( \frac{1}{r(1-r)} + 1 \right) \frac{h_{\max}}{\alpha_j} \le \frac{2e}{r} \cdot Opt(T,C) + \left( \frac{1}{r(1-r)} + 1 \right) \cdot \frac{h_{\max}}{\alpha_{\min}}$$

Лемма 3 доказана. Теорема 3 прямо следует из леммы 3.

**Список литературы**


[1] Bernam P., Charikar M., Karpinski M., On-line load balancing for related machines, Proc. WADS, 1997, LNCS, 1272, Springer-Ferlag, p. 116-125.

[2] Aspens J., Azar Y., Fiat A., Plotkin S., Waarts O., On-line load balancing with applications to machine scheduling and virtual circuit routing, Proc. 25th ACM STOC, 1993, p. 623-631.

[3] Жук С.Н., Приближенные онлайновые алгоритмы упаковки прямоугольников в несколько полос, Дискретная математика, 2007, т. 19, N 4.


## On-line algorithm for scheduling parallel tasks on a group of related clusters


*Zhuk S.N.,*
*Princeton University (szhuk@princeton.edu)*



**Abstract:** An on-line algorithm for scheduling parallel tasks on a group of clusters with different speeds of processors is presented. It is shown that for arbitrary set of parallel tasks the obtained schedule is within 2e of optimal.



**Keywords:** parallel computations, load balancing, scheduling parallel tasks on related clusters.