РОССИЙСКАЯ АКАДЕМИЯ НАУК

СИБИРСКОЕ ОТДЕЛЕНИЕ

А В Т О М Е Т Р И Я

2003, том 39, № 2

МОДЕЛИРОВАНИЕ СИСТЕМ

УДК 681.324

В. Г. Хорошевский, С. Н. Мамойленко

(Новосибирск)

СТРАТЕГИИ СТОХАСТИЧЕСКИ ОПТИМАЛЬНОГО ФУНКЦИОНИРОВАНИЯ РАСПРЕДЕЛЕННЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ*

Развивается теоретико-игровой подход к организации диспетчеризации распределенных вычислительных систем (ВС) в режиме обслуживания потока параллельных задач. Предлагаются параллельные алгоритмы для отыскания оптимальных смешанных стратегий выбора задач для решения на не абсолютно надежных ВС. Приводятся результаты параллельного моделирования функционирования распределенных ВС.

Современный инструментарий индустрии обработки информации – это распределенные вычислительные системы (ВС) высокой производительности (10^9–10^{15} опер./с, GigaFLOPS – PetaFLOPS). Архитектура распределенных ВС представляется в виде композиции множества элементарных машин (ЭМ) или процессоров и сети связей между ними. В таких системах все основные ресурсы (не только арифметико-логические устройства, но и память, и средства управления) являются и логически, и технически распределенными. Число процессоров в распределенных ВС допускает варьирование и заключено в пределах от 10 до 10^6. Например, вычислительная система NEC "Earth-Simulator" обладает быстродействием 35 TeraFLOPS и имеет в своем составе 5120 процессоров. Создаваемая система IBM "Blue Gene" рассчитывается на быстродействие 1 PetaFLOPS и будет состоять из 1 000 000 процессоров. Именно поэтому подобные ВС относят к масштабируемым и большемасштабным.

По архитектурным возможностям промышленные ВС достаточно близки к вычислительным системам с программируемой структурой, разработка

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

концептуальных основ построения которых была завершена в Сибирском отделении АН СССР к началу 70-х годов 20-го столетия [1].

В качестве примеров отечественных ВС с программируемой структурой могут служить [1–3]: первая система «Минск-222» (1965 г.); мультиминимашинные ВС МИНИМАКС (1975 г.) и СУММА (1976 г.); мультипроцессорные системы семейства МИКРОС: МИКРОС-1 (1986 г.), МИКРОС-2 (1992 г.), МИКРОС-Т (1998 г., MIMD-архитектура, варьируемое число транспьютеров, произвольные графы для межтранспьютерных сетей связей со степенями вершин от 2 до 6, живучесть, распределенная операционная система); суперкомпьютеры семейства МВС [4]: МВС-100 и МВС-1000 (1999 г.).

Распределенная ВС по своей природе (например, вследствие отказов ресурсов) – стохастический объект, который предназначается в общем случае для обслуживания вероятностных потоков заданий, представленных параллельными программами со случайными параметрами (числом ветвей, временем решения и т. п.). Следовательно, в этих условиях целесообразны постановки задач о стохастически оптимальном функционировании ВС. Для организации функционирования ВС в случае достаточно высокой интенсивности потока задач (если в системе имеется конечная очередь задач, ожидающих своего решения) могут быть применены методы теории игр [5–9].

1. **Постановка проблемы.** Данная работа продолжает цикл публикаций по теоретико-игровому подходу [5–7] к организации функционирования большемасштабных распределенных ВС. В [7] предполагалось, что надежность ВС позволяет выделять для решения задач столько элементарных машин, сколько их нужно для удовлетворения оптимальной смешанной стратегии. Однако на практике оптимальная смешанная стратегия выбора количества ЭМ может заметно отличаться от стационарного распределения вероятностей состояний ВС [1, 2]. Именно поэтому в данной работе рассматривается общий случай, учитывающий отказы элементарных машин вычислительной системы.

1.1. *Вычислительный центр.* Пусть имеется вычислительный центр (ВЦ), который эксплуатирует распределенную ВС, состоящую из N не абсолютно надежных элементарных машин. Реализация параллельных программ осуществляется на основной подсистеме из n ЭМ; $(N-n)$ машин составляют структурную избыточность [1]. В случае отказа любой машины основной подсистемы в ВС реализуется процедура восстановления, которая заключается в поиске в пределах основной подсистемы отказавшей ЭМ и реконфигурации структуры ВС. Поиск отказавшей ЭМ осуществляется средствами (само)диагностики ВС. Реконфигурация системы заключается в программной настройке новой n-машинной конфигурации из всех исправных ЭМ существующей основной подсистемы и части ЭМ структурной избыточности. Последняя процедура осуществляется реконфигуратором.

Средства (само)диагностики и реконфигуратор являются компонентами операционной системы (ОС). Эта композиция, по сути, является виртуальным восстанавливающим устройством (ВУ). В соответствии с принципом децентрализованного управления можно считать, что в каждой ЭМ есть свое ВУ и, следовательно, в распределенной ВС количество ВУ равно количеству ЭМ, входящих в ее состав.

Пусть λ – интенсивность потока отказов в ЭМ (среднее число отказов, появляющихся в машине в единицу времени), а μ – интенсивность восстановления отказавших ЭМ одним из восстанавливающих устройств.

Будем говорить, что распределенная ВС находится в состоянии $i \in E$, $E = \{0,1,2,\ldots,N\}$, если в ней имеется i исправных ЭМ. Далее, пусть $\{p_i\}$, $i \in E$, – распределение вероятностей состояний системы в стационарном режиме работы [2]:

$$p_i = \frac{N!}{(N-i)!\,i!}\,\frac{\mu^i \lambda^{N-i}}{(\lambda+\mu)^N}. \qquad (1)$$

1.2. *Диспетчер.* Допустим, что на распределенную ВС поступает случайный поток параллельных задач такой интенсивности, что имеет место их конечная очередь $\mathfrak{I}$. Пусть ожидающая решения задача $I_j^r \in \mathfrak{I}$ представлена парой чисел $\langle t_j^r, r\rangle$, где t_j^r – время решения задачи, а r – ее ранг. Считается, что задача имеет ранг $r \in E^* = \{0,1,2,\ldots,n\}$, если для ее решения необходимо использовать r машин ВС (подсистему ранга r).

Пусть $\mathfrak{I}^r$ – подмножество задач I_j^r ранга r: $\bigcup_j I_j^r = \mathfrak{I}^r$, $r \in E^*$. Очевидно, что $\mathfrak{I}^r \subset \mathfrak{I}$, $\bigcup_r \mathfrak{I}^r = \mathfrak{I}$, $\mathfrak{I}^r \cap \mathfrak{I}^k = \varnothing$ для $r \neq k$, $r, k \in E^*$. Считается, что число элементов в подмножестве $\mathfrak{I}^r \subset \mathfrak{I}$ достаточно большое и допускает разбиение $\mathfrak{I}^r$ на такие группы G^r, для которых выполняется равенство

$$\left| \sum_j t_j^r - \Theta \right| = o(\Theta),$$

где $\sum_j t_j^r$ есть суммарное время решения всех задач $I_j^r \in G^r$, а Θ – константа времени, выбранная заранее. В дальнейшем будем считать, что Θ есть единица времени, и оперировать каждой группой G^r как задачей, имеющей ранг $r \in E^*$ и единичное время решения Θ.

Допускаем также, что $\mathfrak{I}^0 \neq \varnothing$, $\mathfrak{I}^0 \subset \mathfrak{I}$. Любая задача, принадлежащая $\mathfrak{I}^0 \subset \mathfrak{I}$, требует для «своего решения» точно единицу времени и нуль машин ВС. Следует заметить, что если некоторое $\mathfrak{I}^r = \varnothing$, то подмножество ранга r должно быть сформировано из групп G^k, $k < r$, $k, r \in E^*$.

Далее, имеется также диспетчер (компонента ОС), который в дискретные моменты времени $t = 0,1,2,\ldots$ назначает параллельные задачи (точнее, группы G^r, $G^r \subset \mathfrak{I}^r \subset \mathfrak{I}$) для решения на распределенной ВС.

Затраты на решение той или иной задачи G^r, $r \in E^*$, будем интерпретировать как «платеж» диспетчера вычислительному центру. Требуется создать такой алгоритм работы комплекса «ВЦ – диспетчер», который в условиях не абсолютной надежности распределенной ВС обеспечивал бы в единицу времени минимум затрат на решение задач.

1.3. *Игровые модели функционирования распределенных ВС.* Имеется ВЦ, который в дискретные моменты времени t выделяет ресурсы распределенной ВС для решения задач. Пусть ресурсы вычислительного центра составляют n элементарных машин распределенной ВС. Известно распределе-

ние вероятностей состояний вычислительной системы в стационарном режиме функционирования: $\mathcal{P}=\{p_i\}$, $i \in E^*$.

Пусть также имеется диспетчер, который в дискретные моменты времени t назначает для решения на ВС задачи G^r различных рангов $r \in E^*$, но с единичным временем решения Θ.

Следуя игровой терминологии, объекты «ВЦ» и «диспетчер» будем называть игроками. Фактически эти игроки являются виртуальными, они представлены соответствующими компонентами ОС.

Далее считаем, что ВЦ и диспетчер используют чистые стратегии i и r, $r \in E^*$, соответственно, если первый игрок для решения задач отводит i машин, а второй игрок назначает задачу ранга r. Если ВЦ выбирает стратегию с номером i, а диспетчер – стратегию с номером r, то диспетчер «платит» ВЦ сумму c_{ir}. Элементы c_{ir}, $i, r \in E^*$, составляют матрицу **C** платежей [5, 8].

Предложенная модель функционирования распределенных ВС относится к классу антагонистических игр двух объектов с нулевой суммой, так как интересами ВЦ является получение максимального дохода от выделенных ресурсов, а интересами диспетчера – минимизация затрат на решение задач.

Подбор элементов платежных матриц **C** должен осуществляться с учетом конкретных условий эксплуатации распределенных ВС. В работе [5] предложена следующая формула для элементов матрицы **C**:

$$c_{ir} = \begin{cases} rc_1 + (i-r)c_2 & \text{при } i \geq r; \\ ic_2 + (r-i)c_3 & \text{при } i < r, \end{cases} \qquad (2)$$

где c_1 – цена эксплуатации одной ЭМ распределенной ВС в единицу времени (платеж вычислительному центру за решение задачи единичного ранга); c_2 – величина штрафа, выплачиваемого диспетчером в единицу времени за одну простаивающую ЭМ; c_3 – величина штрафа, налагаемого на диспетчера в единицу времени, если номер его стратегии больше на единицу номера стратегии ВЦ, т. е. $r = i+1$, $i, r \in E^*$.

Практически наиболее вероятной и общей ситуацией является та, при которой в потоке имеются задачи всех рангов. Поэтому алгоритм функционирования ВС, состоящий в назначении задач одного ранга, представляется неэффективным. Следовательно, рассматриваемая игра не должна иметь решения в чистых стратегиях, т. е. матрица $\mathbf{C} = \|c_{ir}\|$ не должна иметь седловых точек [8]. В работе [5] доказана следующая теорема: матрица **C** не имеет седловых точек тогда и только тогда, когда $c_1 < \min\{c_2, c_3\}$.

Ясно, что при стратегии диспетчера $r > i$, $i, r \in E^*$, ресурсов ВС недостаточно для решения назначенной задачи, следовательно, имеет место простой всех i выделенных машин. Данное замечание позволяет осуществлять расчет элементов матрицы платежей по формуле

$$c_{ir} = \begin{cases} rc_1 + (i-r)c_2 & \text{при } i \geq r; \\ ic_4 & \text{при } i < r, \end{cases} \qquad (3)$$

где c_4 – удельные потери диспетчера из-за невозможности решения задачи вследствие того, что ее ранг больше количества выделенных ЭМ. Матрица **C** не будет иметь седловых точек, если неравенство $c_1 < \min\{c_2, c_4\}$ выполняет-

ся. Доказательство этого утверждения аналогично доказательству, приведенному в [5].

В [1, 5–7] разработаны игровые модели и параллельные алгоритмы отыскания оптимальных смешанных стратегий ВЦ и диспетчера при заданных матрицах платежей.

Модель 1. Пусть заданы матрица платежей $\mathbf{C} = \|c_{ir}\|$, $i, r \in E^*$, и смешанная стратегия $\mathcal{P} = \|p_0, p_1, \ldots, p_i, \ldots, p_n\|$ вычислительного центра. Пусть также $\mathcal{P}$ является распределением вероятностей (1) состояний ВС, тогда все исправные ЭМ системы автоматически выделяются для решения задач. Требуется найти оптимальную смешанную стратегию $\Pi^* = \|\pi_0^*, \pi_1^*, \ldots, \pi_r^*, \ldots, \pi_n^*\|$ диспетчера. Предложенная модель суть игра с «природой», в которой в качестве последней выступает распределенная ВС.

Оптимальная смешанная стратегия Π^* диспетчера есть решение следующей экстремальной задачи:

$$F(\Pi^*) = \min_{\Pi} F(\Pi),$$

$$F(\Pi) = \sum_{r=0}^{n} \pi_r \sum_{i=0}^{n} p_i c_{ir}, \tag{4}$$

где целевая функция $F(\Pi)$ – расходы диспетчера при использовании смешанной стратегии $\Pi = \|\pi_r\|$, $r \in E^*$.

Так как вероятности состояний ВС известны, то оптимальная смешанная стратегия Π^* диспетчера имеет координаты $\pi_r^* = 0$, $r \in E^*$, $r \neq r^*$; $\pi_{r^*}^* = 1$, где r^* – любая стратегия диспетчера, для которой выполняется условие

$$\sum_{i=0}^{n} p_i c_{ir^*} = \min_{r} \left\{ \sum_{i=0}^{n} p_i c_{ir} \right\}. \tag{5}$$

Таким образом, в данной ситуации наиболее эффективно решать задачи ранга r^*, определяемого из условия (5). При решении трудоемких задач это требование легко удовлетворить. В самом деле, для достаточно широкого круга задач с большим объемом вычислений [1] могут быть составлены адаптирующиеся параллельные программы, настраиваемые на любое число ЭМ, в частности, равное r^*.

Модель 2. В отличие от модели 1 здесь будем считать, что если вычислительная система находится в состоянии $k \in E^*$, то для решения задач ВЦ может выделить i $(0 \le i \le k)$ элементарных машин.

Рассматривается следующая трехходовая игра [6]. Первый ход делает случайный механизм, который выбирает число $k \in E^*$ с вероятностью p_k. Вычислительный центр, зная k, выставляет для решения i машин (второй ход), где $0 \le i \le k$. Диспетчер независимо от ВЦ назначает задачу ранга r (третий ход) и платит ВЦ c_{ir} денежных единиц (c_{ir} определяется только i и r, не зависит от p_k, рассчитывается по формуле (2) или (3)).

Модель 2 позволяет организовать матричную игру. В отличие от способа построения матричной игры, предложенного в [1], здесь осуществляется «сокрытие» информации о вероятностях состояний системы от диспетчера. Достичь этого можно, если изменить платежные коэффициенты с учетом распределения вероятностей состояний системы:

$$c'_{ir} = c_{ir} \sum_{k=1}^{n} p_k .$$

Итак, требуется для матрицы платежей $\mathbf{C}' = \| c'_{ir} \|$, $i, r \in E^*$, найти решение $\{\mathcal{P}^*, \Pi^*\}$ и цену V игры, где $\mathcal{P}^*$ и Π^* – оптимальные смешанные стратегии ВЦ и диспетчера соответственно; $V = \mathcal{P}^* \mathbf{C}' (\Pi^*)^{\mathrm{T}}$; $(\Pi^*)^{\mathrm{T}}$ – транспонированный вектор Π^*; c_{ij} вычисляются по формуле (2) или (3); p_k определяется формулой (1).

2. **Методы решения матричных игр.** Конечные матричные игры могут быть решены известными методами, например: итеративными методами Дж. фон Неймана [8] и Брауна-Робинсон [9], симплекс-методом [10]. Последний применим, так как любая матричная игра может быть сведена к каноническому виду задачи линейного программирования. Для решения поставленных задач необходимо использовать такие методы, которые позволяли бы найти решение при больших размерах матриц платежей.

В данной работе сравниваются эффективности метода Брауна – Робинсон и симплекс-метода (последовательных и параллельных). В работе [7] предлагается параллельный ускоренный метод Брауна – Робинсон (ПУМИ), который сочетает в себе достоинства обоих рассматриваемых методов. Метод ПУМИ применим в случае кососимметричности матрицы платежей. В рассматриваемых моделях матрицы платежей не являются кососимметричными, поэтому использование метода ПУМИ является неэффективным. Однако метод ПУМИ позволяет найти заведомо невыгодные стратегии, что не могут сделать рассматриваемые методы. В работе [11] приводится параллельный алгоритм симплекс-метода. В данной работе предлагается иной способ распределения симплекс-матрицы по ЭМ, который позволяет получить легко адаптируемую под имеющееся количество доступных ЭМ параллельную программу.

В данной работе в качестве критерия завершения итерационного процесса для метода Брауна – Робинсон использован критерий $\varepsilon = \varepsilon_0 \max(N, R)$, где ε – значение оценки оптимальности решения; ε_0 – требуемое значение погрешности; N – число ЭМ в ВС; R – максимальный (допустимый) ранг задачи. Последнее позволило увеличить сходимость метода Брауна – Робинсон при адекватном значении погрешности решения (т. е. величина погрешности – разница между максимальным проигрышем и минимальным выигрышем игроков [8] – составляет менее 10 % от их среднего выигрыша).

2.1. *Параллельный метод Брауна – Робинсон.* Суть метода Брауна – Робинсон [8, 9] заключается в фиктивном разыгрывании игровой ситуации до тех пор, пока разница между оценками средних выигрышей при l и $l+1$ шагах не будет меньше заданной величины ε.

Для обеспечения параллельного счета требуется каждую ЭМ заставить вычислять только свои элементы платежной матрицы $\mathbf{C}$, векторов $\mathbf{X}(l)$, $\mathbf{Y}(l)$, Π, $\mathcal{P}$, где $\mathbf{X}(l)$, $\mathbf{Y}(l)$ – векторы распределения относительного выигрыша пер-

вого и второго игроков при l-м разыгрывании партии. При этом все данные можно разместить полностью в каждой ЭМ, однако такое распределение потребует большой памяти. Очевидно, что для хранения данных будет использована минимальная емкость памяти, если матрицу и векторы разбить на n равных частей и в каждую ЭМ разместить по одной такой части.

Организация процесса вычислений (при использовании векторной системы первого типа [8]).

1. Каждая ЭМ находит номера и значения минимальной и максимальной компонент векторов $\mathbf{X}_s(l)$ и $\mathbf{Y}_s(l)$, где l – номер текущей итерации; s – номер ЭМ в системе, $s \in E^*$; $\mathbf{X}_s(l)$ и $\mathbf{Y}_s(l)$ – это части векторов $\mathbf{X}(l)$ и $\mathbf{Y}(l)$, распределенные в ЭМ с номером s.

2. Определяются номера и значения минимальной и максимальной компонент векторов $\mathbf{X}(l)$ и $\mathbf{Y}(l)$ и проверяется условие завершения итераций (разница между минимальным выигрышем и максимальным проигрышем меньше чем ε).

3. Если условие завершения итераций выполняется, то решение найдено и итеративный процесс завершается, иначе переходим на п. 4.

4. Каждая ЭМ прибавляет к $\mathbf{X}_s(l)$ и $\mathbf{Y}_s(l)$ соответственно строку i^+ и столбец r^+, найденные в п. 2. Элементарные машины, в чей диапазон попадают i^+ и r^+, увеличивают значения соответствующих компонент векторов Π_s, $\mathcal{P}_s$.

5. Переходим на п. 1.

Если используется векторная система второго типа [8, 9], то минимальная компонента $\mathbf{X}(s)$ и максимальная компонента вектора $\mathbf{Y}(s)$ находятся последовательно (т. е. сначала определяется i^+, затем соответствующая строка прибавляется к вектору $\mathbf{Y}(s)$ и определяется значение r^+).

2.2. *Параллельный симплекс-метод.* Известно [8], что любую конечную антагонистическую игру можно свести к задаче линейного программирования, которая решается симплекс-методом [10, 11].

Симплекс-таблица распределяется по ЭМ следующим образом: строки матрицы, соответствующие уравнениям ограничений, делятся на s равных частей, строки, соответствующие коэффициентам целевых функций, – на две части (одна часть соответствует коэффициентам основной целевой функции, другая – искусственной), каждая из которых, в свою очередь, также делится на s равных частей.

Такое распределение справедливо для случая, когда симплекс-таблица получена для минимизирующего игрока. При этом начальный базис явно не выражен, и поэтому необходимо использовать дополнительную процедуру для его нахождения.

Если матрица получена относительно максимизирующего игрока, то базис выражен явно и выполнение дополнительных действий не требуется. При этом матрица распределяется между ЭМ аналогичным образом (исключая из деления коэффициенты искусственных переменных).

Итак, вследствие однородного распределения данных получаются одинаковые ветви параллельной программы. Однако ветви обрабатывают различные массивы данных. Так как каждой ветви своих данных недостаточно, они вступают во взаимодействие (обмениваются информацией).

Организация процесса вычислений.

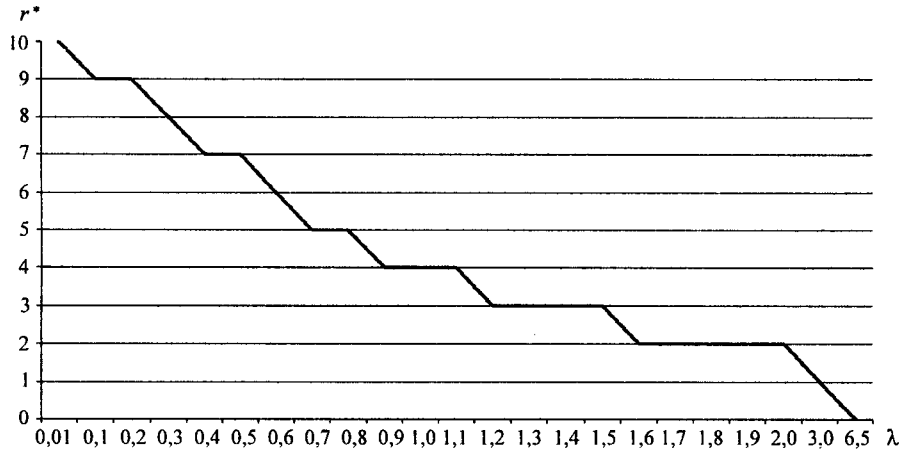

Рис. 1. Зависимость значений оптимальных рангов от надежности ЭМ

1. Каждая ЭМ находит переменную для включения в базис. Номер переменной корректируется исходя из множества номеров, полученных всеми ЭМ.

2. Каждая ЭМ находит переменную для исключения из базиса. Номер корректируется исходя из информации, полученной всеми ЭМ.

3. Строка и столбец, соответствующие разрешающему элементу, передаются всем ЭМ системы.

4. Каждая ЭМ пересчитывает «свои» коэффициенты и получает часть канонической формы.

Процедура повторяется до тех пор, пока на шаге 1 не будет найдена хотя бы одна переменная.

3. **Результаты моделирования.** 3.1. *Модель 1.* Очевидно, для того чтобы определить оптимальный ранг задач в модели 1, достаточно последовательно найти значения в каждой строке и одним из известных способов получить минимум в вектор-столбце. Однако такой поиск также можно распределить по ЭМ системы, что уменьшит время решения.

Например, для ВС, состоящей из 10 ЭМ, при значениях математических ожиданий времени безотказной работы ЭМ $\lambda^{-1} = 10000$ ч и времени восстановления ЭМ $\mu^{-1} = 10$ мин, а также при $c_1 = 2$, $c_2 = 4$, $c_3 = 3$ руб./ч оптимальный ранг задач равен максимальному рангу 10. Зависимость значения оптимального ранга r^* от значения λ показана на рис. 1.

3.2. *Модель 2.* Для решения поставленной задачи разработаны параллельные программы, реализующие рассмотренные алгоритмы (метод Брауна – Робинсон с двумя видами векторной системы и симплекс-метод с матрицами, приведенными относительно обоих игроков). Реализация была выполнена с использованием технологии MPI (LAM v.6.5.5), языка высокого уровня Си и ресурсов кластера Научно-учебного центра параллельных вычислительных технологий Сибирского государственного университета телекоммуникаций и информатики.

Результаты алгоритмов совпали с определенной точностью (см. таблицу). Значения коэффициентов платежной матрицы взяты из [5].

Сравнивая число шагов (рис. 2), требуемых для решения поставленной задачи, видим, что лидирующим методом оказывается симплекс-метод с

Количество ЭМ	МБР с векторной системой первого типа	МБР с векторной системой второго типа	Симплекс-метод с матрицей, приведенной относительно максимизирующего игрока	Симплекс-метод с матрицей, приведенной относительно минимизирующего игрока
1	2,41	2,40	2,40	2,40
2	4,87	4,87	4,86	4,86
4	9,99	9,99	9,97	9,97
8	20,24	20,25	20,21	20,21
16	40,75	40,77	40,66	40,66
32	81,70	81,74	81,58	81,58
64	163,85	163,78	163,46	163,46
128	327,92	327,75	327,22	327,22
256	655,85	656,16	654,74	654,74
512	1311,66	1312,76	1309,77	1309,68
1024	2623,05	2626,01	2619,85	2619,85
2048	5246,12	5252,69	5240,00	5239,98

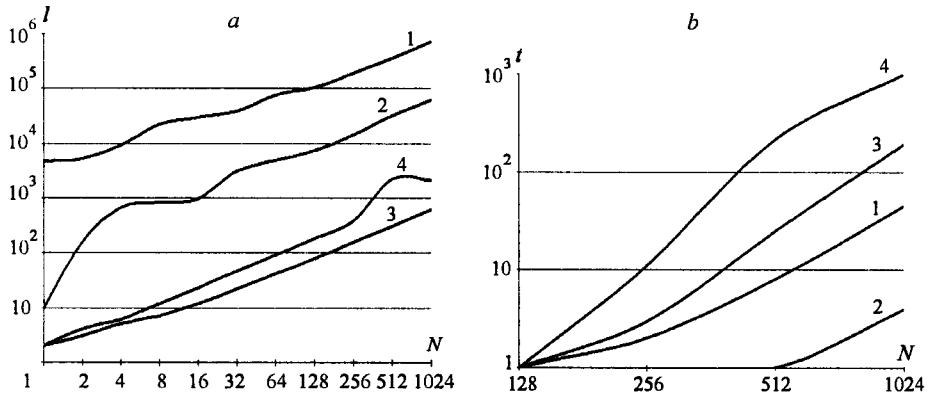

Рис. 2. Эффективность алгоритмов по числу шагов (*a*), по времени решения (*b*): кривая 1 – метод Брауна – Робинсон с векторной системой первого типа; 2 – с векторной системой второго типа; 3 – симплекс-метод с приведенной матрицей относительно максимизирующего игрока; 4 – относительно минимизирующего игрока

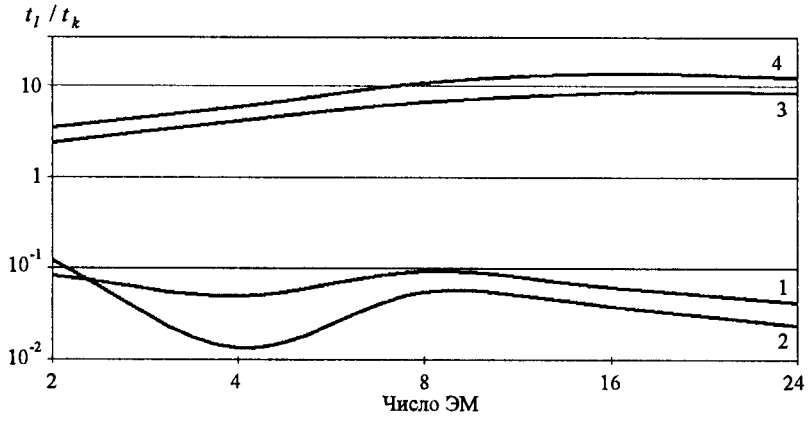

Рис. 3. Среднее ускорение параллельных алгоритмов

матрицей, приведенной относительно максимизирующего игрока. Чтобы найти решение для ВС, состоящей из 2048 ЭМ, ему потребовалось всего лишь 1510 шагов. Однако времени он затратил больше всех. Поэтому говорить о его эффективности сложно. По времени решения наиболее оптимальным оказался метод Брауна – Робинсон с векторной системой второго типа. Следовательно, необходимо рассматривать эти два метода как конкурирующие.

Параллельные методы показали, что симплекс-метод с матрицей, полученной относительно максимизирующего игрока (ВЦ), «ускоряется» уже при размере матрицы, равном 256, и обеспечивает практически линейное «ускорение». Метод Брауна – Робинсон «ускоряется» при существенном увеличении размеров матрицы, когда число операций, выполняемых каждой ЭМ в отдельности, превышает число передач (рис. 3, кривые те же, что и на рис. 2).

СПИСОК ЛИТЕРАТУРЫ

1. **Евреинов Э. В., Хорошевский В. Г.** Однородные вычислительные системы. Новосибирск: Наука, 1978.
2. **Хорошевский В. Г.** Инженерный анализ функционирования вычислительных машин и систем. М.: Радио и связь, 1987.
3. **Khoroshevsky V. G.** MICROS: a family of large-scale distributed programmable structure computer systems // Proc. of Sixth Internat. Workshop on Distributed Data Processing. Novosibirsk: RAS Siberian Branch Publ., 1998. P. 65.
4. **Zabrodin A. V., Levin V. K., Korneev N. V.** The structure of MBC-100 massively parallel computer system and large scale applications // Proc. of 15th IMACS World Congress on Scientific Computation, Modeling and Applied Mathematics. Berlin: Wissenshaft & Technik Verlag, 1997. V. 6. P. 387.
5. **Хорошевский В. Г.** Об алгоритмах функционирования однородных вычислительных систем // Вычислительные системы. 1970. Вып. 39. С. 3.

6. **Хорошевский В. Г., Талныкин Э. А.** Теоретико-игровой подход к проблеме функционирования однородных вычислительных систем // Вычислительные системы. 1972. Вып. 51. С. 20.

7. **Хорошевский В. Г., Власюк В. В.** Теоретико-игровой подход к организации стохастически оптимального функционирования распределенных вычислительных систем // Автометрия. 2000. № 3. С. 17.

8. **Дрешер М.** Стратегические игры. Теория и приложения. М.: Сов. радио, 1964.

9. **Робинсон Дж.** Итеративный метод решения игр // Матричные игры. М.: Сов. радио, 1964. С. 110.

10. **Заварыкин В. М., Житомирский В. Г., Лапчик М. П.** Численные методы: учебное пособие для студентов физико-математических специальностей педагогических университетов. М.: Просвещение, 1990.

11. **Евреинов Э. В., Косарев Ю. Г.** Однородные вычислительные системы высокой производительности. Новосибирск: Наука, 1966.

Институт физики полупроводников ОИФП СО РАН,
Сибирский государственный университет
телекоммуникаций и информатики,
E-mail: khor@isp.nsc.ru,
msn@neic.nsk.su

Поступила в редакцию
26 ноября 2002 г.