

**МОДЕЛИРОВАНИЕ
В ФИЗИКО-ТЕХНИЧЕСКИХ ИССЛЕДОВАНИЯХ**

УДК 681.324

В. Г. Хорошевский, М. С. Седельников

(Новосибирск)

**ЭВРИСТИЧЕСКИЕ АЛГОРИТМЫ
РАСПРЕДЕЛЕНИЯ ЗАДАЧ
ПО МАШИНАМ ВЫЧИСЛИТЕЛЬНОЙ СИСТЕМЫ***

Рассматривается функционирование распределенной вычислительной системы в режиме обработки набора параллельных задач. Предлагается модификация одного последовательного стохастического алгоритма и ее параллельная реализация. Приводятся результаты моделирования, подтверждающие, что полученные алгоритмы обеспечивают, по крайней мере, субминимальное время решения набора задач.

Современный инструментарий индустрии обработки информации – это распределенные большемасштабные вычислительные системы (ВС) высокой производительности. Архитектура распределенных ВС представляется в виде композиции множества элементарных машин (ЭМ) или процессоров и сети связей между ними. В таких системах все основные ресурсы являются и логически и технически распределенными. Число процессоров в большемасштабных распределенных ВС может быть достаточно большим и достигать 10^6 . Например, вычислительная система NEC Earth-Simulator имеет в своем составе 5120 процессоров, а создаваемая система IBM Blue Gene будет иметь 10^6 процессоров.

Распределенная ВС может быть использована для обслуживания потоков задач, представленных параллельными программами со случайными параметрами (числом ветвей, временем решения и т. п.). Интенсивность потоков такова, что на входе ВС имеется конечная очередь, и, следовательно, возникает необходимость в распределении набора задач. Для решения этой проблемы могут быть использованы эвристические и стохастические методы [1]. В данной работе рассматривается один из таких методов, предлагаются его последовательная и параллельная модификации.

* Работа выполнена при поддержке Российского фонда фундаментальных исследований (гранты № 02-07-90379, № 02-07-90380).

1. Постановка задачи. Имеются множество $\mathcal{X} = \{J_1, J_2, \dots, J_n\}$ одинаковых элементарных машин, образующих распределенную ВС, и множество $\mathcal{T} = \{I_1, I_2, \dots, I_L\}$ независимых задач. Задача $I_i \in \mathcal{T}$ характеризуется рангом $1 \leq r_i \leq n$ (числом параллельных ветвей в ее программе) и временем решения t_i . Иначе говоря, для решения задачи $I_i \in \mathcal{T}$ требуется r_i машин и t_i единиц времени. Необходимо построить алгоритмы распределения задач по ЭМ системы, которые обеспечивали бы минимумы целевых функций, характеризующих эффективность ВС. В качестве целевой функции используется общее время решения задач T на ВС.

Эта задача является труднорешаемой [2]. Применение методов математического программирования требует больших вычислительных ресурсов [1], поэтому в представленной работе предлагаются эвристические подходы. Построенные алгоритмы эффективно реализуются и на ВС и ЭВМ и гарантируют, по крайней мере, субминимум целевой функции.

Алгоритмы описаны в виде операторных схем, содержащих последовательность операторов, каждый из которых представляет достаточно крупную группу элементарных операций. При этом используются в основном обозначения, принятые в [1, 3]: A -, Π -, P -, \mathcal{G} -, F -, Φ -, S -, T -, $Я$ -операторы.

A -операторы относятся к классу арифметических. Каждый A -оператор представляет собой совокупность операций, реализующих какое-нибудь соотношение или систему соотношений между величинами. Запись A_j^i означает, что от арифметического оператора с номером j (т. е. от A_j) управление передается оператору (любого из перечисленных выше классов) с номером i .

Π -операторы осуществляют передачу управления. Запись $\Pi_j := A_i$ (или Π_j^i) означает, что при реализации оператора Π осуществляется переход к оператору A_i .

P -операторы относятся к логическим. Например, запись $P_j: P\{x\}$, $P=1 \rightarrow A_j$, $P=0 \rightarrow P_k$ означает проверку выполнения условия x ; если окажется, что условие выполнено ($P=1$), то осуществить переход к арифметическому оператору A_j , а если же условие не выполнено ($P=0$), то перейти к логическому оператору P_k , иначе $P_j \uparrow \downarrow k$.

Обозначение передачи управления от одного оператора к другому, непосредственно за ним следующему, опускается. Слева вверху от символа данного оператора ставятся номера тех операторов, от которых передается управление. Например, ${}^{i,j}A_k$ означает, что оператору A_k управление передается от операторов с номерами i или j .

\mathcal{G} -операторы относятся к классу операторов присваивания значения выражения переменной. Например, запись $\mathcal{G}_j: T := T + t$ означает, что при выполнении оператора \mathcal{G}_j переменной T присваивается значение суммы $T + t$.

F -операторы служат для формирования подмножеств и последовательностей задач.

Φ -операторы предназначены для формирования последовательностей как реализаций случайных процессов.

S -операторы предназначены для сортировки последовательностей по определенному признаку.

T -операторы служат для передачи и приема данных между ветвями параллельного алгоритма.

$Я$ -операторы означают конец алгоритма.

2. Стохастический алгоритм распределения набора задач. Рассмотрим алгоритм распределения набора задач, использующий метод цепей Монте-Карло [1]. Для краткости будем называть его алгоритмом МС (Monte-Carlo). Общая схема работы этого алгоритма состоит из двух частей. В первой части множество задач различных рангов с различным временем решения преобразуется во множество укрупненных задач также разных рангов, но с одинаковым (заранее заданным) временем решения Θ . Другими словами, множество всех задач ранга r , $1 \leq r \leq n$, разбивается на подмножества, в каждое из которых входят задачи, суммарное время решения которых близко к Θ . Во второй части алгоритма множество всех укрупненных задач также разбивается на подмножества, так что суммарный ранг всех укрупненных задач в каждом подмножестве максимально близок к n . В результате получается распределение набора задач по ВС с субминимальным суммарным временем решения.

Опишем основные операции первой части алгоритма МС. Пусть подмножества $\mathfrak{Z}^r \subseteq \mathfrak{Z}$, $r=1, n$, построены так, что в \mathfrak{Z}^r входят все задачи $I_i^r \in \mathfrak{Z}$, $i=1, 2, \dots, a$, которые имеют ранг r . Время решения этих задач

$$T^r = \sum_{i=1}^n t_i^r.$$

Пусть также

$$\widehat{\mathfrak{Z}} = \{I_1^r, I_2^r, \dots, I_i^r, \dots, I_a^r\}$$

– некоторая последовательность, членами которой являются элементы $I_i^r \in \mathfrak{Z}$. Подмножество $\mathfrak{Z}_j \subseteq \widehat{\mathfrak{Z}}$ включает в себя k_j задач, причем

$$\mathfrak{Z}_j = \bigcup_{i=K_j+1}^{K_j+k_j} I_i^r,$$

где

$$K_j = \sum_{s=0}^{j-1} k_s, \quad j=1, \dots, L_r, \quad k_0 = 0, \quad L_r = \lceil T^r / \Theta \rceil$$

(L_r – ближайшее к T^r / Θ целое число, $L_r \geq T^r / \Theta$). Каждое подмножество $\mathfrak{Z}_j \subseteq \widehat{\mathfrak{Z}}$, $j=1, 2, \dots, L_r$, назовем укрупненной задачей I_j^r ранга r . Время решения таких задач

$$T_j = \sum_{i=K_j+1}^{K_j+k_j} t_i^r.$$

Будем считать, что пакет укрупненных задач ранга r сформирован, если справедливо равенство

$$\left| \widehat{T} - \Theta \right| = o(\widehat{T}), \quad (1)$$

где

$$\hat{T} = \max_{\mathfrak{J}_j \subseteq \hat{\mathfrak{Z}}} \{T_j\};$$

$o(\hat{T})$ – бесконечно малая более высокого порядка малости, чем \hat{T} . Этому требованию можно удовлетворить, если за единицу времени взять величину Θ такую, что $\Theta \gg t_i^r$ для каждой задачи $I_i^r \in \mathfrak{Z}^r$.

Подмножества $\mathfrak{Z}_j \subseteq \hat{\mathfrak{Z}}$ выбираются следующим образом. Пусть построенные подмножества $\mathfrak{Z}_s \subseteq \hat{\mathfrak{Z}}$, $s=1,2,\dots,j-1$. Пусть также $\hat{\mathfrak{Z}}_j'$ – часть последовательности $\hat{\mathfrak{Z}}$ такая, что

$$\hat{\mathfrak{Z}}_j' = \{I_{K_j+1}, I_{K_j+2}, \dots, I_{K_j+k_j'}\}, \quad T^j = \sum_{i=K_j+1}^a t_i^r,$$

а для величин $(\Theta_v)_j$ и $(\Theta_n)_j$ справедливо

$$\sum_{i=K_j+1}^{K_j+k_j'} t_i^r = (\Theta_n)_j \leq \Theta < (\Theta_v)_j = \sum_{i=K_j+1}^{K_j+k_j'+1} t_i^r.$$

Тогда подмножества

$$\mathfrak{Z}_j := \begin{cases} \hat{\mathfrak{Z}}_j', & k_j := k_j', \quad \text{если } (\Theta_v)_j > [T^j - (\Theta_n)_j](L_r - j)^{-1}; \\ \hat{\mathfrak{Z}}_j' \cup I_{K_j+k_j'+1}, & k_j := k_j' + 1, \quad \text{иначе.} \end{cases}$$

Требуется выбрать такую последовательность $\hat{\mathfrak{Z}}^*$ задач, которая обеспечит выполнение условия (1).

Последовательность $\hat{\mathfrak{Z}}^*$ отыскивается с помощью метода цепей Монте-Карло [4]. Для этого последовательность $\hat{\mathfrak{Z}}$ принимают за базовую. Затем рассматривают перестановки, находящиеся на расстоянии не больше k , $k \leq a$, от базовой. В качестве расстояния между двумя последовательностями $\hat{\mathfrak{Z}}$ и $\hat{\mathfrak{Z}}'$ принимают число индексов в $\hat{\mathfrak{Z}}'$, которые не следуют за теми же индексами, что и в базовой $\hat{\mathfrak{Z}}$. Метод получения последовательности $\hat{\mathfrak{Z}}'$ с расстоянием не больше k основан на псевдослучайных числах.

Сначала получаем $k-1$ независимых переменных x_i в непрерывном интервале $(0, a)$: $x_i = a\xi_i$, $i=1,2,\dots,k-1$, где ξ_i – переменные в интервале $(0, 1)$, полученные генератором псевдослучайных чисел. Если $0 = x_0 \leq x_1 \leq \dots \leq x_i \leq \dots \leq x_k = a$, то x_i делят последовательность $\hat{\mathfrak{Z}}$ на k частей $\hat{\mathfrak{Z}}^s \subseteq \hat{\mathfrak{Z}}$, $s=1,2,\dots,k$, содержащих такие задачи, номера которых являются целыми числами в $(x_{i-1}, x_i]$. Некоторые части могут оказаться пустыми. Случайная перестановка этих частей дает новую последовательность $\hat{\mathfrak{Z}}'$ с расстоянием не больше k от базовой. Если \hat{T}' для новой последовательности $\hat{\mathfrak{Z}}'$ меньше \hat{T} , т. е. лучшим образом удовлетворяет (1), то $\hat{\mathfrak{Z}}'$ берется в качестве базовой.

Если сделано d попыток моделирования последовательностей с расстоянием k от базовой без ее изменения, то рассматриваются последовательности

с расстоянием $k/2$ и т. д. до тех пор, пока не будут смоделированы последовательности с расстоянием 2.

В результате работы первой части алгоритма МС мы имеем набор укрупненных задач $\mathfrak{Z}' = \{I'_i\}$, $i = \overline{1, L'}$, $L' = \sum_{r=1}^n L_r$. Каждая задача $I'_i \in \mathfrak{Z}'$, $i = \overline{1, L'}$,

характеризуется временем решения $t'_i = \Theta$ и имеет ранг $1 \leq r'_i \leq n$.

Опишем основные операции второй части алгоритма МС. Из элементов множества \mathfrak{Z}' строится базовая последовательность $\tilde{\mathfrak{Z}}$. Как в первой части, строятся подмножества $\mathfrak{Z}_j \subseteq \tilde{\mathfrak{Z}}$. Однако здесь в подмножество \mathfrak{Z}_j (в случае, когда уже построены подмножества $\mathfrak{Z}_s \subseteq \tilde{\mathfrak{Z}}$, $s = 1, 2, \dots, j-1$) включается k'_j задач, для которых справедливы неравенства

$$\sum_{i=K_j+1}^{K_j+k'_j} r_i \leq n, \quad \sum_{i=K_j+1}^{K_j+k'_j+1} r_i > n.$$

В последнее подмножество $\mathfrak{Z}_M \subseteq \tilde{\mathfrak{Z}}$ включается $k'_M = L' - K_{M-1}$ задач. Подмножество $\mathfrak{Z}_j \subseteq \tilde{\mathfrak{Z}}$ реализуется за время Θ на шаге j , число шагов равно M .

Очевидно, что нижние границы числа шагов и времени, которые требуются для реализации всей совокупности \mathfrak{Z}' укрупненных задач, соответственно равны

$$M^0 = \left\lceil n^{-1} \sum_{i=1}^{L'} r'_i \right\rceil, \quad \Theta M^0.$$

Необходимо найти последовательность $\tilde{\mathfrak{Z}}^*$, для которой число шагов M минимально, или $(M - M^0) = o(M)$. Следовательно, здесь в качестве целевой функции может быть взято не только время реализации задач множества \mathfrak{Z}' , но и число шагов M .

Последовательность $\tilde{\mathfrak{Z}}^*$ находится с помощью метода цепей Монте-Карло, аналогично первой части.

Введем операторы:

F_1 : формирование подмножеств $\mathfrak{Z}^r \subseteq \mathfrak{Z}$, $r = 1, 2, \dots, n$;

\mathfrak{G}_2 : $r := 1$; \mathfrak{G}_3 : $L_r :=]T^r / \Theta[$;

F_4 : формирование базовой последовательности $\tilde{\mathfrak{Z}}$;

F_5 : формирование подмножеств $\mathfrak{Z}_l \subseteq \tilde{\mathfrak{Z}}$, $l = 1, 2, \dots, L_r$;

A_6 : вычисление \hat{T} ; \mathfrak{G}_7 : $g := 1$;

Φ_8 : формирование последовательности $0 = x_0 \leq x_1 \leq \dots \leq x_k = a$;

F_9 : формирование частей $\tilde{\mathfrak{Z}}^s \subseteq \tilde{\mathfrak{Z}}$, $s = 1, 2, \dots, k$;

Φ_{10} : формирование последовательности $\hat{\mathfrak{Z}}$;

F_{11} : формирование подмножеств $\mathfrak{Z}'_l \subseteq \hat{\mathfrak{Z}}$, $l = 1, 2, \dots, L_r$;

A_{12} : вычисление \hat{T}' ;

P_{13} : $P\{\widehat{T}' < \widehat{T}\}$ – проверка выполнения условия $\widehat{T}' < \widehat{T}$, $P = 0 \rightarrow \mathfrak{D}_{17}$ – переход на \mathfrak{D}_{17} при невыполнении условия; \mathfrak{D}_{14} : $\widehat{T} := \widehat{T}'$;

F_{15} : формирование базовой последовательности $\widetilde{\mathfrak{Z}} := \widetilde{\mathfrak{Z}}'$, $\mathfrak{Z}_j := \mathfrak{Z}'_j$, $j = \overline{1, L_r}$; Π_{16} : $\rightarrow \mathfrak{D}_7$; \mathfrak{D}_{17} : $g := g + 1$; P_{18} : $P\{g > d\}$, $P = 0 \rightarrow \Phi_8$;

\mathfrak{D}_{19} : $k := \frac{1}{2}k$; P_{20} : $P\{k < 2\}$, $P = 0 \rightarrow \mathfrak{D}_7$; F_{21} : $I'_l := \mathfrak{Z}_l \in \mathfrak{Z}'$, $l = 1, 2, \dots, L_r$;

\mathfrak{D}_{22} : $r := r + 1$; P_{23} : $P\{r > n\}$, $P = 0 \rightarrow \mathfrak{D}_3$;

F_{24} : формирование базовой последовательности $\widetilde{\mathfrak{Z}}$ из укрупненных задач

I'_l , $l = 1, 2, \dots, L'$, $L' = \sum_{r=1}^n L_r$;

F_{25} : формирование подмножеств $\mathfrak{Z}_j \subseteq \widetilde{\mathfrak{Z}}$, $j = 1, 2, \dots, M$; \mathfrak{D}_{26} : $g := 1$;

Φ_{27} : формирование последовательности $0 = x_0 \leq x_1 \leq \dots \leq x_k = L'$;

F_{28} : формирование частей $\widetilde{\mathfrak{Z}}^s \subseteq \widetilde{\mathfrak{Z}}$, $s = 1, 2, \dots, k$;

Φ_{29} : формирование последовательности $\widetilde{\mathfrak{Z}}'$;

F_{30} : формирование подмножеств $\mathfrak{Z}'_l \subseteq \widetilde{\mathfrak{Z}}'$, $l = 1, 2, \dots, M'$;

P_{31} : $P\{M' < M\}$ – проверка выполнения условия $M' < M$, $P = 0 \rightarrow \mathfrak{D}_{35}$ – переход на \mathfrak{D}_{35} при невыполнении условия; \mathfrak{D}_{32} : $M := M'$;

F_{33} : формирование базовой последовательности $\widetilde{\mathfrak{Z}} := \widetilde{\mathfrak{Z}}'$, $\mathfrak{Z}_j := \mathfrak{Z}'_j$, $j = \overline{1, M}$;

Π_{34} : $\rightarrow \mathfrak{D}_{26}$; \mathfrak{D}_{35} : $g := g + 1$; P_{36} : $P\{g > d\}$, $P = 0 \rightarrow \Phi_{27}$;

\mathfrak{D}_{37} : $k := \frac{1}{2}k$; P_{38} : $P\{k < 2\}$, $P = 0 \rightarrow \mathfrak{D}_{26}$;

\mathfrak{Y}_{39} : конец.

Операторная схема алгоритма МС имеет следующий вид:

$F_1 \mathfrak{D}_2 \overset{2, 23}{\mathfrak{D}_3} F_4 F_5 A_6 \overset{6, 16, 20}{\mathfrak{D}_7} \overset{7, 18}{\Phi_8} F_9 \Phi_{10} F_{11} A_{12} P_{13} \downarrow_{17} \mathfrak{D}_{14} F_{15} \Pi_{16} \overset{7, 13}{\mathfrak{D}_{17}} P_{18} \downarrow_8 \mathfrak{D}_{19}$

$P_{20} \downarrow_7 F_{21} \mathfrak{D}_{22} P_{23} \downarrow_3 F_{24} F_{25} \overset{25, 34, 38}{\mathfrak{D}_{26}} \overset{26, 36}{\Phi_{27}} F_{28} \Phi_{29} F_{30} P_{31} \downarrow_{35} \mathfrak{D}_{32} F_{33} \Pi_{34} \overset{26, 31}{\mathfrak{D}_{35}}$

$P_{36} \downarrow_{27} \mathfrak{D}_{37} P_{38} \downarrow_{26} \mathfrak{Y}_{39}$.

3. Модификация алгоритма МС (алгоритм РАСК). В модификации алгоритма МС вместо метода цепей Монте-Карло используется алгоритм решения известной комбинаторной задачи упаковки [2]. Далее модификацию алгоритма МС будем называть алгоритмом РАСК (packing – упаковка). Общая схема работы этого алгоритма совпадает со схемой алгоритма МС.

Алгоритм МС в процессе работы неявно решает задачу упаковки в контейнеры. Эта задача формулируется следующим образом: задано конечное множество $U = \{u_1, u_2, \dots, u_L\}$ предметов и размеры $s_i \in [0, 1]$ каждого предмета $u_i \in U$ (размер предмета представлен рациональным числом). Требуется найти такое разбиение множества U на непересекающиеся подмножества U_1, U_2, \dots, U_k , чтобы сумма размеров предметов в каждом подмножестве U_i не превосходила 1 и чтобы k было наименьшим возможным. Иначе говоря,

предметы, принадлежащие каждому множеству U_i , упаковываются в один контейнер размера 1, а цель – упаковать предметы множества U в как можно меньшее число контейнеров.

Для решения задачи упаковки используется алгоритм «первый подходящий в порядке убывания» [2]. Далее будем обозначать его FFD (First Fit Decreasing – первый в порядке убывания). Пусть задана последовательность предметов, упорядоченных по мере убывания их размеров. Предметы помещаются в контейнеры в порядке возрастания номеров. Очередной предмет u_i помещается в контейнер с наименьшим номером, в который он может попасть, не нарушая ограничения по размеру контейнера. Для этого алгоритма существует оценка точности получаемого решения. Пусть $FFD(I)$ – количество контейнеров, которое было получено в результате решения алгоритмом некоторой задачи I . $OPT(I)$ – точное решение этой задачи. Справедлива следующая

Теорема [2]:

1. Для всех задач упаковки I выполняется неравенство

$$FFD(I) \leq \frac{11}{9} OPT(I) + 4.$$

2. Существуют задачи I , для которых $OPT(I)$ произвольно велико и

$$\frac{11}{9} OPT(I) \leq FFD(I).$$

Таким образом, имеем гарантию, что алгоритм FFD даже в худшем случае выдает решение, отличающееся от оптимального не более, чем на 22 %, и эта оценка точная, так как существуют задачи, при решении которых она достигается. Следует заметить, что задача упаковки в контейнеры является NP-трудной в сильном смысле [2], т. е. существуют веские основания для отказа от поиска для нее точного алгоритма с полиномиальной трудоемкостью.

Опишем алгоритм РАСК. Сначала алгоритм упаковывает задачи, имеющие размер t_i , в контейнеры размера Θ , затем – укрупненные (упакованные) задачи размера r_i в контейнеры размера n (n – число элементарных машин ВС, Θ – некоторая заранее заданная величина). Таким образом, используя алгоритм, решающий задачу упаковки в контейнеры, мы решаем задачу распределения конечного набора задач по ЭМ вычислительной системы.

Рассмотрим основные операции первой части алгоритма РАСК. Пусть подмножества $\mathfrak{I}^r \subseteq \mathfrak{I}$, $r = 1, n$, построены так, что в \mathfrak{I}^r входят все задачи $I_i^r \in \mathfrak{I}$, $i = 1, 2, \dots, a$, которые имеют ранг r . Пусть также

$$\widehat{\mathfrak{I}} = \{I_1^r, I_2^r, \dots, I_i^r, \dots, I_a^r\}$$

– последовательность, упорядоченная по невозрастанию времени решения t_i^r , членами которой являются элементы $I_i^r \in \mathfrak{I}$. Подмножество $\mathfrak{I}_j \subseteq \widehat{\mathfrak{I}}$ включает в себя k_j задач, причем

$$\mathfrak{I}_j = \bigcup_{i=K_j+1}^{K_j+k_j} I_i^r,$$

где

$$K_j = \sum_{r=0}^{j-1} k_r, \quad k_0 = 0.$$

Каждое подмножество $\mathfrak{Z}_j \subseteq \widehat{\mathfrak{Z}}$ назовем укрупненной задачей Ψ_j ранга r .
Время решения таких задач

$$T_j = \sum_{i=K_j+1}^{K_j+k_j} t'_i.$$

Подмножества $\mathfrak{Z}_j \subseteq \widehat{\mathfrak{Z}}$ формируются по следующему правилу.

Правило 1. Очередная задача I'_j из последовательности $\widehat{\mathfrak{Z}}$ включается в \mathfrak{Z}_j с минимальным номером j таким, что $\Theta - T_j \geq t_j$.

В результате работы первой части алгоритма РАСК мы имеем набор укрупненных задач $\mathfrak{Z}' = \{I'_i\}$, $i = \overline{1, L'}$, $L' = \sum_{r=1}^n L_r$. Каждая задача $I'_i \in \mathfrak{Z}'$,

$i = \overline{1, L'}$, характеризуется временем решения $t'_i = \Theta$ и имеет ранг $1 \leq r'_i \leq n$.

Опишем основные операции второй части алгоритма РАСК. Из элементов множества \mathfrak{Z}' строится последовательность $\widetilde{\mathfrak{Z}}$, упорядоченная по невозрастанию рангов r'_i . Строятся подмножества $\mathfrak{Z}_j \subseteq \widetilde{\mathfrak{Z}}$ по следующему правилу.

Правило 2. Очередная задача I'_j из последовательности $\widetilde{\mathfrak{Z}}$ включается в \mathfrak{Z}_j с минимальным номером j таким, что $n - \sum_{i=K_j+1}^{K_j+k'_j} r_i \geq r_j \left(k'_j - \text{число задач во множестве } \mathfrak{Z}_j, K_j = \sum_{r=0}^{j-1} k'_r, k'_0 = 0 \right)$.

В результате работы алгоритма определяется максимальный индекс $M = \max_{\mathfrak{Z}_j} j$ подмножеств \mathfrak{Z}_j . Подмножество $\mathfrak{Z}_j \subseteq \widetilde{\mathfrak{Z}}$ реализуется за время Θ , число подмножеств – M , общее время выполнения набора задач на ВС равно ΘM .

Введем операторы:

F_1 : формирование подмножеств $\mathfrak{Z}^r \subseteq \mathfrak{Z}$, $r = 1, 2, \dots, n$;

\mathfrak{Z}_2 : $r := 1$; F_3 : формирование последовательности задач $\widetilde{\mathfrak{Z}}$;

S_4 : сортировка последовательности задач $\widetilde{\mathfrak{Z}}$ по невозрастанию времени решения;

F_5 : формирование подмножеств $\mathfrak{Z}_l \subseteq \widetilde{\mathfrak{Z}}$ по правилу 1;

F_6 : $I'_l := \mathfrak{Z}_l \in \mathfrak{Z}'$; \mathfrak{Z}_7 : $r := r + 1$; P_8 : $P\{r > n\}$, $P = 0 \rightarrow F_3$;

F_9 : формирование последовательности задач $\widetilde{\mathfrak{Z}}$;

S_{10} : сортировка последовательности задач $\widetilde{\mathfrak{Z}}$ по невозрастанию ранга;

F_{11} : формирование подмножеств $\mathfrak{I}_j \subseteq \tilde{\mathfrak{I}}$ по правилу 2, $j = 1, 2, \dots, M$;

$Я_{12}$: конец.

Операторная схема алгоритма РАСК имеет следующий вид:

$$F_1 \mathfrak{D}_2^{2,8} F_3 S_4 F_5 F_6 \mathfrak{D}_7 P_{8 \downarrow 3} F_9 S_{10} F_{11} Я_{12}.$$

4. Параллельный алгоритм РРАСК. Распараллеливание алгоритма РАСК идет по операторам S_4 и F_5 , которые реализуются одновременно на различных ЭМ для различных рангов r . Параллельную реализацию алгоритма РАСК далее будем называть РРАСК (parallel – параллельный).

Опишем основные операции алгоритма РРАСК. Пусть N – число ветвей алгоритма. Множество \mathfrak{I} разбивается на N подмножеств G^k , так что выполняется условие: либо задачи из подмножеств G^k и G^s имеют различные ранги для всех $k \neq s$, либо если существуют подмножества, содержащие задачи одного ранга, то, по крайней мере, одно из них полностью состоит из задач этого ранга. Задачи каждого ранга сортируются параллельным алгоритмом сортировки PSort [5] по неубыванию времени решения. Для подмножеств задач каждого ранга параллельно реализуется процедура формирования укрупненных задач со временем выполнения не больше заданной величины Θ (см. разд. 3, первая часть алгоритма РАСК). Получившиеся укрупненные задачи пересылаются на одну ЭМ, на которой реализуется процедура упаковки укрупненных задач в контейнеры размера n (n – число ЭМ). В результате определяется максимальное число контейнеров M и время реализации всей совокупности задач ΘM .

Пусть первый индекс оператора показывает номер ветви, которую он реализует, второй индекс – его порядковый номер.

Введем операторы:

A_{11} : формирование подмножеств G^k , $k = 1, \dots, n$;

T_{12} : передача подмножества G^k в ветвь k , $k = 2, \dots, n$;

T_{i2} ($i = 2, \dots, n$): прием подмножества G^i от ветви 1;

S_{i3} ($i = 1, \dots, n$): параллельная сортировка подмножества G^i алгоритмом PSort по невозрастанию ранга;

F_{i4} ($i = 1, \dots, n'$): формирование подмножеств $\mathfrak{I}_l^s \subseteq \tilde{\mathfrak{I}}_l^s$ по правилу 1, $l = 1, \dots, L'$; $\tilde{\mathfrak{I}}_l^s$ – отсортированная последовательность задач ранга s , n_r – общее число рангов, $n' = \min \{n, n_r\}$ – число ветвей после использования алгоритма параллельной сортировки;

F_{i5} ($i = 1, \dots, n'$): $\Psi_l^s := \mathfrak{I}_l^s$ – укрупненная задача ранга s , $l = 1, \dots, L'$;

T_{i6} : прием укрупненных задач Ψ_l^s из ветвей k , $k = 2, \dots, n'$, $l = 1, \dots, L'$, $i = 1, \dots, n'$;

T_{i6} ($i = 2, \dots, n'$): передача укрупненных задач Ψ_l^s в ветвь 1, $l = 1, \dots, L'$;

F_{i7} : формирование из задач Ψ_l^s последовательности $\tilde{\mathfrak{I}}_l$, $l = 1, \dots, L'$, $i = 1, \dots, n'$;

F_{i8} : формирование подмножеств $\mathfrak{I}_j \subseteq \tilde{\mathfrak{I}}$ по правилу 2, $j = 1, 2, \dots, M$;

Я₁₉: конец.

Операторная схема алгоритма РРАСК имеет следующий вид:

$$\begin{array}{cccccccc}
 A_{11} & T_{12} & S_{13} & F_{14} & F_{15} & T_{16} & F_{17} & F_{18} & Я_{19} \\
 & & & & & & & & \\
 & T_{22} & S_{23} & F_{24} & F_{25} & T_{26} & & & \\
 & & \dots & \dots & \dots & \dots & \dots & & \\
 & & & & & & & & \\
 & T_{n2} & S_{n3} & F_{n'4} & F_{n'5} & T_{n'6} & & &
 \end{array}$$

5. Моделирование алгоритмов. При моделировании алгоритмов был использован вычислительный кластер Центра параллельных вычислительных технологий СибГУТИ [6]. Элементарная машина кластера имеет конфигурацию: процессор Intel Celeron 667, 128 Мбайт памяти, сеть связи Fast Ethernet 100 Мбит, Switch 3Com. Программы, реализующие алгоритмы, были написаны под операционной системой ASPLinux 7.3. В качестве средства написания параллельных программ использовались реализация стандарта MPI LAM 6.5.6 и C++. Рассматривались тестовые наборы задач со случайным рангом $r \in [1, 10]$ и со случайным временем решения $t \in [1, 100]$. Число задач в наборе изменялось в диапазоне $10^5 - 2 \cdot 10^6$. В качестве показателя эффективности рассматривалось время работы алгоритмов на ВС и получаемое ими время реализации набора задач. Число ЭМ n изменялось в пределах от 2 до 24. В алгоритме МС $d = k = 8, \Theta = 400$.

Моделирование проводилось в два этапа. На первом сравнивались алгоритмы МС и РАСК как по времени поиска решения, так и по величине получаемого времени реализации набора задач на ВС. Как видно из рис. 1, алгоритм РАСК распределяет набор задач эффективнее алгоритма МС на 20–30 %. Из рис. 2 видно, что время работы алгоритма РАСК меньше времени работы алгоритма МС в 1,5–7,5 раза в зависимости от числа задач в наборе, причем на меньших наборах РАСК работает существенно быстрее, чем МС.

На втором этапе вычислительного эксперимента проводилось исследование параллельного алгоритма РРАСК. Из рис. 3 следует, что параллельный алгоритм РРАСК работает эффективнее последовательного РАСК на достаточно больших пакетах задач ($L \geq 10^6$). При увеличении числа используемых

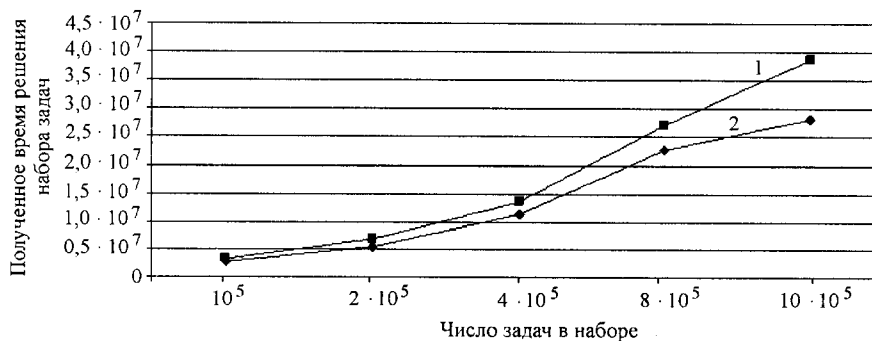


Рис. 1. Сравнение алгоритмов МС (кривая 1) и РАСК (кривая 2) по получаемому решению

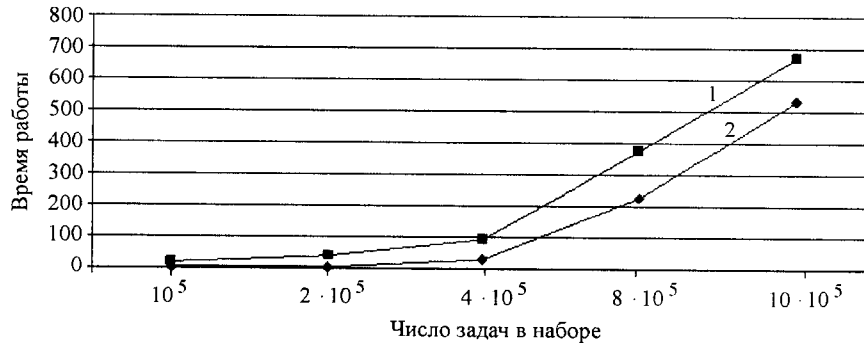


Рис. 2. Сравнение алгоритмов МС (кривая 1) и РАСК (кривая 2) по времени работы

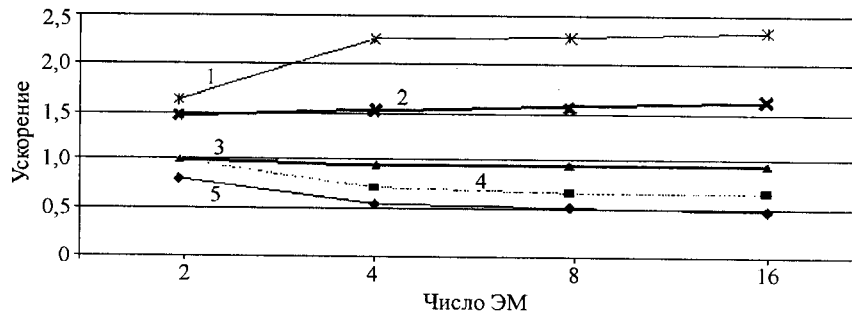


Рис. 3. Ускорение алгоритма РРАСК по сравнению с алгоритмом РАСК. Число задач в наборе: $10 \cdot 10^5$ (кривая 1), $8 \cdot 10^5$ (2), $4 \cdot 10^5$ (3), $2 \cdot 10^5$ (4), 10^5 (5)

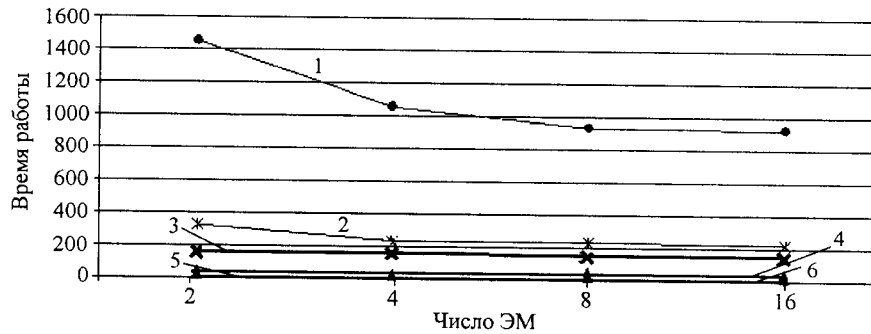


Рис. 4. Время работы алгоритма РРАСК при использовании различного числа ЭМ. Число задач в наборе: $20 \cdot 10^5$ (кривая 1), $10 \cdot 10^5$ (2), $8 \cdot 10^5$ (3), $4 \cdot 10^5$ (4), $2 \cdot 10^5$ (5), 10^5 (6)

ЭМ время работы алгоритма РРАСК сокращается незначительно из-за существенного роста времени, затрачиваемого на обмен информацией между параллельными ветвями (рис. 4).

Заключение. Сравнительный анализ алгоритмов позволяет рекомендовать последовательный алгоритм РАСК для применения на небольших ВС в случае, если число распределяемых задач не велико. Моделирование под-

твердило его преимущества перед алгоритмом МС как по времени поиска решения, так и по величине получаемого времени реализации набора задач.

Параллельный алгоритм РРАСК имеет смысл применять на большемасштабных ВС при распределении очень больших наборов задач, более 10^6 . Ограничения, накладываемые пропускной способностью сети связи, не позволяют эффективно использовать его при решении меньших наборов задач, что также было показано моделированием. Однако при распределении достаточно большого набора задач и при увеличении пропускной способности сети связи параллельный алгоритм РРАСК будет превосходить последовательные алгоритмы.

СПИСОК ЛИТЕРАТУРЫ

1. Евреинов Э. В., Хорошевский В. Г. Однородные вычислительные системы. Новосибирск: Наука, 1978.
2. Гэри М., Джонсон Д. Вычислительные машины и труднорешаемые задачи. М.: Мир, 1982.
3. Бусленко Н. П. Моделирование сложных систем. М.: Наука, 1968.
4. Page E. S. On Monte-Carlo methods in congestion on problems: Searching for an optimum in discrete situations // Operation Research. 1965. 13, N 2. P. 291.
5. Седельников М. С. Параллельный алгоритм сортировки // Матер. междунар. науч.-техн. конф. «Интеллектуальные и многопроцессорные системы-2003». Таганрог: Изд-во ТРТУ, 2003. Том. 2. С. 107.
6. Хорошевский В. Г., Майданов Ю. С., Мамоиленко С. Н. и др. Живучая кластерная вычислительная система // Тр. шк.-сем. «Распределенные кластерные вычисления». Красноярск: Изд-во Красноярского гос. ун-та, 2001.

*Институт физики полупроводников СО РАН,
Сибирский государственный университет
телекоммуникаций и информатики,
E-mail: khor@isp.nsc.ru*

*Поступила в редакцию
25 декабря 2003 г.*