РОССИЙСКАЯ АКАДЕМИЯ НАУК
СИБИРСКОЕ ОТДЕЛЕНИЕ

АВТОМЕТРИЯ

2008, том 44, № 2

УДК 681.324

АЛГОРИТМЫ РАСПРЕДЕЛЕНИЯ ВЕТВЕЙ ПАРАЛЛЕЛЬНЫХ ПРОГРАММ ПО ПРОЦЕССОРНЫМ ЯДРАМ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ*

В. Г. Хорошевский, М. Г. Курносов

Институт физики полупроводников им. А. В. Ржанова СО РАН, г. Новосибирск
E-mail: mkurnosov@isp.nsc.ru

Поставлена задача оптимального назначения ветвей параллельной программы на процессорные ядра распределенной вычислительной системы (ВС) с целью минимизации времени ее выполнения. В постановке задачи учтены иерархическая организация коммуникационной среды распределенной ВС, многоядерность процессоров и структура информационного графа параллельной программы. Предложены стохастические последовательный и параллельный алгоритмы решения задачи. Приведены результаты моделирования алгоритмов на кластерной ВС.

Введение. Одной из основных тенденций современной индустрии высокопроизводительной обработки информации является построение большемасштабных пространственно распределенных вычислительных и GRID-систем. В архитектурном плане распределенная вычислительная система (ВС) представляется как множество элементарных машин (ЭМ), взаимодействие между которыми осуществляется через коммуникационную среду [1, 2]. Основным функциональным элементом в таких системах является процессорное ядро, количество которых может варьироваться от десятков (в небольших кластерных системах) до сотен тысяч (например, в IBM BlueGene и Cray XT).

Коммуникационные среды современных распределенных ВС имеют иерархическую организацию, где каждый уровень иерархии характеризуется своими значениями показателей производительности. Например, в GRID-системах и пространственно распределенных мультикластерных ВС, образованных путем объединения высокопроизводительных кластеров и/или систем с массовым параллелизмом, 0-й уровень коммуникационной среды есть сеть связи между вычислительными кластерами, 1-й уровень – сеть связи внутри кластеров, 2-й уровень – среда доступа процессоров вычислитель-

* Работа выполнена при поддержке Российского фонда фундаментальных исследований (гранты № 08-07-00018, № 06-07-89089, № 08-07-00022, № 07-07-00142, № 08-08-00300) и Совета по грантам Президента РФ (грант № НШ- 2121.2008.9).

ного узла к общей памяти, 3-й уровень – средства доступа ядер процессора к сверхоперативной памяти (кэш-памяти).

Время выполнения параллельной программы на распределенной ВС существенно зависит от того, на какие процессорные ядра назначены ее ветви и через какие каналы связи они взаимодействуют. В этих условиях, при назначении ветвей параллельных программ на процессорные ядра распределенной ВС, необходимо учитывать и структуру информационного графа программы.

В работах [3–5], посвященных оптимизации назначения ветвей параллельных программ на процессорные ядра распределенных ВС, предполагается, что коммуникационная среда имеет одноуровневую организацию с определенной топологией сети межмашинных связей (например, в виде трехмерного тора, решетки) и не учитывается возможность взаимодействия процессоров одного вычислительного узла через его общую память, а ядер одного процессора через общую сверхоперативную память. Кроме того, в других публикациях в качестве показателей производительности каналов связи рассматривается лишь пропускная способность и не учитывается их латентность, хотя для многих параллельных программ учет этого показателя важен [6].

Данная работа посвящена оптимизации назначения ветвей параллельных программ на процессорные ядра распределенных ВС с учетом иерархической организации коммуникационной среды распределенных ВС и структуры информационного графа параллельной программы.

Постановка задачи. Пусть распределенная ВС состоит из H подсистем, объединенных каналами связи. Введем обозначения: $S = \{1, 2, \ldots, H\}$ – множество подсистем, образующих распределенную ВС; n_i – количество процессорных ядер, входящих в состав подсистемы $i \in S$; $N = \sum_{i=1}^{H} n_i$ – количество процессорных ядер, входящих в распределенную ВС; ω_i – производительность процессорного ядра i-й подсистемы ($[\omega_i] = \text{FLOPS}$, $i \in S$).

Коммуникационная среда i-й подсистемы допускает представление в виде дерева высотой h_i. Для каждого элемента на уровне $k \in \{1, 2, \ldots, h_i - 1\}$ задано количество n_{ik} его дочерних элементов. Дочерними элементами 0-го уровня являются вычислительные подсистемы. Далее считаем $n_{i0} = H$.

Каждый уровень $k \in \{1, 2, \ldots, h_i\}$ коммуникационной среды i-й подсистемы характеризуется своими значениями показателей производительности: $l_i^{(k)}(a, b, m)$ – зависимость латентности канала связи между элементами a и b на уровне k от размера m передаваемого сообщения ($a, b \in \{1, 2, \ldots, n_{ik}\}$, $[l_i^{(k)}(a, b, m)] = \text{с}$); $b_i^{(k)}(a, b, m)$ – зависимость пропускной способности канала связи между элементами a и b на уровне k от размера m передаваемого сообщения ($a, b \in \{1, 2, \ldots, n_{ik}\}$, $[b_i^{(k)}(a, b, m)] =$ бит/с). Аналогично для 0-го уровня коммуникационной среды заданы $l^{(0)}(a, b, m)$ и $b^{(0)}(a, b, m)$.

Обозначим через $C = \{c_1, c_2, \ldots, c_N\}$ множество процессорных ядер, входящих в распределенную ВС. Пусть g_r – номер подсистемы, которой принадлежит процессорное ядро с номером $r \in \{1, 2, \ldots, N\}$. Каждый элемент $c_r = (e_0^r, e_1^r, \ldots, e_{h_{g_r}-1}^r)$, $c_r \in C$, определяет координату размещения процессорного ядра в распределенной ВС (рис. 1). Через $e_j^r \in \{1, 2, \ldots, n_{g_r, j}\}$ обозначен номер дочернего элемента уровня j ($j = 0, 1, \ldots, h_{g_r} - 1$).

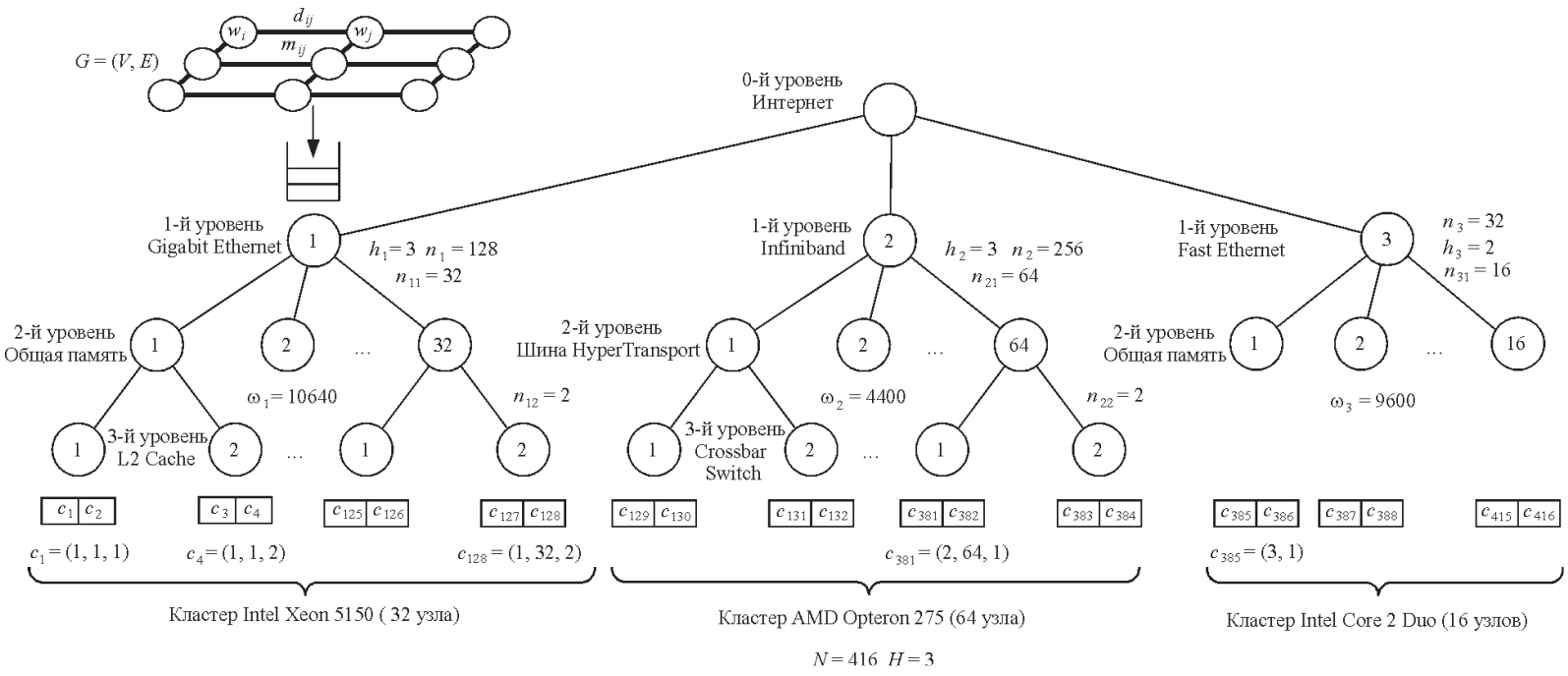

Рис. 1. Пример иерархической организации коммуникационной среды распределенной ВС

Пусть процедура назначения ветвей параллельной программы на процессорные ядра осуществляется с подсистемы $s \in S$ и поступившая на выполнение параллельная программа представлена ее информационным графом $G = (V, E)$, где $V = \{1, 2, \ldots, M\}$ – множество параллельных ветвей программы; $E \subset V \times V$ – множество информационно-логических связей между ее параллельными ветвями. Будем полагать: w_i – количество элементарных операций (арифметических и логических), выполняемых ветвью $i \in V$; d_{ij} – объем данных, передаваемый по ребру $(i, j) \in E$ за время выполнения параллельной программы ($[d_{ij}]$ = байт, $i, j \in V$); m_{ij} – средний размер сообщения, передаваемого по ребру $(i, j) \in E$ за одну операцию приема/передачи ($[m_{ij}]$ = = байт, $i, j \in V$); z – размер файла параллельной программы ($[z]$ = байт).

Задача назначения ветвей параллельной программы на процессорные ядра распределенной ВС заключается в отыскании инъективной функции $f : V \to C$, ставящей в соответствие ветвям параллельной программы процессорные ядра распределенной ВС. Искомая функция допускает представление в виде вектора

$$X = \{x_{ij} : i \in V,\ j \in C\}, \qquad x_{ij} = \begin{cases} 1, & \text{если } f(i) = j, \\ 0 & \text{иначе.} \end{cases}$$

Качество назначения ветвей параллельной программы на процессорные ядра будем оценивать ожидаемым временем t ее обслуживания, которое складывается из времени доставки программы на процессорные ядра и времени выполнения ветвей на них.

Время t' доставки программы на процессорные ядра распределенной ВС определяется максимальным из времен передачи файла программы до подсистем, процессорные ядра которых назначены для выполнения ее ветвей:

$$t' = \max_{i \in V} \{t'_i\} = \max_{i \in V} \left\{ \sum_{j=1}^{N} x_{ij}\, t^{(0)}(s, g_j, z) \right\},$$

где t'_i – время доставки программы с подсистемы $s \in S$ на подсистему, ядро которой выделено для выполнения параллельной ветви i. Величина

$$t^{(0)}(a, b, z) = l^{(0)}(a, b, z) + z \big/ b^{(0)}(a, b, z)$$

– время передачи сообщения размером z байт между двумя подсистемами через латентность и пропускную способность канала связи между ними.

Считаем, что после доставки файла программы на процессорные ядра осуществляется одновременный запуск всех параллельных ветвей на выполнение.

Время выполнения t'' параллельной программы определяется максимальным из времен выполнения ее ветвей. Время t''_i выполнения ветви $i \in V$ параллельной программы складывается из времени выполнения арифметических и логических операций процессорным ядром и времени взаимодействия со смежными ветвями:

$$t'' = \max_{i \in V} \{t''_i\} = \max_{i \in V} \{t_i^{\text{comp}} + t_i^{\text{comm}}\},$$

где

$$t_i^{\text{comp}} = \sum_{j=1}^{N} x_{ij} \frac{w_i}{\omega_{g_j}}$$

– время выполнения арифметических и логических операций процессорным ядром, на которое назначена ветвь;

$$t_i^{\text{comm}} = \sum_{j=1}^{M} \sum_{p=1}^{N} \sum_{q=1}^{N} x_{ip} x_{jq} t^{(k(p,q))}(i, j, p, q)$$

– время взаимодействия со смежными ветвями; $t^{(k)}(i, j, p, q)$ – время взаимодействия между ветвями $i, j \in V$, назначенными на процессорные ядра p и q соответственно ($p, q \in C$):

$$t^{(k)}(i, j, p, q) = \begin{cases} l^{(0)}(e_0^p, e_0^q, m_{ij}) d_{ij} \big/ m_{ij} + d_{ij} \big/ b^{(0)}(e_0^p, e_0^q, m_{ij}), & \text{если } k = 0, \\ l_{g_p}^{(k)}(e_k^p, e_k^q, m_{ij}) d_{ij} \big/ m_{ij} + d_{ij} \big/ b_{g_p}^{(k)}(e_k^p, e_k^q, m_{ij}) & \text{иначе.} \end{cases}$$

Функция $k(p, q)$ ставит в соответствие номерам процессорных ядер $p, q \in C$ номер уровня коммуникационной среды, являющегося ближайшим общим предком для них.

Учитывая инъективность функции $f : V \to C$, получаем задачу оптимального назначения ветвей параллельной программы на процессорные ядра распределенной ВС с целью минимизации времени ее выполнения:

$$\min_{(x_{ij})} \left\{ \max_{i \in V} \left\{ \sum_{j=1}^{N} x_{ij} t^{(0)}(s, g_j, z) \right\} + \right.$$

$$\left. + \max_{i \in V} \left\{ \sum_{j=1}^{N} x_{ij} \frac{w_i}{\omega_{g_j}} + \sum_{j=1}^{M} \sum_{p=1}^{N} \sum_{q=1}^{N} x_{ip} x_{jq} t^{(k(p,q))}(i, j, p, q) \right\} \right\}, \qquad (1)$$

при следующих ограничениях:

$$\sum_{j=1}^{N} x_{ij} = 1, \quad i = 1, 2, \ldots, M, \qquad (2)$$

$$\sum_{i=1}^{M} x_{ij} \le 1, \quad j = 1, 2, \ldots, N, \qquad (3)$$

$$x_{ij} \in \{0, 1\}, \quad i \in V, \; j \in C. \qquad (4)$$

Ограничения (2), (4) гарантируют назначение каждой ветви параллельной программы на единственное процессорное ядро, ограничения (3) обеспечивают назначение на ядро не более одной ветви.

Задача (1)–(4) представляет собой задачу дискретной оптимизации, мощность множества D допустимых решений которой оценивается величиной

$$\left|D\right| = A_N^M = N!/(N-M)!$$

Для большемасштабных распределенных ВС отыскание точного решения связано со значительной вычислительной сложностью. С учетом данного обстоятельства целесообразно применять приближенные методы решения задачи.

Метод решения. На основе метаэвристики «имитации отжига» [7, 8] разработан стохастический алгоритм, позволяющий отыскивать приближенные решения задачи за приемлемое время. Для краткости будем называть его PTA_SA (Parallel Task Assignment Simulated Annealing). Общая схема алгоритма такова. На первом шаге алгоритма строится начальное допустимое решение $x^{(0)} \in D$. На k-м шаге имеется текущее решение $x^{(k)} \in D$ и наилучшее на данный момент $x^* \in D$. Шаг начинается с выбора случайным образом соседнего решения $y \in N(x^{(k)})$ из окрестности текущего. Если значение целевой функции от соседнего решения меньше значения целевой функции от текущего, то соседнее решение принимается за текущее. Кроме того, даже если соседнее решение по целевой функции превосходит текущее, то соседнее решение принимается за текущее с вероятностью

$$P(x^{(k)}, y, k) = \begin{cases} 1, & \text{если } F(y) \le F(x), \\ \exp((F(x)-F(y))/c_k) & \text{иначе.} \end{cases}$$

Данный шаг позволяет алгоритму имитации отжига не «застревать» в локальных оптимумах и исследовать большее количество допустимых решений.

Опишем основные операции алгоритма PTA_SA. Представим решение задачи в виде вектора $x = (x_1, x_2, \ldots, x_M)$, где $x_i \in \{1, 2, \ldots, N\}$ – номер процессорного ядра, на которое назначена ветвь $i \in V$. Начальное решение $x^{(0)} \in D$ строится по следующему правилу.

Правило 1. Подсистемы распределенной ВС сортируются в порядке невозрастания количества процессорных ядер в них, после чего ветви параллельной программы последовательно назначаются на процессорные ядра, начиная с самой большой подсистемы.

Решение из окрестности текущего $y \in N(x)$ выбирается по следующему правилу.

Правило 2. Формируется равномерно распределенное псевдослучайное целое число $\xi_1 \in [0, N-1]$. Компоненты вектора решения $x = (x_1, x_2, \ldots, x_M)$ циклически сдвигаются на значение ξ_1:

$$x_i := \begin{cases} x_i + \xi_1, & \text{если } x_i + \xi_1 \le N, \\ (x_i + \xi_1) \bmod N & \text{иначе,} \end{cases} \quad i = 1, 2, \ldots, M.$$

После чего вектор решения в случайной позиции $\xi_2 \in [1, M-1]$ делится на две части: $[1, \xi_2]$ и $[\xi_2 + 1, M]$, которые переставляются местами.

Последовательность $\{c_k\}$ выбирается так, чтобы $c_k \to 0$ при $k \to \infty$. Это обеспечивает на начальных этапах работы алгоритма принятия с большей вероятностью решений со значениями целевой функции, большими текущего. В алгоритме PTA_SA последовательность $\{c_k\}$ строится следующим образом: $c_k = \alpha/(k+1)\beta$, где $\alpha = ((c_0 - c_R)(R+1))/R$, $\beta = c_0 - \alpha$, $R = \log_2 N$, $c_0 = \Delta F_{\max}$, $c_R = 0{,}1$ (R – номер последнего элемента последовательности, $\Delta F_{\max}$ – разность между верхней и нижней оценками времени выполнения параллельной программы при заданном назначении ветвей на процессорные ядра).

Алгоритмы описаны в виде операторных схем. Используются обозначения, принятые в [2]:

A – оператор, представляющий совокупность операций, которые реализуют систему соотношений между величинами. Запись A_j^i означает, что от арифметического оператора с номером j (т. е. от A_j^i) управление передается оператору с номером i.

П – операторы, осуществляющие передачу управления. Запись $\Pi_j :\to A_i$ (или Π_j^i) означает, что при реализации оператора П осуществляется переход к оператору A_i.

P – операторы, относящиеся к логическим. Например, запись P_i: $P\{x\}$, $P = 1 \to A_j$, $P = 0 \to P_k$, или $P_{j\downarrow k}^{\uparrow i}$, означает проверку выполнения условия x: если условие выполнено ($P = 1$), то осуществляется переход к арифметическому оператору A_j, если же условие не выполнено ($P = 0$), то переход осуществляется к логическому оператору P_k. Обозначение передачи управления от одного оператора к другому, непосредственно за ним следующему, опускается. Слева сверху от символа данного оператора ставятся номера тех операторов, от которых передается управление.

L – операторы присваивания значения выражения переменной.

F – операторы формирования начального решения и поиска минимального элемента в заданном множестве.

Ф – операторы, предназначенные для формирования решений из окрестности текущего решения и получения псевдослучайных чисел.

S, R, T – операторы, служащие для передачи и приема данных между ветвями параллельного алгоритма.

Я – операторы, которые означают конец алгоритма.

Введем операторы алгоритма PTA_SA:

F_1: формирование начального решения $x^{(0)} \in D$; L_2: $x^* := x^{(0)}$; L_3: $k := 0$;

A_4: вычисление $\Delta F_{\max}$; L_5: $c_0 := \Delta F_{\max}$;

P_6: $P\{c_k \geq c_R\}$, $P = 0 \to Я_{19}$ – переход на $Я_{19}$ при невыполнении условия; L_7: $d := 0$;

P_8: $P\{d \leq M\}$, $P = 0 \to L_{17}$ – переход на L_{17} при невыполнении условия;

$Ф_9$: выбор решения $y \in N(x^{(k)})$ из окрестности текущего;

$Ф_{10}$: формирование равномерно распределенного псевдослучайного числа $\xi \in [0, 1]$;

P_{11}: $P\{\xi < P(x^{(k)}, y, k)\}$, $P = 0 \to P_{13}$ – переход на P_{13} при невыполнении условия;

L_{12}: $x^{(k+1)} := y$;

P_{13}: $P\{F(y) \le F(x^*)\}$, $P = 0 \to L_{15}$ – переход на L_{15} при невыполнении условия;

L_{14}: $x^* := y$; L_{15}: $d := d + 1$; $\Pi_{16} :\to P_8$; L_{17}: $k := k + 1$; $\Pi_{18} :\to P_6$; $Я_{19}$: конец.

Операторная схема алгоритма PTA_SA имеет следующий вид:

$$F_1 L_2 L_3 A_4 L_5 {}^{18}P_{6\downarrow 19} L_7 {}^{16}P_{8\downarrow 17}$$

$$\Phi_9 \Phi_{10} P_{11\downarrow 13} L_{12} {}^{11}P_{13\downarrow 15} L_{14} {}^{13}L_{15} \Pi_{16}^{8} {}^{8}L_{17} \Pi_{18}^{6} Я_{19}.$$

Параллельный алгоритм PTA_PSA. Параллельный алгоритм разработан в соответствии с методикой крупноблочного распараллеливания [1, 2]. Итерации цикла $P_8 - \Pi_{16}$ алгоритма PTA_SA реализуются одновременно на разных процессорах. Параллельную версию алгоритма PTA_SA далее будем называть PTA_PSA (Parallel Task Assignment Parallel Simulated Annealing). Обозначим через n количество параллельных ветвей алгоритма. Выделим корневую ветвь (ветвь с номером 0). Рабочие ветви получают решение от корневой ветви и выполняют $M/(n-1)$ итераций цикла $P_8 - \Pi_{16}$, после чего отправляют корневой ветви значение целевой функции лучшего найденного решения. Корневая ветвь определяет наилучшее найденное решение, делает его текущим, и итерации цикла по значениям последовательности $\{c_k\}$ продолжаются.

Пусть первый индекс оператора обозначает номер параллельной ветви, которую он реализует, второй индекс – его порядковый номер. Обозначим через rank номер текущей ветви, реализующей операторы. Введем операторы ветви 0:

$F_{0,1}$: формирование начального решения $x^{(0)} \in D$; $L_{0,2}$: $x^* := x^{(0)}$; $L_{0,3}$: $k := 0$;

$A_{0,4}$: вычисление $\Delta F_{\max}$; $L_{0,5}$: $c_0 := \Delta F_{\max}$; $P_{0,6}$: $P\{c_k \ge c_R\}, P = 0 \to Я_{0,14}$;

$S_{0,7}$: передача решения $x^{(k)}$ ветвям $1, 2, \ldots, n-1$;

$T_{0,8}$: прием значений целевых функций $F(x^*)$ из каждой ветви;

$F_{0,9}$: определение номера r ветви, передавшей наименьшее значение целевой функции;

$R_{0,10}$: прием в $x^{(k+1)}$ решения x^* от ветви r; $L_{0,11}$: $x^* := x^{(k+1)}$; $L_{0,12}$: $k := k + 1$;

$\Pi_{0,13} :\to P_{0,6}$; $Я_{0,14}$: конец.

Операторы ветвей $i = 1, 2, \ldots, n-1$:

$L_{i,1}$: $k := 0$; $A_{i,2}$: вычисление $\Delta F_{\max}$; $L_{i,3}$: $c_0 := \Delta F_{\max}$;

$P_{i,4}$: $P\{c_k \ge c_R\}$, $P = 0 \to Я_{i,23}$; $R_{i,5}$: прием решения $x^{(k)}$ от ветви 0; $L_{i,6}$: $x^* := x^{(k)}$;

$L_{i,7}$: $d := 0$; $P_{i,8}$: $P\{d \le M/(n-1)\}$, $P = 0 \to T_{i,17}$;

$\Phi_{i,9}$: выбор решения $y \in N(x^{(k)})$ из окрестности текущего;

$\Phi_{i,10}$: формирование псевдослучайного числа $\xi \in [0, 1]$;

$P_{i,11}$: $P\{\xi < P(x^{(k)}, y, k)\}$, $P = 0 \to P_{i,13}$; $L_{i,12}$: $x^{(k=1)} := y$;

$P_{i,13}$: $P\{F(y) \le F(x^*)\}$, $P = 0 \to L_{i,15}$; $L_{i,14}$: $x^* := y$; $L_{i,15}$: $d := d + 1$; $\Pi_{i,16} :\to$ $:\to P_{i,8}$;

$T_{i,17}$: передача значений целевых функций $F(x^*)$ из каждой ветви во все остальные;

$F_{i,18}$: определение номера r ветви, передавшей наименьшее значение целевой функции;

$P_{i,19}$: $P\{r = \text{rank}\}$, $P = 0 \to L_{i,21}$; $S_{i,20}$: передача решения x^* ветви 0; $L_{i,21}$: $k := k + 1$;

$\Pi_{i,22} :\to P_{i,4}$; $Я_{i,23}$: конец.

Операторная схема алгоритма PTA_PSA имеет следующий вид:
Ветвь 0:

$$F_{0,1} L_{0,2} L_{0,3} A_{0,4} L_{0,5}\, {}^{13}P_{0,6\downarrow 14} S_{0,7} T_{0,8} F_{0,9} R_{0,10} \Pi^{6}_{0,13} Я_{0,14}.$$

Ветви $i = 1, 2, \ldots, n-1$:

$$L_{i,1} A_{i,2} L_{i,3}\, {}^{22}P_{i,4\downarrow 23} R_{i,5} L_{i,6} L_{i,7}\, {}^{18}P_{i,8\downarrow 17} \Phi_{i,9} \Phi_{i,10} P_{i,11\downarrow 13} L_{i,12}\, {}^{11}P_{i,13\downarrow 15}$$

$$L_{i,14}\, {}^{13}L_{i,15} \Pi^{8}_{i,16}\, {}^{8}T_{i,17} F_{i,18} P_{i,19\downarrow 21} S_{i,20}\, {}^{19}L_{i,21} \Pi^{4}_{i,22} Я_{i,23}.$$

Моделирование алгоритмов. Описанные алгоритмы реализованы на языке программирования C++ и экспериментально исследованы на кластере Центра параллельных вычислительных технологий Сибирского государственного университета телекоммуникаций и информатики [9]. При разработке параллельной версии алгоритма использовалась библиотека стандарта MPI – MPICH2 для операционной системы GNU/Linux.

Для исследования работы алгоритмов генерировались тестовые конфигурации распределенных ВС с числом процессорных ядер $N = 256$, 1024, 4096, 16384, 65536 путем случайного выбора множества подсистем с количеством ядер $n_i \in \{64, 128, 256, 512, 1024, 2048, 4096, 16384, 65536\}$. В качестве коммуникационных сред для 1-го уровня рассматривались технологии: Gigabit Ethernet, Infiniband и Myrinet. При наличии 2-го уровня брались значения показателей производительности среды доступа к общей памяти. Значения показателей производительности каналов связи, через которые взаимодействуют подсистемы, выбирались случайно из множества {0,1, 1,0, 10, 100, 1000 Мбит/с}.

В качестве структур информационных графов параллельных программ рассматривались: линейка, кольцо, звезда и решетка. Для каждой структуры строились графы с количеством вершин $M = 256, 512$, 1024, 2048 с однородными и неоднородными по значениям w_i и d_{ij} ребрами.

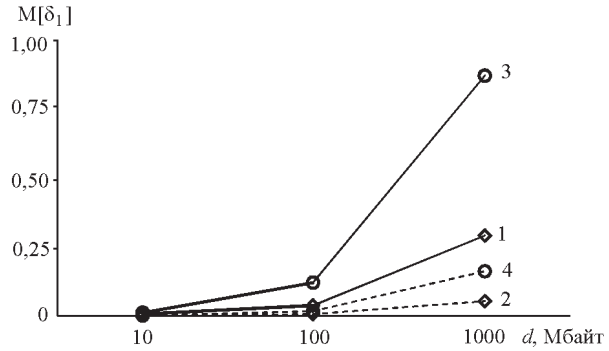

Рис. 2. Зависимость оценки $\mathrm{M}[\delta_1]$ от объема данных, передаваемых между ветвями параллельной программы ($N = 16384$, $n_i = 256$, $M = 1024$). Структуры информационных графов: 1 – линейка $m_{ij} = 1$ кбайт; 2 – линейка $m_{ij} = 1$ Мбайт; 3 – решетка $m_{ij} = 1$ кбайт; 4 – решетка $m_{ij} = 1$ Мбайт

В качестве оценок эффективности работы алгоритмов рассматривались величины

$$\delta_1 = (\widetilde{F} - F)/F, \quad \delta_2 = (\widetilde{F} - F - t)/(F + t), \quad \delta_3 = (F_0 - F)/F,$$

где t – время работы алгоритма, F_0 – значение целевой функции от начального решения, а F и $\widetilde{F}$ – значения целевой функции от решения, получаемого алгоритмом и случайным назначением соответственно.

Моделирование показало, что для $\delta_1, \delta_2, \delta_3$ оценки математических ожиданий и среднеквадратических отклонений для рассмотренных тестовых наборов данных составили $\mathrm{M}[\delta_1] = 74{,}13$, $\sigma[\delta_1] = 17{,}06$, $\mathrm{M}[\delta_2] = 74{,}13$, $\sigma[\delta_2] = 17{,}06$, $\mathrm{M}[\delta_3] = 0{,}27$, $\sigma[\delta_3] = 0{,}02$. В среднем алгоритм PTA_SA на рассмотренных наборах модельных данных позволяет получать решение задачи в 75 раз лучше случайного назначения ветвей на процессорные ядра и в 1,27 раза лучше начального решения.

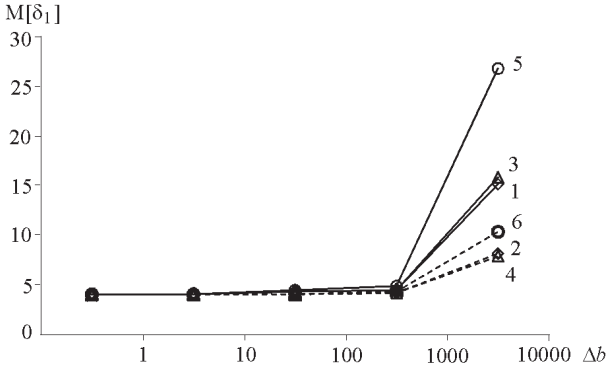

Рис. 3. Зависимость оценки $\mathrm{M}[\delta_1]$ от Δb ($N = 16384$, $H = 20$, $M = 1024$). Структуры информационных графов: 1 – однородная линейка, 2 – неоднородная линейка, 3 – однородное кольцо, 4 – неоднородное кольцо, 5 – однородная решетка, 6 – неоднородная решетка

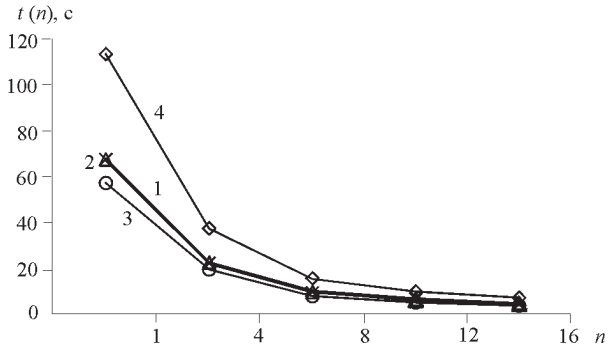

Рис. 4. Зависимость времени выполнения алгоритма PTA_PSA от числа используемых процессоров (N = 131072, M = 2048). Структуры информационных графов: 1 – линейка, 2 – кольцо, 3 – звезда, 4 – решетка

Качество назначения существенно зависит от объемов данных, передаваемых между ветвями параллельной программы, а также от размеров передаваемых сообщений. На рис. 2 приведены графики зависимости значений оценки $M[\delta_1]$ от объемов данных, передаваемых между ветвями однородной параллельной программы (конфигурация распределенной ВС однородная).

Неоднородность каналов связи на различных уровнях коммуникационной подсистемы распределенной ВС также оказывает заметное влияние на качество назначения. На рис. 3 представлены графики зависимости значения $M[\delta_1]$ от величины Δb – отношения максимального значения пропускной способности каналов связи внутри кластеров (1-й уровень) к максимальному значению пропускной способности каналов связи между кластерами (0-й уровень).

Графики зависимости времени работы параллельного алгоритма PTA_PSA от количества используемых процессоров приведены на рис. 4.

Заключение. Результаты исследования алгоритма PTA_SA в данной работе позволяют рекомендовать его для применения на большемасштабных распределенных ВС, особенно на гетерогенных конфигурациях (например, мультикластерных ВС и GRID-системах), а также при решении сложных параллельных задач. Разработанные алгоритмы могут быть использованы для оптимизации запуска параллельных MPI-программ на мультикластерных ВС.

СПИСОК ЛИТЕРАТУРЫ

1. **Хорошевский В. Г.** Архитектура вычислительных систем. М.: МГТУ им. Н. Э. Баумана, 2005.
2. **Евреинов Э. В., Хорошевский В. Г.** Однородные вычислительные системы. Новосибирск: Наука, 1978.
3. **Bhanot G., Gara A., Heidelberger P. et al.** Optimizing task layout on the Blue Gene/L supercomputer // IBM Journ. Research and Development. 2005. **49**, N 2. P. 489.
4. **Bokhari S. H.** On the mapping problem // IEEE Trans. on Comput. 1981. **30**, N 3. P. 207.

5. **Lee C. H., Shin K. G.** Optimal task assignment in homogeneous networks // IEEE Trans. Parallel Distributed Systems. 1997. **8**, N 2. P. 119.

6. **Dimitrov R., Skjellum A.** Impact of latency on applications' performance // Proc. of the Fourth MPI Developer's and User's Conference. N. Y.: Cornell University, 2000.

7. **Береснев В. Л.** Дискретные задачи размещения и полиномы от булевых переменных. Новосибирск: ИМ СО РАН, 2005.

8. **Кочетов Ю. А.** Вероятностные методы локального поиска для задач дискретной оптимизации // Дискретная математика и ее приложения. М: МГУ, 2001. С. 87.

9. **Khoroshevsky V. G., Mamoilenko S. N., Maidanov Y. S., Sedelnikov M. S.** Space-distributed multi-cluster computer system for parallel multiprogramme regimes modeling // Proc. of the Intern. conf. "SIMULATION-2006". Kiev: Institute for Modeling in Energy Engineering NASc of Ukraine, 2006. P. 67.

Поступила в редакцию 23 июля 2007 г.