РОССИЙСКАЯ АКАДЕМИЯ НАУК
СИБИРСКОЕ ОТДЕЛЕНИЕ
АВТОМЕТРИЯ
№ 1 2001

ВЫЧИСЛИТЕЛЬНЫЕ СЕТИ И СИСТЕМЫ ПЕРЕДАЧИ ДАННЫХ

УДК 681.3

В. А. Воробьев
(Архангельск)

ОБ ЭФФЕКТИВНОСТИ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ

Поставлен вопрос о существовании алгоритмических ограничений эффективности параллельных вычислений, показана неэффективность всех алгоритмов, основанных на сдваивании, и предложен эффективный подход к решению таких задач. Рассмотрена эффективность параллельных вычислений с учетом потерь на коммуникации между ветвями программы при различных системах коммутации каналов между процессорами (кроссбар, параллельная шина, дельта-сеть, регулярный канал однородной вычислительной системы) и различных видах обменов (разделение времени, параллельный обмен с разделением пространства коммутатора или регулярного канала, локальный обмен по регулярному каналу с соседями в ограниченной окрестности). Показано, что асимптотически эффективны только вычисления на однородной вычислительной системе с локальными обменами.

Постановка проблемы. В 1980–1990 гг. во всем мире наблюдалась [1] экспансия параллельных ЭВМ – массово-параллельных процессоров (МПП), причем потребность в сверхскоростных вычислениях на два-три порядка опережала существующие возможности суперЭВМ [2]. Между тем вопрос о том, какие именно архитектуры и параллельные алгоритмы обеспечивают требуемый рост производительности МПП, не получил еще окончательного решения. С одной стороны, известны утверждения [3], что любой «достаточно короткий» алгоритм для обработки «достаточно большого объема данных» эффективно распараллеливается. С другой стороны, [4] вслед за Амдалем [5] утверждается, что «массовый параллелизм эффективен, если только доля параллельных вычислений растет с ростом размера задачи».

Указанная проблема рассмотрена в двух аспектах: алгоритмическом и архитектурном. Во-первых, дается ответ на вопрос, имеются ли принципиальные алгоритмические ограничения на рост производительности за счет параллельного исполнения. Во-вторых, рассмотрены причины снижения производительности для различных архитектур параллельных машин, особенно для разных типов коммутаторов, с учетом задержек в длинных линиях передач между процессорами.

Алгоритмические пределы роста производительности. Существуют ли алгоритмические ограничения роста скорости вычислений? Для вектор-

ных машин такие ограничения существуют [6]. Пусть $v(N)$ – число операций векторного алгоритма; $s(N)$ – число операций наилучшего последовательного алгоритма решения той же задачи порядка N. Говорят [6], что векторный алгоритм согласован с наилучшим последовательным, если при исполнении его на векторной машине отношение $v(N)/s(N)$ ограничено при $N \to \infty$. Векторный алгоритм не согласован с наилучшим последовательным, если $v(N)/s(N) \to \infty$ при $N \to \infty$. Согласованные алгоритмы эффективны на векторных машинах при любой степени векторизации N, а несогласованные алгоритмы имеют верхний предел степени векторизации, после чего они уступают последовательным.

Для параллельных ЭВМ этот вопрос связан с давней дискуссией об их эффективности. Известен, например, закон Амдаля [5], который лежит в основе многих спекулятивных рассуждений на эту тему. Так, в [7] параллелизм рассматривается как революция в вычислениях, дается обзор архитектур векторных, потоковых и параллельных ЭВМ, а закон Амдаля приводится в следующей форме:

$$\text{ускорение вычислений } S = t_{\text{пос}} / t_{\text{пар}} = (\alpha_1 + \alpha_2 / N)^{-1},$$

где α_1 – доля последовательных вычислений; α_2 – доля параллельных вычислений; N – число элементарных машин (ЭМ). В такой форме при $N \to \infty$ ускорение стремится к $1/\alpha_1$ и параллелизм неэффективен. Противоположный вывод получается, если взять за единицу время параллельного решения задачи, а за $(\alpha_1 + N\alpha_2)$ – время последовательного решения. Тогда ускорение растет с ростом параллелизма. Аналогично в [4] отмечается, что с ростом размера задачи доля последовательной работы падает, если это некоторые фиксированные накладные расходы, и, следовательно, закон Амдаля дает в этом случае заниженную оценку производительности параллельной машины.

Для выхода из этой ситуации следует рассматривать более подробную модель параллельных вычислений [6, 8] с учетом затрат на подготовку данных. Даже рассмотрение конкретных параллельных систем не гарантирует безошибочных общих выводов. Например, в [3] для системы типа ОКМД (Connection Machine) рассмотрены некоторые алгоритмы с логарифмической оценкой времени $O(\log_2 N)$:

– суммирование элементов массива длины N методом сдваивания,

– вычисление частичных сумм массива методом рекурсивного сдваивания,

– поразрядная сортировка N чисел, лексический анализ слова,

– обработка списков длины N,

– сегментация изображений.

Утверждается, что для алгоритмов, у которых отношение объема обрабатываемых данных к длине программы достаточно велико, всегда возможно эффективное распараллеливание. Найти опровергающий пример не удалось.

Рассмотрим, однако, первые два алгоритма из [3]. На рис. 1 показаны схемы обменов на различных этапах вычисления: *а* – суммы N чисел методом сдваивания; *b* – частичных сумм N чисел методом рекурсивного сдваивания на линейной однородной вычислительной системе (ОВС) из N ЭМ, каждая из которых имеет два соседа: слева и справа. Заштрихованы процессоры, выполняющие сложение входных величин.

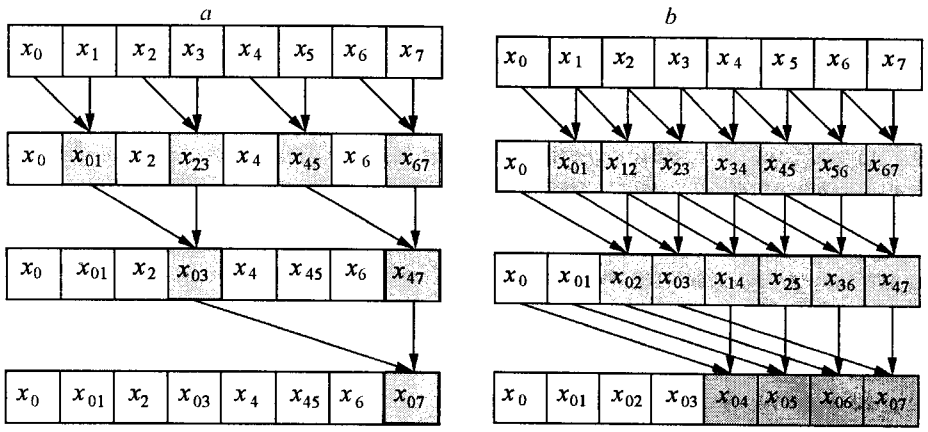

Рис. 1. Параллельное суммирование методом сдваивания (*a*) и параллельное вычисление частичных сумм (*b*): x_{ij} – сумма от i до j

Известно [6], что метод сдваивания согласован с наилучшим последовательным алгоритмом, поскольку имеет место

$$\mathrm{v}(N)/s(N) = (N-1)/(N-1) = 1.$$

Метод рекурсивного сдваивания, напротив, не согласован с наилучшим последовательным алгоритмом, поскольку для него, как не трудно видеть из рисунка,

$$\mathrm{v}(N)/s(N) = (N\log_2 N - (N-1))/(N-1) \to \infty \quad \text{при } N \to \infty.$$

Мерами параллельности алгоритмов являются степень параллелизма p – число операций, выполняемых одновременно (ширина в терминах [9]), и средняя степень параллелизма r – отношение $\mathrm{v}(N)/l$ общего числа его операций $\mathrm{v}(N)$ к числу l его этапов (в терминах [9] l – длина параллельного алгоритма). Для рассматриваемых алгоритмов величина r определяется очевидным образом:

– сдваивание $r_{\text{сдв}} = (N-1)/\log_2 N \ll N$ при $N \to \infty$,

– рекурсивное сдваивание $r_{\text{рсд}} = (N\log_2 N - (N-1))/\log_2 N = N - r_{\text{сдв}} \to \infty$ при $N \to \infty$.

Итак, $r_{\text{рсд}} \gg r_{\text{сдв}}$ и обе величины имеют порядок N, что позволяет надеяться на эффективную реализацию обоих методов на параллельной ЭВМ. Однако кроме вычислений, требуется еще доставка данных к месту вычислений по всей линейке процессоров. Пусть сложение занимает время t, а однократная доставка данных к процессору – время βt. Варьируя β, можно учитывать свойства систем:

при $\beta = 0$ время на доставку данных вообще не учитывается;

при $\beta = \text{const}$ в системе имеется неблокирующий коммутатор типа кроссбар;

при $\beta = O(\log_2 N)$, где N – число процессоров, в системе имеется неблокирующий многоступенчатый коммутатор типа дельта-сеть с числом ступеней, равным $\log_2 N$;

при $\beta = f(N$, алгоритм) рассматривается ОВС, в которой ЭМ связаны в систему регулярным каналом (РК) с топологией линейки, кольца, решетки или тора.

По определению [6] ускорение параллельного алгоритма по сравнению с наилучшим последовательным $S = t_{\text{пос min}} / t_{\text{пар}}$. В нашем случае $t_{\text{пос min}} = (N-1)t$. Воспользуемся прямым подсчетом величины $t_{\text{пар}}$. Из рис. 1 видно, что оба алгоритма сдваивания затрачивают на вычисления одинаковое время $t_{\text{пар}} = t \log_2 N$, пропорциональное числу этапов. Большее число операций алгоритма рекурсивного сдваивания компенсируется большей степенью параллелизма. Точно так же, если в параллельной ЭВМ используется неблокирующий коммутатор, то на доставку данных к процессорам оба алгоритма тратят одинаковое время, пропорциональное числу этапов: $t_{\text{д}} = \beta t \log_2 N$. Итак, $t_{\text{пар}} = t(1+\beta)\log_2 N$.

Рассмотрим различные системы, варьируя β и полагая, что $N \to \infty$. Эффективность реализации алгоритмов на различных параллельных системах оценивается величиной $E = S/N\ (0 \le E \le 1)$, которая позволяет учесть равномерность загрузки процессоров. Алгоритм эффективен на заданной вычислительной системе, если $\lim E \neq 0$ при $N \to \infty$.

В таблице содержатся результаты соответствующих расчетов, упрощенные за счет отбрасывания слагаемых типа $1, -1, c, -c$, малых по сравнению с $N \to \infty$. Величина $t_{\text{д}}$ для регулярного канала линейной ОВС получается прямым подсчетом в соответствии с рис. 1, а β в этом случае является средней по этапам вычислений.

Итак, алгоритмы сдваивания неэффективны даже теоретически, т. е. без учета структуры связей и затрат на передачу данных в параллельной ЭВМ. Они дают ускорение вычислений только в системах с централизованными неблокирующими коммутаторами, а в распределенных системах, какими являются ОВС, алгоритмы сдваивания не ускоряют счет и даже медленнее последовательных при $c > 1$. Вместе с ними неэффективны все алгоритмы с подобной организацией вычислений и обменов данными (например, поиск максимума или минимума получается по схеме рис. 1, *а* заменой сложения сравнением). Вывод, сделанный авторами [3] относительно исследованных ими алгоритмов, основан на недоразумении: ускорение вычислений и эффективность не одно и то же, а результаты, относящиеся к Connection Machi-

Ускорение и эффективность алгоритмов сдваивания для различных систем связи в параллельных ЭВМ

Тип системы связи	β	$t_{\text{д}}$	Оценка $t_{\text{пар}}$	Оценка S	$\lim\limits_{N\to\infty} S$	$\lim\limits_{N\to\infty} E$
Теория	0	0	$t\log_2 N$	$N/\log_2 N$	∞	0
Кроссбар	c_0	$tc\log_2 N$	$tc\log_2 N$	$N/c\log_2 N$	∞	0
Дельта-сеть	$c_0 \log_2 N$	$tc(\log_2 N)^2$	$tc(\log_2 N)^2$	$N/c(\log_2 N)^2$	∞	0
РК ОВС	$\dfrac{c_0(N-1)}{\log_2 N}$	$tc(N-1)$	$t\log_2 N + cN$	$N/(\log_2 N + cN)$	$1/c$	0

Примечание: c_0 и c – технические константы, близкие к 1.

ne, несправедливы для перспективных систем с массовым параллелизмом и распределенным коммутатором.

В частности, задачу суммирования массива совсем не обязательно решать методом сдваивания. При массовой обработке данных упомянутая задача часто решается многократно или для двумерных массивов. Если в кольцевой ОВС столбцы матрицы размером $N \times N$ расположены в памяти отдельных ЭМ, то суммирование всех N столбцов происходит параллельно по наилучшему последовательному алгоритму за время $t(N-1)$ с ускорением N и эффективностью, приблизительно равной 1. Для суммирования строк вовсе не обязательно доставлять данные к месту суммирования. Проще и быстрее доставлять суммы туда, где есть данные для дальнейшего суммирования. Идея этого алгоритма представлена на рис. 2. Он суммирует параллельно N строк за время Nt с ускорением $(N-1)$ и эффективностью $(N-1)/N \to 1$ при $N \to \infty$. Каждая ЭМ содержит в локальной памяти один столбец длиной N и на индекс-регистре адреса – номер строки, сумму которой она накапливает на текущем этапе. Символ s_{ij}^{k} обозначает сумму от i-го до j-го элемента k-й строки, перечисляемых циклически слева направо, например, $s_{42}^{3} = x_{34} + x_{31} + x_{32}$. Вектор-сумма s_{ij} на каждом этапе вычислений циклически сдвигается на один элемент вправо, и компонента s_{ij}^{k} суммируется с очередным элементом k-й строки (на рисунке элемент затенен). Очередные значения индекс-регистра получаются тем же циклическим сдвигом вправо или вычисляются как обычно, если это быстрее.

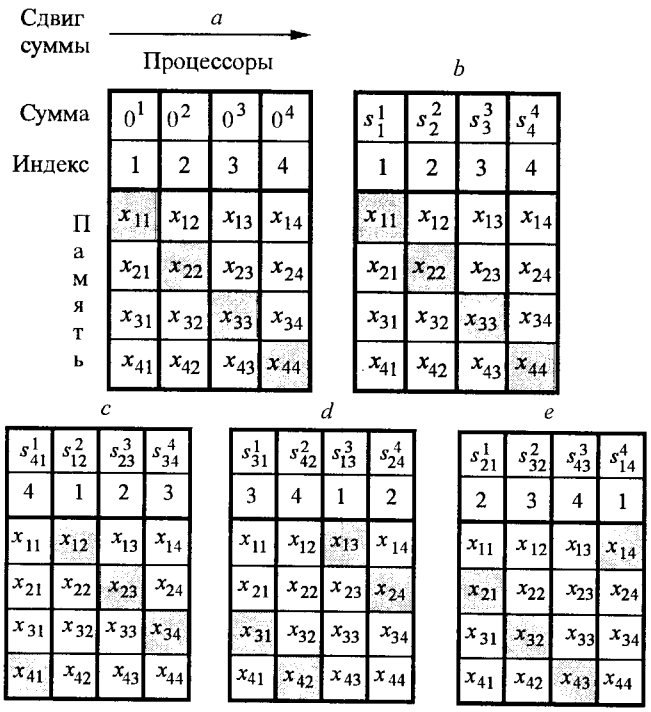

Рис. 2. Локально-параллельное суммирование строк $(N \times N)$-матрицы на кольцевой ОВС, имеющей N элементарных машин и обмен циклическим сдвигом $(N = 4)$: a – начальное состояние, $b-e$ – этапы 1–4 соответственно

Архитектурные пределы роста производительности. Рассмотрим общую модель эффективности параллельных вычислений. Согласно [6],

$$S = t_{\text{пос}}/t_{\text{пар}} = t_{\text{пос}}/[(\alpha_1 + \alpha_2/k + \alpha_3/N)t_{\text{пос}} + t_{\text{д}}],$$

где α_1 – доля операций, выполняемых одним процессором; α_2 – доля операций, выполняемых со средним ускорением k; α_3 – доля операций, выполняемых с максимальным ускорением N; $t_{\text{д}}$ – время подготовки данных, т. е. задержка, необходимая для размещения данных в тех местах системы, где происходит их обработка.

Закон Амдаля – Уэра можно получить, если положить $t_{\text{д}} = 0, \alpha_1 = \alpha_2 = \alpha$, $\alpha_3 = 1 - \alpha$, где α – доля последовательных операций. Тогда

$$S = [\alpha + (1-\alpha)/N]^{-1} = N/[\alpha(N-1)+1] \to 1/\alpha \text{ при } N \to \infty.$$

Это соотношение слишком оптимистично, так как не учитывает роста величины $t_{\text{д}}$ с ростом размера системы. Воспользуемся более реальной моделью:

$$S = [\alpha + (1-\alpha)/N + t_{\text{д}}/t_{\text{пос}}]^{-1}.$$

Варьируя формой представления $t_{\text{д}}$, можно описывать разные типы систем:

$t_{\text{дд}} = l \log_2 N f(t_{\text{пос}}, N)$ для дельта-сетей,

$t_{\text{д1}} = lNf(t_{\text{пос}}, N)$ для одномерных РК и шин,

$t_{\text{д2}} = lN^{1/2} f(t_{\text{пос}}, N)$ для двумерных РК и кроссбар,

где l – некоторая константа с размерностью времени.

Выражение для $t_{\text{д}}$ состоит из двух сомножителей: времени, затраченного передаваемой информацией на преодоление расстояния в системе и зависящего от типа и размера системы; числа $f(t_{\text{пос}}, N)$ последовательных актов обмена одним машинным словом (числом, символом и прочими единицами данных), зависящим от программы.

Предполагается, что:

1. Расстояние, на которое передается информация в системе, пропорционально числу процессоров на одном из измерений структуры или числу ступеней коммутатора.

2. В сети связи нет задержек из-за конфликтов между коммутациями и ограничений на вид коммутаций. Для неблокирующих коммутаторов эта идеализация не кажется чрезмерной, хотя и требует специальной организации параллельной программы. Для РК ОВС – это серьезное ограничение, улучшающее оценку эффективности параллельных вычислений.

3. Число обменов пропорционально временной сложности последовательного алгоритма (или числу витков внутреннего цикла), т. е. $f(t_{\text{пос}}, N) = ct_{\text{пос}} h(N)$, где $h(N)$ зависит от организации обменов. Большее количество обменов, очевидно, невозможно, а меньшее свидетельствует о малой сложности задачи или о неполном распараллеливании. Сложные задачи линейной алгебры и математической физики соответствуют нашему предположению.

4. В последовательной ЭВМ данные доставляются в процессор за постоянное время, независимое от размера памяти. Это серьезное допущение дает последовательной ЭВМ некоторое преимущество и обосновано тем, что в современных системах приняты специальные меры ускорения доставки дан-

ных: расщепление, кэш, сверхоперативное ЗУ на регистрах, векторные регистры и процессоры.

Далее будут рассмотрены популярные системы связи между процессорами: дельта-сеть и регулярный канал. Для кроссбар, как будет показано ниже, подходят оценки расстояний двумерной решетки, а для параллельной шины – одномерной.

Рассмотрим случай, когда взаимодействия происходят в режиме разделения времени, что автоматически исключает конфликты. При этих условиях $h(N)=1$ и, учитывая наши предположения, имеем с точностью до множителя l (где учтена и константа c):

$S_д = 1/[\alpha + (1-\alpha)/N + l\log_2 N] \to 0$ как $1/\log_2 N$ при $N \to \infty$ в дельта-сети;

$S_1 = 1/[\alpha + (1-\alpha)/N + lN] \to 0$ как $1/N$ при $N \to \infty$ в одномерном РК;

$S_2 = 1/[\alpha + (1-\alpha)/N + lN^{1/2}] \to 0$ как $1/N^{1/2}$ при $N \to \infty$ в двумерном РК.

Характер стремления к 0 получается при $\alpha = 0$, т. е. при полном параллелизме.

Итак, в данном случае параллельные вычисления могут быть (и, скорее всего, будут) асимптотически медленнее последовательных. Для того чтобы эти вычисления были эффективны, необходимо обеспечить такой вид функции $f(t_{пос}, N)$, который отражал бы и технические возможности коммутаторов параллельных суперЭВМ, и реальные классы решаемых задач, но в то же время ограничивал бы рост затрат на взаимодействия.

Будем считать, что распараллеливаются не только вычисления, но и обмены данными, а именно $f(t_{пос}, N) = O(t_{пос})/N$, т. е. все N (или $N^{1/2}$ в пределах одной строки или столбца) процессоров одновременно передают и принимают информацию в каждом сеансе связи. При этих условиях ускорение имеет вид:

$S_д = N/[N\alpha + (1-\alpha) + l\log_2 N] \to \infty$ как $N/\log_2 N$ при $N \to \infty$ для дельта-сети;

$S_1 = [\alpha + (1-\alpha)/N + l]^{-1} \to (\alpha + l)^{-1}$ при $N \to \infty$ для одно- и двумерного РК;

$S_2 = [\alpha + (1-\alpha)/N + lN^{-1/2}]^{-1} \to 1/\alpha$ при $N \to \infty$ для двумерного РК ОВС.

Итак, даже для неблокирующих дельта-сетей эффективность E параллельных вычислений стремится к нулю как $1/\log_2 N$ и при полном параллелизме, когда $\alpha = 0$ и все коммутации одновременны.

Заметим, что до сих пор мы считали задержку в дельта-сети логарифмической, пренебрегая, как это часто делается, длиной линий связи. Это соответствовало подходу авторов [3] и не влияло на результат. Неэффективен был сам алгоритм сдваивания. В анализе эффективности архитектур вычислительных систем мы обязаны учитывать длину линий связи, так как при современной технологии именно она определяет быстродействие. Более того, тактовые частоты современных микропроцессоров столь высоки, что классическая теория вычислений непригодна для описания процессов, происходящих в вычислительных системах, состоящих из большого числа этих элементов. В [10] для таких систем, выполненных на неразрезных СБИС, предложено название «релятивистская ЭВМ», поскольку их размер превышает величину $c\tau$ – расстояние, на которое может воздействовать сигнал, распространяющийся со скоростью света, за тактовое время τ (тем более, что групповая

скорость электрического сигнала в линии связи гораздо меньше c). Далее ОВС – релятивистские ЭВМ.

С учетом последнего замечания следует признать полученную низкую оценку ускорения и эффективности параллельных вычислений на системах с дельта-сетью завышенной. Правильнее считать, что задержка в дельта-сети на плоскости равна

$$t_{\text{дд}} = l(N^{1/2} + \log_2 N) f(t_{\text{пос}}, N),$$

что хуже, чем для двумерного регулярного канала с обменами в пределах одной строки или столбца.

Для одномерных ОВС ускорение программ при $\alpha = 0$ ограничено некоторой технической величиной $1/l$ и остается константой независимо от N. Эффективность $E = 1/lN$ стремится к нулю с ростом N.

Для двумерных ОВС (и, тем более, для систем с дельта-сетью) ускорение вычислений определяется долей последовательных операций α и при $\alpha = 0$ растет как $N^{1/2}/l$, а эффективность стремится к нулю: $E = 1/lN^{1/2}$.

Эти соотношения не дают высокой эффективности параллельных вычислений, и причина теперь в том, что вид взаимодействий изначально не был ограничен. Более того, трудно себе представить параллельный обмен информацией всех ЭМ через регулярный канал, если коммутации не упорядочены специальным образом.

Рассмотрим системы и параллельные программы с локальными взаимодействиями, т. е. такими, что расстояние между передатчиком и приемником ограничено и не зависит от размера системы N. Таковы, например, сдвиги информации на ограниченное расстояние k вдоль кольца в одномерной ОВС или вдоль строк (столбцов) в двумерной ОВС. В этом случае можно полагать $t_{\text{д}} = lkf(t_{\text{пос}}, N)$, где $f(t_{\text{пос}}, N)$ зависит от типа системы и организации обменов. Итак,

$f(t_{\text{пос}}, N) = ct_{\text{пос}}/N$ и $t_{\text{д}} = lkt_{\text{пос}}/N$ для одномерных и двумерных РК,

$f(t_{\text{пос}}, N) = ct_{\text{пос}}/N^{1/2}$ и $t_{\text{д}} = lkt_{\text{пос}}/N^{1/2}$ для двумерных РК при сдвигах в пределах одной строки (столбца).

Для систем с централизованным коммутатором (кроссбар, шина или дельта-сеть) условие локальности выполнить невозможно, поэтому они в дальнейшем не рассматриваются.

Подставляя принятые выражения $t_{\text{д}}$ в закон Амдаля – Уэра, имеем

$S_1 = N/[\alpha(N-1)+1+lk] \cong N/[\alpha N + lk] \to 1/\alpha$ при $N \to \infty$,

$S_2 = N/[\alpha(N-1)+1+lkN^{1/2}] \cong N/[\alpha N + lkN^{1/2}] \to 1/\alpha$ при $N \to \infty$.

Итак, ускорение параллельных вычислений с локальными взаимодействиями зависит только от доли последовательных вычислений. В частности, при $\alpha = 0$ имеем

$S_1 = N/[1+lk]$, $\quad E = 1/[1+lk] = \text{const}$,

$S_2 = N/[1+lkN^{1/2}] \cong N^{1/2}/lk$, $\quad E = 1/lkN^{1/2} \to 0$ при $N \to \infty$.

Вывод ясен: среди рассмотренных альтернатив асимптотически эффективны только вычисления с параллельными локальными взаимодействиями через распределенный регулярный канал сложности $O(N)$ и только они.

Что касается заключительного вывода [3], то, как показано, тезис о возможности эффективного распараллеливания любого короткого алгоритма с

большим объемом данных неверен. Однако это не препятствие для развития параллельных систем. Мы будем придерживаться следующего давно известного [9] тезиса: для любой достаточно сложной задачи можно найти параллельный алгоритм, эффективно реализуемый на релятивистской ОВС.

Такие алгоритмы для задач линейной алгебры и математической физики можно найти, например, в [11].

СПИСОК ЛИТЕРАТУРЫ

1. **ЭВМ** с массовым параллелизмом завоевывают сферу бизнеса // Computer World-Moskow. 1994. N 49. P. 53, 55.
2. **Бобровский С.** Призрак ядерной войны // PC Week. Компьютерная неделя. 1996. № 49(73). С. 1.
3. **Hillis W. D., Steele G. L.** Data parallel algorithms // Commun. ACM. 1986. **29**, N 12. P. 1170.
4. **Gustafson J. L.** Reevaluation Amdahl's law // Commun ACM. 1988. **31**, N 5. P. 532.
5. **Amdahl G.** The validity of the single processor approach to achieving large-scale computing capabilities // Proc. AFIPS Spring Joint Computer Conf. (Atlantic City, NJ, Apr. 18–20, 1967). Reston, VA: AFIPS Press, 1967. P. 483.
6. **Ортега Дж.** Введение в параллельные и векторные методы решения линейных систем. М.: Мир, 1991.
7. **Gallant J.** Parallel processing ushers in a revolution in computing // EDN. 1988. **33**, N 18. P. 86, 89–94, 96, 98, 100.
8. **Buzbee B.** Parallel processing makes tough demands // Comput. Des. 1984. **23**, N 10. P. 137.
9. **Евреинов Э. В., Косарев Ю. Г.** Однородные универсальные вычислительные системы высокой производительности. Новосибирск: Наука, 1966.
10. **Воробьев В. А., Лаходынова Н. В.** Пределы надежности однородных вычислительных систем // Экспертные системы и распознавание образов. Вычислительные системы. Новосибирск: ИМ СО АН СССР, 1988. Вып. 126. С. 122.
11. **Мишин А. И., Леус В. А.** Асинхронно-локальные вычислительные системы и среды. Новосибирск: ИМ СО АН СССР, 1991.

Поморский государственный университет им. М. В. Ломоносова,
E-mail: vorob@sanet.ru

Поступила в редакцию
2 марта 1998 г.