РОССИЙСКАЯ АКАДЕМИЯ НАУК
СИБИРСКОЕ ОТДЕЛЕНИЕ
А В Т О М Е Т Р И Я
№ 2 2000

ПЕРСПЕКТИВНЫЕ ВЫЧИСЛИТЕЛЬНЫЕ СИСТЕМЫ

УДК 681.324

В. Г. Хорошевский, М. Н. Подаков
(Новосибирск)

ПОИСК СТОХАСТИЧЕСКИ ОПТИМАЛЬНОГО РАЗБИЕНИЯ БОЛЬШЕМАСШТАБНЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ НА ПОДСИСТЕМЫ*

Рассматривается стохастический подход к организации функционирования большемасштабных распределенных вычислительных систем (ВС). Ставится задача о стохастически оптимальном разбиении ВС на подсистемы в зависимости от потока параллельных программ. Решение задачи отыскивается методом динамического программирования: предлагается параллельный алгоритм.

Характерная особенность индустрии обработки информации – производство распределенных вычислительных систем (ВС) высокой производительности ($10^9 - 10^{12}$ опер./с). Такие ВС представляются в виде композиции множества элементарных машин (ЭМ, микропроцессоры) и сети межмашинных связей. В них основные ресурсы (не только процессоры, но и память, средства управления) являются физически распределенными. Число ЭМ в распределенных ВС достигает десятков тысяч, например, это число в системе CRAY T3D может быть равно 2048, в ASCI RED – 9152, в Connection Machine – 65536. Именно поэтому подобные ВС относятся к большемасштабным.

По своим архитектурным возможностям промышленные распределенные ВС близки к вычислительным системам с программируемой структурой [1–3]. В частности, они масштабируемы и обладают способностью программно настраиваться в зависимости от характера поступления задач, их структуры и параметров.

Распределенная ВС по своей природе (например, вследствие отказов ресурсов) – стохастический объект, который предназначается в общем случае для обслуживания случайных 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. Впервые такие постановки задач и подходы к их решению сделаны в [4].

* Работа выполнена при поддержке Российского фонда фундаментальных исследований (гранты № 97-01-00883, № 99-07-90206).

1. **Постановка задачи.** Основные режимы и особенности организации функционирования вычислительных систем коллективного пользования были описаны в работах [1–3]. Пусть ВС состоит из N элементарных машин, на систему через L терминалов поступает поток задач (программ). Под терминалом здесь понимается не только физическое устройство, предназначенное для загрузки программ, но и виртуальное устройство, формирующее поток задач, объединенных по некоторому признаку. Такая постановка особенно актуальна, когда работа с ВС осуществляется через локальную сеть или Internet. Итак, рассматривается функционирование ВС в мультипрограммном режиме в условиях потока задач (т. е. задачи поступают в случайные моменты времени и их параметры являются случайными величинами).

Каждая из задач, поступающих в систему, характеризуется наиболее приемлемыми для нее количеством ЭМ и структурой связей между ними. Выделенное множество, состоящее из j ЭМ, в дальнейшем будем называть подсистемой ранга j, $j \in \{1,2,\ldots,N\}$. Будем считать, что задача имеет ранг j, если для ее наиболее эффективного решения требуется подсистема ранга j. Итак, для того чтобы обеспечить решение задач, поступающих на терминалы, необходимо для каждого терминала выделить некоторое множество подсистем.

Считается, что на каждый терминал поступает поток задач различных рангов. Спросом на подсистемы ранга j с терминала l, $j \in \{1,2,\ldots,N\}$, $l \in \{1,2,\ldots,L\}$, назовем величину a_{jl}, выражающую количество подсистем, требующихся за некоторый промежуток времени T терминалу l для решения задач ранга j. Другими словами, a_{jl} – среднее число подсистем ранга j, которые могут быть загружены с терминала l за время T, $j = \overline{1,N}$, $l = \overline{1,L}$. Таким образом, для каждого терминала спрос на подсистемы различных рангов заранее неизвестен.

Допустим, известно, что a_{jl} – случайная величина с плотностью распределения вероятностей $p_{jl}(a)$; очевидно, что

$$\int_0^\infty p_{jl}(a)da = 1, \qquad j \in \{1,2,\ldots,N\}, \quad l \in \{1,2,\ldots,L\}. \tag{1}$$

Обозначим ρ_{jl} – математическое ожидание спроса на подсистему ранга j с терминала l:

$$\rho_{jl} = \int_0^\infty a p_{jl}(a)da, \qquad j \in \{1,2,\ldots,N\}, \quad l \in \{1,2,\ldots,L\}. \tag{2}$$

Полагаем, что разбиение ВС на подсистемы происходит в фиксированные моменты времени, причем T – интервал между разбиениями. Особенность такого подхода заключается в том, что для каждого промежутка времени T расчет разбиения производится только один раз, и считается, что в течение этого промежутка состав подсистем не меняется.

В начале очередного разбиения для терминала $l \in \{1,2,\ldots,L\}$ выделяется некоторое количество k_l элементарных машин, причем $\sum_{l=1}^{L} k_l = N^0 \ll N$ (условие большемасштабности ВС). Эти k_l элементарных машин образуют так называемое резервное множество для терминала l. Затем проводится рас-

пределение остальных $N - N^0$ ЭМ. Полагаем, что полученные таким образом множества ЭМ предназначены для решения задач, поступающих только на соответствующие им терминалы, и не могут быть использованы с других терминалов.

Пусть d_{jl} – цена эксплуатации подсистемы ранга j, а c_{jl} – стоимость ее формирования и обслуживания для терминала l. Для определения этих параметров может быть разработана некоторая схема приоритетов для потоков поступающих задач либо взяты реальные цены и стоимости обслуживания, выраженные в денежных единицах (в случае коммерческой эксплуатации ВС).

Рассмотрим случай, когда требуется обязательно выделить y_{jl} подсистем ранга j для терминала l: $\sum_{j=1}^{N}\sum_{l=1}^{L} jy_{jl} = N^1 < N$. Обозначим x_{jl} – количество подсистем ранга j, дополнительно выделяемых для терминала l. Таким образом, всего для терминала l выделяется $z_{jl} = y_{jl} + x_{jl}$ подсистем ранга j, где $x_{jl} \in \{0,1,\ldots,[n/j]\}$, $n = N - N^1 - N^0$, $j \in \{1,2,\ldots,N\}$, $l \in \{1,2,\ldots,L\}$ ($[x]$ – целая часть x). Ясно, что

$$\sum_{j=1}^{n}\sum_{l=1}^{L} jx_{jl} \le n, \qquad x_{jl} = 0 \text{ при } j = \overline{n+1, N}. \tag{3}$$

Итак, ожидаемые потери от избытка и недостатка подсистем ранга j на терминале l составят соответственно:

$$c_{jl}\int_0^{z_{jl}} (z_{jl} - a)p_{jl}(a)da, \qquad (d_{jl} - c_{jl})\int_{z_{jl}}^{\infty} (a - z_{jl})p_{jl}(a)da.$$

Тогда ожидаемая прибыль от эксплуатации подсистем ранга j с терминала l в силу (1) и (2) равна:

$$(d_{jl} - c_{jl})\rho_{jl} - (d_{jl} - c_{jl})\int_{z_{jl}}^{\infty} (a - z_{jl})p_{jl}(a)da - c_{jl}\int_0^{z_{jl}} (z_{jl} - a)p_{jl}(a)da =$$

$$= (d_{jl} - c_{jl})z_{jl} - d_{jl}\int_0^{z_{jl}} (z_{jl} - a)p_{jl}(a)da.$$

Таким образом, учитывая ограничения (3), получаем задачу:

$$\sum_{j=1}^{n}\sum_{l=1}^{L} r_{jl}(x_{jl}) \to \max_{\{x_{jl}\}}, \qquad j = \overline{1,n}, \quad l = \overline{1,L};$$

$$\sum_{j=1}^{n}\sum_{l=1}^{L} jx_{jl} \le n, \qquad x_{jl} \in \{0,1,\ldots,[n/j]\}, \quad n = N - \sum_{j=1}^{N}\sum_{l=1}^{L} jy_{jl} - \sum_{l=1}^{L} k_l;$$

$$r_{jl}(x)=(d_{jl}-c_{jl})(x+y_{jl})-d_{jl}\int_0^{x+y_{jl}}(x+y_{jl}-a)p_{jl}(a)da,$$

причем $x_{jl}=0$ при $j=\overline{n+1,N}$.

Эта задача может быть решена методом динамического программирования. Далее рассмотрим метод и алгоритм решения поставленной задачи.

2. **Метод решения (построение рекуррентных соотношений).** Динамическое программирование – один из наиболее мощных методов, используемых для решения экстремальных задач. Основная идея метода заключается в построении рекуррентных соотношений, связывающих исходную задачу с семейством аналогичных задач. Если среди этого семейства удается найти сравнительно просто решаемую задачу, то оптимальное решение исходной задачи может быть получено с помощью рекуррентных соотношений. Подробное описание метода можно найти в [5, 6], здесь же ограничимся построением рекуррентных соотношений для задачи, сформулированной в п. 1. Рассмотрим семейство $\langle p, j, l\rangle$-задач:

$$\sum_{i=j}^{n}\sum_{k=l}^{L}r_{ik}(x_{ik})\to\max_{\{x_{ik}\}},\qquad i=\overline{j,n},\quad k=\overline{l,L};$$

$$\sum_{i=j}^{n}\sum_{k=l}^{L}ix_{ik}\le p,\ x_{ik}\in\{0,1,\ldots,[p/i]\},\ j\in\{1,2,\ldots,n\},\ l\in\{1,2,\ldots,L\},\ p\in\{0,1,\ldots,n\};$$

$$r_{ik}(x)=(d_{ik}-c_{ik})(x+y_{ik})-d_{ik}\int_0^{x+y_{ik}}(x+y_{ik}-a)p_{ik}(a)da.$$

Обозначим $Q_{jl}(p)=\{0,1,\ldots,[p/j]\}$, $f_{jl}(p)$ – оптимальное (максимальное) значение функции прибыли $\sum_{i=j}^{n}\sum_{k=l}^{L}r_{ik}(x_{ik})$. В силу доказанных в [5] утверждений справедливы следующие рекуррентные соотношения:

$$f_{nL}(p)=\max_{x\in Q_{nL}(p)}\{r_{nL}(x)\},\ldots,f_{nl}(p)=\max_{x\in Q_{nl}(p)}\{r_{nl}(x)+f_{nl+1}(p-nx)\},\ldots$$

$$\ldots,f_{n1}(p)=\max_{x\in Q_{n1}(p)}\{r_{n1}(x)+f_{n2}(p-nx)\};\ldots$$

$$\ldots;f_{jL}(p)=\max_{x\in Q_{jL}(p)}\{r_{jL}(x)+f_{j+11}(p-jx)\},\ldots,f_{jl}(p)=$$

$$=\max_{x\in Q_{jl}(p)}\{r_{jl}(x)+f_{jl+1}(p-jx)\},\ldots,f_{j1}(p)=\max_{x\in Q_{j1}(p)}\{r_{j1}(x)+f_{j2}(p-jx)\};\ldots$$

$$\ldots;f_{1L}(p)=\max_{x\in Q_{1L}(p)}\{r_{1L}(x)+f_{21}(p-x)\},\ldots,f_{1l}(p)=$$

$$=\max_{x\in Q_{1l}(p)}\{r_{1l}(x)+f_{1l+1}(p-x)\},\ldots,f_{11}(p)=\max_{x\in Q_{11}(p)}\{r_{11}(x)+f_{12}(p-x)\},$$

где $l = L-1, L-2, \ldots, 2$; $j = n-1, n-2, \ldots, 2$; $p = 0, 1, \ldots, n$.

Обозначим $\{x_{11}^0, x_{12}^0, \ldots, x_{1L}^0; \ldots; x_{j1}^0, x_{j2}^0, \ldots, x_{jL}^0; \ldots; x_{n1}^0, x_{n2}^0, \ldots, x_{nL}^0\}$ – оптимальное решение; $\{p_{11}^0, p_{12}^0, \ldots, p_{1L}^0; \ldots; p_{j1}^0, p_{j2}^0, \ldots, p_{jL}^0; \ldots; p_{n1}^0, p_{n2}^0, \ldots, p_{nL}^0\}$ – соответствующие оптимальные состояния; $x_{jl}(p)$ – функция ($j \in \{1,2,\ldots,n\}, l \in \{1,2,\ldots,L\}$), значением которой в точке $p \in \{0,1,\ldots,n\}$ является такое $x \in Q_{jl}(p)$, что на нем достигается максимум в соответствующем рекуррентном соотношении. Тогда следующая вычислительная процедура даст искомое оптимальное решение:

$$p_{11}^0 = n, \quad x_{11}^0 = x_{11}(p_{11}^0); \ldots; p_{1l}^0 = p_{1l-1}^0 - x_{1l-1}^0,$$

$$x_{1l}^0 = x_{1l}(p_{1l}^0); \ldots; p_{1L}^0 = p_{1L-1}^0 - x_{1L-1}^0,$$

$$x_{1L}^0 = x_{1L}(p_{1L}^0); \ldots; p_{j1}^0 = p_{j-1L}^0 - (j-1)x_{j-1L}^0,$$

$$x_{j1}^0 = x_{j1}(p_{j1}^0); \ldots; p_{jl}^0 = p_{jl-1}^0 - jx_{jl-1}^0,$$

$$x_{jl}^0 = x_{jl}(p_{jl}^0); \ldots; p_{jL}^0 = p_{jL-1}^0 - jx_{jL-1}^0,$$

$$x_{jL}^0 = x_{jL}(p_{jL}^0); \ldots; p_{n1}^0 = p_{n-1L}^0 - (n-1)x_{n-1L}^0,$$

$$x_{n1}^0 = x_{n1}(p_{n1}^0); \ldots; p_{nl}^0 = p_{nl-1}^0 - nx_{nl-1}^0,$$

$$x_{nl}^0 = x_{nl}(p_{nl}^0); \ldots; p_{nL}^0 = p_{nL-1}^0 - nx_{nL-1}^0, \qquad x_{nL}^0 = x_{nL}(p_{nL}^0),$$

где $j = 2,3,\ldots,n-1$; $l = 2,3,\ldots,L-1$.

3. **Алгоритм.** Последовательные алгоритмы решения распределительных задач методом динамического программирования описаны в [5, 6]. Однако, рассчитывая оптимальное поведение вычислительной системы, эффективнее было бы воспользоваться ресурсами самой системы, но для этого необходим параллельный алгоритм. Для построения параллельного алгоритма будем использовать методику крупноблочного распараллеливания [1, 2]. Смысл данного подхода заключается в «расщеплении» алгоритма на слабо связанные ветви, число вычислительных операций в которых много больше числа операций обмена. Выполнение каждой ветви алгоритма может быть организовано на отдельной элементарной машине системы, при этом взаимодействия между ветвями будут осуществляться через сеть межмашинных связей.

Для распараллеливания выберем классический алгоритм с одним обратным и одним прямым ходом. Первый этап алгоритма (обратный ход) содержит nL шагов. Сначала вычисляются значения функций $f_{nL}(p), x_{nL}(p)$ в точках $p \in \{0,1,\ldots,n\}$. Далее, последовательно вычисляются функции $f_{jl}(p)$, $x_{jl}(p)$ в точках $p \in \{0,1,\ldots,n\}$. На последнем шаге достаточно по функции $f_{12}(p)$ вычислить $f_{11}(p), x_{11}(p)$ в точке $p = n$. Второй этап (прямой ход) также включает nL шагов. На первом шаге берется $p_{11}^0 = n$ и отыскивается $x_{11}^0 =$

$= x_{11}(p_{11}^0)$. Затем функции, найденные на предыдущем шаге, используются для вычисления p_{jl}^0, x_{jl}^0 и осуществляется переход к следующему шагу.

Рассматривая этот алгоритм, можно выделить сравнительно слабо связанные ветви на этапе вычисления значений функции $f_{jl}(p), j \in \{1,2,...,n\}$, $l \in \{1,2,...,L\}, p \in \{0,1,...,n\}$, а именно для каждой ветви выделить свой набор значений p, в которых будет вычисляться функция $f_{jl}(p)$. Так как на каждом этапе значения этой функции в конкретной точке p^* не зависят от ее значений в других точках p, то можно вычислять значения функции $f_{jl}(p)$ параллельно для всех точек $p \in \{0,1,...,n\}$. Используя это свойство, приведем схему параллельного алгоритма, имеющего конвейерную структуру. Предположим, что вычисления выполняются на $k+1$ элементарных машинах, причем на ЭМ с номером 0 производится вычисление только конечного результата, а на остальных ЭМ – промежуточные вычисления. Для описания алгоритмов введем следующие операторы:

Input – оператор ввода, предварительной обработки исходных данных и загрузки их в оперативную память ЭМ;

Output – оператор вывода результатов;

[E] означает: взятие целой части от выражения E;

x := E; означает: переменной x присвоить значение выражения E;

for x := E1 **to|downto** E2 **do** A; означает: циклическое выполнение оператора A, при котором переменной x последовательно присваиваются значения E1, E1 + 1, ..., E2 либо E1, E1 – 1, ..., E2;

if C **then** A1 **else** A2; означает: выполнение оператора A1, если верно условие C, и выполнение оператора A2, если условие C не верно;

фигурные скобки { , } ограничивают оператор, который может состоять из нескольких операторов;

First, *Last* – соответственно наименьшее и наибольшее значения p для каждой ЭМ, в которых будет определяться функция $f_{jl}(p)$ (здесь и далее $j \in \{1,2,...,n\}, l \in \{1,2,...,L\}$);

trans – вспомогательная переменная;

Take(x, a) – оператор записи данных, передаваемых процессором с номером a;

Put(x, a) – оператор передачи данных процессору с номером a, $a \in \{0,1,...,k\}$;

F[p] – массив ($p = 0,1,...,n$) для хранения значений функций $f_{jl}(p)$;

x^0[j, l] – массив для хранения оптимального решения (используется только ЭМ с номером 0);

X[p] – массив для хранения значений $x_{jl}(p)$ (при фиксированных j, l используется только ЭМ с номером 0);

x[j, l, p] – массив для хранения значений функций $x_{jl}(p)$, $p \in \{0,1,...,[n/k]-1\}$;

$\mathbf{r_{jl}}$(x) – оператор-функция, возвращающий значение функции $r_{jl}(x)$;

max(j, l, p) – оператор-функция, возвращающий значение функции $f_{jl}(p)$. Приведем подробнее его описание:

```
max(j, l, p):
        p^1 := p – First;
        if j = n & l = L then { max := r_nL(0);  x[n, L, p^1] := 0;  x^1 := [p/n];
                for x := 1 to x^1 do { prob := r_nL(x);
```

```
                if max < prob then { max := prob;  x[n, L, p^1] := x; } }
            max(n, L, p) := max; }
        else { max := r_jl(0) + F[p];  x[j, l, p^1] := 0;  x^1 := [p/j];
            for x := 1 to x^1 do { prob := r_jl(x) + F[p - j * x];
    if max < prob then { max := prob;  x[j, l, p^1] := x; } }
            max(j, l, p) := max; }
```

Опишем схему алгоритма для ЭМ с номером 0, которая производит загрузку остальных ЭМ и получает окончательный результат:

```
Input;  p^0 := n;
for j := 1 to n do
    for l := 1 to L do {
        for p := 0 to n do  Take(X[p], 1);
        x^0[j, l] := X[p^0];  p^0 := p^0 - j * x^0[j, l]; }
Output;
```

Теперь приведем схему алгоритма для ЭМ с номером $a, a \in \{1,2,\ldots,k-1\}$:

```
First := (a - 1) * [n/k];  Last := a * [n/k] - 1;
for j := n downto 1 do
    for l := L downto 1 do {
        for p := Last downto First do F[p] := max(j, l, p);
        for p := 0 to First - 1 do Take(F[p], a + 1);
        for p := 0 to Last do Put(F[p], a - 1); }
for j := 1 to n do
    for l := 1 to L do {
        for p := First to Last do Put(x[j, l, p - First], a - 1);
        for p := Last + 1 to n do { Take(trans, a + 1);  Put(trans, a - 1); } }
```

Алгоритм работы ЭМ с номером k:

```
First := (k - 1) * [n/k];  Last := n;
for j := n downto 1 do
    for l := L downto 1 do {
        for p := Last downto First do F[p] := max(j, l, p); }
for j := 1 to n do
    for l := 1 to L do
        for p := First to Last do Put(x[j, l, p], k - 1);
```

4. **Реализация алгоритма.** Описанный выше алгоритм поиска стохастически оптимального разбиения вычислительной системы на подсистемы был промоделирован на языке 3L Parallel C. Вычисления проводились на транспьютерной ВС Transtech, состоящей из транспьютеров Inmos T800 20MHz с оперативной памятью 2048 Кбайт. Время (в секундах) работы алгоритмов в зависимости от числа транспьютеров приведено в таблице.

Особенность описанного подхода заключается в том, что для расчета разбиения множества ЭМ ресурсы ВС используются только один раз для про-

n	А	АП6	АП8
16	~0	~0	~0
32	2	~0	~0
64	5	2	1
128	25	9	7
256	101	42	36

Обозначения: А – последовательный алгоритм с одним обратным и одним прямым ходом; АП6, АП8 – параллельные алгоритмы для шести и восьми транспьютеров

должительного промежутка времени. Рассмотренный выше параллельный алгоритм может быть применен в операционных системах для планирования обработки потока задач в вычислительных системах.

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. Sixth Internat. Workshop on Distributed Data Process. Novosibirsk: RAS Siberian Branch Publ., 1998. P. 65.
4. **Хорошевский В. Г.** Исследование функционирования однородных вычислительных систем: Дис. на соиск. учен. степени д-ра техн. наук. Новосибирск: ИМ СО АН СССР, 1972.
5. **Береснев В. Л., Дементьев В. Т.** Исследование операций. Новосибирск: НГУ, 1979.
6. **Вентцель Е. С.** Элементы динамического программирования. М.: Наука, 1964.

Институт физики полупроводников СО РАН,
E-mail: khor@isp.nsc.ru

Поступила в редакцию
1 апреля 1999 г.