РОССИЙСКАЯ АКАДЕМИЯ НАУК
СИБИРСКОЕ ОТДЕЛЕНИЕ
А В Т О М Е Т Р И Я
№ 2 1998

УДК 681.3.06 : 681.323

В. А. Мелентьев
(Новосибирск)

АНАЛИЗ ЭФФЕКТИВНОСТИ И ОПТИМИЗАЦИЯ ПАРАЛЛЕЛЬНЫХ ПРОГРАММ ДЛЯ РАСПРЕДЕЛЕННЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ*

Предложена модель оценки эффективности реализации параллельных программ. Выделена область эффективной реализации параллельных программ, связывающая коэффициент ускорения с числом элементарных машин в подсистеме. Осуществлена декомпозиция временных затрат на реализацию параллельного алгоритма, проведен анализ этих затрат, даны рекомендации по их оптимизации с целью повышения эффективности. Приведен пример анализа программы перемножения двух матриц, реализованной на транспьютерной вычислительной системе.

1. Эффективность программной реализации параллельного алгоритма. Оценку эффективности программной реализации параллельного алгоритма будем производить исходя из того, что последний должен быть выполнен с требуемым качеством, в заданном временном интервале и с минимальными потерями используемых вычислительных мощностей (при максимальном использовании задействованных ресурсов):

$$E_n = q_n h_n k_n / n.$$

Здесь индекс n отождествлен с числом элементарных машин (ЭМ) в подсистеме, реализующей алгоритм; E — эффективность реализации; q — коэффициент качества; h — коэффициент актуальности; k — коэффициент ускорения.

Коэффициент качества q_n характеризует адекватность результата обработки требованиям пользователя и исходным данным. Значение q_n равно единице, если на любом наборе исходных данных (из множества допускаемых по ТЗ) результат обработки этих данных на подсистеме из n ЭМ полностью соответствует требованиям пользователя. К этим требованиям могут быть отнесены, например, достоверность идентификации изображения $q_n = \frac{\text{достоверность } (n)}{\text{достоверность (ТЗ)}}$ или вероятность селекции объекта из множества заданных $q_n = \frac{\text{вероятность селекции } (n)}{\text{вероятность селекции (ТЗ)}}$.

Коэффициент актуальности $h_n = t^*/t_n$. Здесь t^* — время, превышение которого означает потерю актуальности результата; t_n — время реализации алгоритма на подсистеме из n ЭМ. Очевидно, что актуальность реализации алгоритма (режим реального времени) достигается при $t_n \le t^*$, или $h_n \ge 1$.

Коэффициент ускорения $k_n = t_1/t_n$. Здесь t_1 — время реализации алгоритма на вырожденной подсистеме ($n = 1$). Отношение коэффициента ускорения k_n к числу n ЭМ в подсистеме показывает, насколько эффективно

* Работа выполнена при поддержке Российского фонда фундаментальных исследований (проект № 97-01-00883).

используются задействованные вычислительные ресурсы, т. е. характеризует эффективность их использования.

$$E_n = h_n k_n / n = \frac{t^* t_1}{t_n t_n n} = \frac{k_n^2 t^*}{n t_1} = \frac{h_n^2 t_1}{n t^*}.$$

Обозначив отношение t_1/t^* как требуемое значение коэффициента ускорения k^*, получим

$$E_n = \frac{k_n^2}{n k^*} = \frac{h_n^2 k^*}{n}. \qquad (1.1)$$

Очевидно, что увеличение ранга задачи (размера подсистемы) имеет целью достижение режима актуальности, т. е. увеличение коэффициента ускорения от значения $k_1 = 1$ при $n = 1$ до значения $k_n \geq k^*$. Из (1.1) видно, что при реализации параллельного алгоритма на одной ЭМ ($n = 1$) эффективность принимает значение $E_1 = 1/k^*$. Тогда, для того чтобы эффективность параллельной реализации алгоритма на подсистеме из n ЭМ была не ниже эффективности реализации этого алгоритма на одной ЭМ, необходимо, чтобы значение коэффициента ускорения, полученное при параллельной реализации, было не менее корня квадратного из числа ЭМ в подсистеме, т. е.

$$E_n \geq E_1, \quad \text{если } k_n \geq \sqrt{n}. \qquad (1.2)$$

Итак, существуют, по крайней мере, четыре области изменения коэффициента ускорения, определяющие эффективность реализации параллельного алгоритма (рис. 1):

1) $k_n \leq 1$ — неэффективная реализация алгоритма. Увеличение числа ЭМ не приводит к увеличению коэффициента ускорения. Следовательно, подобная параллельная реализация алгоритма не имеет смысла. Эффективность такой реализации асимптотически стремится к нулю с коэффициентом пропорциональности n^{-1}, т. е. $E_n \leq E_1/n$;

2) $1 < k_n \leq \sqrt{n}$ — пониженная эффективность параллельной реализации. Результатом параллельной реализации является некоторое ускорение, но эф-

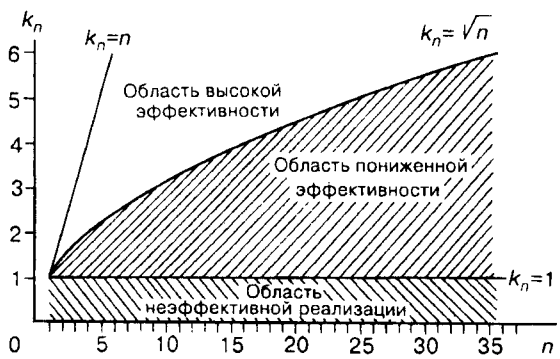

Рис. 1

фективность такой реализации не настолько высока, чтобы превысить значение эффективности последовательной реализации: $E_n \le E_1$;

3) $\sqrt{n} < k_n < n$ — высокая эффективность параллельной реализации. В этом случае ускорение достаточно высоко для того, чтобы значение эффективности превышало эффективность последовательной реализации: $E_1 < E_n < nE_1$;

4) $k_n \ge n$ — очень высокая эффективность, практически недостижимая без изменения алгоритма, здесь $E_n \ge nE_1$.

2. Анализ реализации параллельного алгоритма на распределенных ВС. Полное время t_1 реализации алгоритма на одной ЭМ ($n = 1$) разобьем на две составляющие: $t_1 = t_1' + t_1''$. Здесь t_1' — время «чистой» обработки данных в алгоритме, не включающее издержки, обусловленные использованием операционной системы (ОС), такие как работа с подпрограммами, обработка прерываний, использование драйверов и т. п. Все накладные расходы, связанные с использованием ОС, учитывает время t_1''. Выполнение того же алгоритма на подсистеме из n ЭМ при условии однотипности всех ЭМ и равном разбиении алгоритма (или данных) по всем ЭМ подсистемы требует времени $t_n = t_n' + t_n''$. При этом $t_n' = t_1'/n$, а время t_n'' включает в себя, кроме указанных выше, также издержки, обусловленные некоторыми другими причинами, связанными с распараллеливанием. Это могут быть, например, издержки, вызванные необходимостью синхронизации процессов, обменами между ЭМ, увеличением суммарного объема обрабатываемой информации вследствие дублирования некоторой ее части (например, дублирование смежных фрагментов при распараллеливании данных) и т. п.

Отношение t_n'/t_n характеризует степень использования ЭМ для «чистой» обработки, обозначим его η_n. Тогда

$$t_1 = \eta_1 t_1 + (1 - \eta_1)t_1, \quad t_n = \eta_n t_n + (1 - \eta_n)t_n, \quad \eta_n t_n = \eta_1 t_1/n \text{ и } \eta_n = \eta_1 k_n/n.$$

Из последнего выражения следует, что при $k_n/n < 1$ с увеличением числа ЭМ в подсистеме доля накладных расходов возрастает и снижается значение коэффициента использования ЭМ для «чистой» обработки.

Предположим, что в результате распараллеливания часть накладных расходов $t_1'' = (1 - \eta_1)t_1$ сохраняется с коэффициентом α_1, а оставшаяся часть $(1 - \alpha_1)(1 - \eta_1)t_1$ равномерно распределена по всем ЭМ подсистемы. Отклонения от указанного распределения, а также увеличение издержек, вызванное необходимостью синхронизации процессов, задержками на реализацию сетевых протоколов, дублированием данных и т. п., будем учитывать некоторым приращением δ_n, причем из изложенного выше $\delta_1 = 0$. Тогда

$$\begin{aligned} t_n'' = (1 - \eta_n)t_n &= \alpha_1(1 - \eta_1)t_1 + (1 - \alpha_1)(1 - \eta_1)t_1/n + \delta_n = \\ &= (1 - \eta_1)t_1 \frac{1 + (n - 1)\alpha_1}{n} + \delta_n, \\ t_n = \frac{\eta_1 t_1}{n} + t_n'' &= \frac{1 + (n - 1)\alpha_1(1 - \eta_1)}{n} t_1 + \delta_n. \end{aligned} \tag{2.1}$$

Из (2.1) видно, что при увеличении размера подсистемы время $t_n = f(n)$ реализации алгоритма будет уменьшаться (а коэффициент ускорения возрастать) до такого n, пока уменьшение времени, обусловленное распараллеливанием, не будет компенсировано возрастанием δ_n, т. е.

$$\forall n \le N \; k_n > k_{n-1}, \; t_n < t_{n-1}, \text{ пока } \delta_n - \delta_{n-1} < \frac{1 - \alpha_1(1 - \eta_1)}{n(n - 1)} t_1. \tag{2.2}$$

Используя (2.1), получим значение коэффициента ускорения:

$$k_n = \left(\frac{1 + (n-1)\alpha_1(1-\eta_1)}{n} + \frac{\delta_n}{t_1}\right)^{-1}. \qquad (2.3)$$

Из выражения (2.3) видно, что условием максимального ускорения $k_n = n$ является сокращение до нуля издержек δ_n при обязательном равенстве нулю произведения $\alpha_1(1 - \eta_1)$. Последнее замечание не требует обязательного сокращения до нуля накладных расходов ($\eta_1 = 1$), что было бы нереально при использовании ОС, но предполагает необходимость максимально возможного сокращения их постоянной составляющей, обусловленной коэффициентом α_1. Это означает, что при распараллеливании алгоритма или данных необходимо минимизировать часть программы, не связанную непосредственно с обработкой данных (подготовительно-заключительные операции и т. п.).

Оценим потенциальные значения коэффициентов ускорения и эффективности, считая, что в результате оптимизации параллельной программы достигнуто нулевое значение части δ_n накладных расходов. Очевидно, что добиться подобного результата для накладных расходов, связанных с использованием операционной системы (т. е. $\eta_1 = 1$), не удастся, хотя эта часть также может быть подвергнута оптимизации путем разумного использования стандартных функций, оптимизации драйверов и т. п. Итак, примем $\delta_n = 0$.

Считаем, что верхнего предела изменения n не существует. Тогда

$$\lim_{n \to \infty} k_n = (\alpha_1(1 - \eta_1))^{-1}.$$

Если требуемое значение коэффициента ускорения k^* меньше рассчитанного предельного значения, то можно определить минимально необходимое для $k_n \geq k^*$ число ЭМ:

$$n^* = \left[\frac{k^*(1 - \alpha_1(1 - \eta_1))}{1 - k^*\alpha_1(1 - \eta_1)}\right],$$

здесь $[x]$ — целая часть числа x. Так как $\delta_n = 0$ и при достижении требуемого результата распараллеливания $t_n = t^*$, то

$$E_n = k_n/n = (1 + (n-1)\alpha_1(1 - \eta_1))^{-1}.$$

Определим минимальное значение времени реализации параллельного алгоритма при $\delta_n = 0$:

$$\lim_{n \to \infty} t_n = \alpha_1(1 - \eta_1)t_1.$$

Пусть в результате оптимизации программы мы смогли добиться уменьшения удельного числа операций на единицу данных от ν_1 до $\nu_1(\beta) = \beta\nu_1$, $\beta < 1$. Это влечет за собой соответствующее уменьшение значения «чистого» времени обработки данных от $t_1' = \eta_1 t_1$ до $t_1'(\beta) = \beta\eta_1 t_1$ и $t_n'(\beta) = \beta t_n'$. При этом $t_n(\beta) = \beta t_n' + (1 - \eta_1)t_1$ и

$$\lim_{n \to \infty} t_n(\beta) = \lim_{n \to \infty} t_n = \alpha_1(1 - \eta_1)t_1.$$

Таким образом, если требуемое значение времени $t^* \leq \alpha_1(1 - \eta_1)t_1$, то никакое уменьшение удельного числа операций на единицу данных не поможет в достижении режима актуальности.

Если же $\alpha_1(1-\eta_1)t_1 < t^*$, то оптимизация времени «чистой» обработки ($t'_1(\beta) = \beta t'_1$) дает возможность достижения режима актуальности при использовании меньшего числа ЭМ:

$$n^*(\beta) = \left[\frac{k^*(1-\eta_1(1-\beta)-\alpha_1(1-\eta_1))}{1-k^*\alpha_1(1-\eta_1)}\right].$$

Рассмотрим случай, когда функция $\delta_n = f(n)$ в некотором интервале изменения числа ЭМ от i до n может быть аппроксимирована уравнением прямой:

$$\delta_n = \delta_i + (n-i)\Delta, \qquad 2 \le i < n.$$

Продифференцировав выражение (2.1) по переменной n, получим значение верхнего предела числа ЭМ в подсистеме при условии сохранения линейного закона изменения δ_n до полученного предела:

$$\frac{\partial t_n}{\partial n} = -\frac{t_1(1-\alpha_1(1-\eta_1))}{n^2} + \Delta,$$

$$n^k_{\max} = \left[\sqrt{t_1(1-\alpha_1(1-\eta_1))/\Delta}\,\right].$$

Подставив значение $n^k_{\max}$ в (2.3), можно определить величину максимального ускорения для исследуемой программной реализации параллельного алгоритма.

Определим предельное значение числа ЭМ в подсистеме, при котором может быть достигнуто максимальное значение эффективности $n^E_{\max}$:

$$\frac{\partial E_n}{\partial n} = \left(\frac{k_n^2}{nk^*}\right)' = \frac{k_n}{nk^*}\left(2k'_n - \frac{k_n}{n}\right).$$

Таким образом, максимальное значение эффективности будет достигнуто при условии $2k'_n = k_n/n$ или

$$n^E_{\max} = \left[\left(b^2 + \frac{t_1 - \alpha_1(1-\eta_1)t_1}{3\Delta}\right)^{1/2} - b\right],$$

здесь $b = \dfrac{\alpha_1(1-\eta_1)t_1 + \delta_i - i\Delta}{6\Delta}$.

Итак, проведенный выше анализ показал:

1) величина $\alpha_1(1-\eta_1)$ характеризует качество параллельной программной системы (параллельная программа + операционная система ВС) и определяет пределы распараллеливания задачи вне зависимости от физической структуры ВС;

2) увеличение быстродействия программного комплекса более чем в $(\alpha_1(1-\eta_1))^{-1}$ раз, равно как и достижение режима актуальности при $\alpha_1(1-\eta_1)t_1 > t^*$, невозможно, если достижение цели связывается только с наращиванием числа ЭМ в подсистеме;

3) оптимизация «чистой» обработки не дает необходимого эффекта в достижении требуемого быстродействия, если требуемое время реализации $t^* < \alpha_1(1-\eta_1)t_1$;

4) требуемое быстродействие параллельной программной системы может быть достигнуто оптимизацией программы за счет снижения накладных расходов (увеличения η_1), а также их части, определяемой коэффициентом α_1, и за счет снижения издержек, определяемых величиной δ_n.

3. **Зависимость эффективности реализации параллельного алгоритма от объема обрабатываемых данных.** Проведем анализ зависимости изменения времени выполнения параллельного алгоритма от объема обрабатываемых

данных. Считаем при этом, что W_1 — минимально необходимый объем данных, достаточный для достижения требуемого значения коэффициента качества $q^* = 1$, а увеличение объема $W_m = mW_1$, $m > 1$, не ухудшает качества результата. Кроме того, изменение m осуществляем в пределах использования памяти одного уровня, т. е. без изменения ее быстродействия. К используемым в предшествующих разделах индексам (t_1, k_n и т. д.) добавим индекс m, указывающий на кратность изменения объема обрабатываемых данных (t_{1m}, k_{nm} и т. д.).

Исходя из предлагаемой модели,

$$t_{11} = \eta_{11}t_{11} + \alpha_{11}(1 - \eta_{11})t_{11} + (1 - \alpha_{11})(1 - \eta_{11})t_{11},$$

$$t_{1m} = \eta_{1m}t_{1m} + \alpha_{1m}(1 - \eta_{1m})t_{1m} + (1 - \alpha_{1m})(1 - \eta_{1m})t_{1m}.$$

Значение η_{1m} несложно выразить через объем обрабатываемых данных W_m, среднее значение числа операций на единицу данных (удельную трудоемкость) ν_m и среднее значение производительности ЭМ на используемом для обработки наборе операций ω_m:

$$\eta_{1m} = \frac{\nu_m W_m}{t_{1m}\omega_m}.$$

Далее считаем, что набор (смесь) операций для обработки единицы данных не зависит от объема обрабатываемого массива и $\omega_1 = \omega_m = \omega$ (как было оговорено выше, m изменяем в пределах памяти одного уровня). Тогда

$$\eta_{11} = \frac{\nu_1 W_1}{t_{11}\omega}, \qquad \eta_{1m} = \frac{\nu_m W_m}{t_{1m}\omega} = m\frac{\nu_m W_1}{t_{1m}\omega},$$

$$\eta_{1m}t_{1m} = m\frac{\nu_m W_1}{\omega} = m\frac{\nu_m}{\nu_1}\eta_{11}t_{11},$$

$$\eta_{1m} = m\frac{t_{11}}{t_{1m}}\frac{\nu_m}{\nu_1}\eta_{11}.$$

В случае перехода m за границы памяти одного уровня при определении η_{11} и η_{1m} следует учитывать соответствующие изменения производительности от ω_1 до ω_m:

$$\eta_{11} = \frac{\nu_1 W_1}{t_{11}\omega_1}, \qquad \eta_{1m} = \frac{mt_{11}}{t_{1m}}\frac{\nu_m}{\nu_1}\frac{\omega_1}{\omega_m}\eta_{11}.$$

Эксперимент с измерением времени реализации алгоритма на «пустом» массиве ($m = 0$) позволит определить значение коэффициента α:

$$t_{10} = \alpha_{11}(1 - \eta_{11})t_{11} = \alpha_{1m}(1 - \eta_{1m})t_{1m},$$

$$\alpha_{11} = \frac{t_{10}}{(1 - \eta_{11})t_{11}}, \qquad \alpha_{1m} = \frac{t_{10}}{(1 - \eta_{1m})t_{1m}} = \frac{\alpha_{11}(1 - \eta_{11})t_{11}}{(1 - \eta_{1m})t_{1m}}.$$

Итак, варьируя объемом обрабатываемых данных, не представляет сложности вычислить значения коэффициентов α, η и провести покомпонентный анализ времени реализации параллельного алгоритма при изменении размера подсистемы.

Существует возможность графического определения значения компоненты $\alpha_{11}(1 - \eta_{11})t_{11}$, не прибегая к эксперименту с «пустым» массивом. Для этого можно построить график функции $y = f(x)$, где $y = (1 - \eta_{1m})t_{1m} = t_{1m} - m\frac{\nu_m}{\nu_1}\eta_{11}t_{11}$, а $x = m$. Аппроксимируя функцию в интервале $x = [0, m]$, получим искомое значение при пересечении ею оси ординат. Если же ап-

проксимация указанной функции близка к линейной, то приближенное значение компоненты $\alpha_{11}(1 - \eta_{11})t_{11}$ может быть получено из выражения

$$\alpha_{11}(1 - \eta_{11})t_{11} = \frac{1}{m-1}\left(\left(1 + \left(\frac{\nu_m}{\nu_1} - 1\right)\eta_{11}\right)mt_{11} - t_{1m}\right).$$

4. Оптимизация параллельных вычислений. Предложенная выше модель декомпозиции временных затрат на реализацию параллельного алгоритма позволяет провести покомпонентный количественный анализ этих затрат с целью их последующей оптимизации.

Оптимизация параллельной реализации алгоритма носит комплексный характер и включает в себя: 1) оптимизацию прикладной программы, 2) оптимизацию структуры подсистемы, 3) оптимизацию системных средств.

Ниже приведены лишь общие рекомендации по оптимизации параллельных вычислений. Более детальное изложение материала с анализом конкретных алгоритмов и программных конструкций предполагается в последующих публикациях.

4.1. *Оптимизация прикладной программы.* Если анализ показал, что никакое наращивание вычислительной мощности не позволит достичь требуемого быстродействия при использовании исследуемого варианта программы, т. е. $t^* < \alpha_1(1 - \eta_1)t_1$, то, как показано выше, оптимизация только «чистой» обработки, т. е. снижение удельного числа операций на единицу данных, не приводит к искомому результату, так как абсолютная величина накладных расходов при этом остается прежней и значение компоненты $\alpha_1(1 - \eta_1)t_1$ сохраняется. Следовательно, в рассматриваемом случае необходимо прежде всего оптимизировать последовательную часть параллельного алгоритма или межмодульные интерфейсы при последовательном использовании нескольких параллельных модулей. При этом, возможно, придется отказаться от минимизации размера программы и памяти для хранения данных в пользу оптимизации временных характеристик. Не следует использовать подпрограммы или средства операционной системы, если их вызов и использование обходятся дороже выполнения. Среднее время вызова подпрограммы не должно превышать $(k^*)^{-1}$-ю часть от времени ее выполнения.

Если в результате оптимизации последовательной части параллельного алгоритма достигнуто условие $t^* > \alpha_1(1 - \eta_1)t_1$, то последующие действия, заключающиеся в минимизации удельного числа операций на единицу обрабатываемых данных или снижении степени дублирования данных, также внесут дополнительный вклад в повышение эффективности параллельной реализации алгоритма. При этом прежде всего следует обратить внимание на организацию итерационных структур и использование предикатов.

4.2. *Оптимизация структуры подсистемы.* Необходимость подобной оптимизации возникает, если при наращивании размера подсистемы величина δ_n растет настолько быстро, что требуемое быстродействие не может быть достигнуто.

В этом случае необходимо оптимизировать рассылку программы и размещение данных и частей алгоритма в пределах возможных изменений структуры ВС. Хронометраж вариантов покажет наиболее оптимальную структуру. С целью более точной оптимизации можно прибегнуть к приему искусственного «масштабирования» обменных операций путем их программного замедления.

Если изменение структуры не дает существенных результатов, достаточных для достижения требуемого быстродействия, то необходимо перейти к оптимизации системных средств в части операторов системных взаимодействий и алгоритмов маршрутизации или прибегнуть к использованию более быстродействующих каналов связи.

4.3. *Оптимизация системных средств.* Необходимость в такой оптимизации возникает, если предшествующие действия не обеспечили нужного эффекта. Оптимизация заключается в поиске тех компонент операционной системы, которые в основном определяют долю указанных выше накладных расходов, и в замене этих компонент на более эффективные. Это могут быть

Таблица 1

M	$m\frac{\nu_m}{\nu_1}$	n	t_n, с	η_n	t'_n, с	$\frac{(1-\eta_1)t_1}{n}$	δ_n, с	k_n	k_n/n, %	E_n
36	1	1	0,142	0,8	0,1136	0,0284		1	100	0,33
		4	0,058	0,49	0,0284	0,0076	0,0225	2,45	61,3	0,5
		9	0,042	0,30	0,0126	0,0032	0,0262	3,38	37,6	0,423
64	5,62	1	0,731	0,873	0,6383	0,0927		1	100	0,33
		4	0,246	0,65	0,1596	0,0232	0,0633	2,97	74,3	0,735
		9	0,152	0,467	0,071	0,010	0,071	4,81	53,4	0,857
100	21,4	1	2,676	0,91	2,435	0,241		1	100	0,33
		4	0,817	0,745	0,609	0,06	0,148	3,28	82	0,897
		9	0,461	0,588	0,271	0,026	0,164	5,80	64,4	1,246
128	45	1	5,52	0,925	5,106	0,414		1	100	0,33
		4	1,618	0,789	1,277	0,103	0,238	3,41	85,3	0,969
		9	0,864	0,657	0,567	0,046	0,251	6,39	71,0	1,512

протоколы межмашинных взаимодействий и соответствующие им драйверы, алгоритмы маршрутизации, средства контроля и обеспечения отказоустойчивости и т. п.

Если цель оптимизации, заключающаяся в упорядоченном, основанном на количественном анализе поиске возможностей минимизации каждой из компонент времени реализации параллельного алгоритма, не достигнута, т. е. $\lim\limits_{n \to n^k_{\max}} t_n > t^*$, то: 1) следует провести поиск и исследование менее трудоемких алгоритмов, 2) сменить техническую базу, 3) использовать специализированные вычислительные средства или аппаратно-реализованную логику управления.

5. Пример анализа эффективности. В качестве примера для демонстрации возможностей предлагаемой методики анализа использованы данные хронометража перемножения двух матриц $M \times M$ [1].

В табл. 1, 2 приведены результаты расчетов (полужирные цифры — исходные данные, взятые из [1]). В целях удобства сравнительной оценки эффективности требуемое значение коэффициента ускорения принято равным трем ($k^* = 3$) независимо от размеров перемножаемых матриц. Величина $\alpha_1(1 - \eta_1)t_1 = 0$ получена путем линейной аппроксимации функции накладных расходов при $n = 1$ до значения $m = 0$.

Из семейства кривых (рис. 2) функции $\eta_n(M) = f(n)$ видно, что:

1) увеличение размера подсистемы ($n \to N$) сопровождается некоторым снижением коэффициента использования транспьютерных элементов: $\forall n > 1\ \eta_n(M) < \eta_{n-1}(M)$, причем $\eta_n(M) - \eta_{n+1}(M) < \eta_{n-1}(M) - \eta_n(M)$, или $\eta_n(M) < (\eta_{n-1}(M) + \eta_{n+1}(M))/2$, т. е. график указанной функции имеет вид вогнутой кривой. Как показано в разд. 2, это снижение обусловлено невозможностью полного распараллеливания накладных расходов при обработке;

Таблица 2

M	$n^k_{\max}$	$k_{\max}$	E_n	$n^E_{\max}$	$k^E_{\max}$	$E_{\max}$
36	14	3,546	0,299	5	2,75	0,504
64	22	5,883	0,524	8	4,545	0,861
100	29	8,355	0,802	11	6,47	1,27
128	46	11,815	1,012	16	8,987	1,683

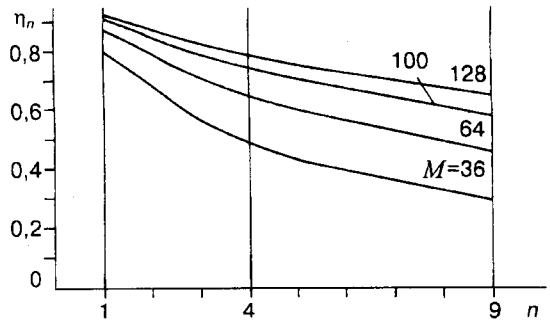

Рис. 2

2) снижение коэффициента использования транспьютерных элементов при увеличении размера подсистемы тем менее существенно, чем больше объем обрабатываемых данных, т. е.

$$\forall n > 1, M_1 < M_2 \; \eta_{n-1}(M_2) - \eta_n(M_2) < \eta_{n-1}(M_1) - \eta_n(M_1);$$

3) коэффициент использования транспьютеров возрастает с увеличением объема обрабатываемых программой данных: $\forall n \geq 1, M_1 < M_2 \; \eta_n(M_1) < \eta_n(M_2)$, что обусловлено снижением относительной доли накладных расходов в общем объеме обработки, увеличившемся за счет объема обрабатываемых данных.

Из структуры временных затрат на реализацию параллельного алгоритма совершенно очевидно, что приведенные выше утверждения справедливы не только для рассматриваемой, но и для любой параллельной программной системы.

Из графиков функций $k_n(M) = f(n)$ и $E_n(M) = f(n)$ видно (рис. 3), что возможности параллельной реализации исследуемой программы возрастают с увеличением массива обрабатываемой информации:

1) значение эффективности распараллеливания $E_n(M)$ возрастает до некоторого значения $n^E_{\max}(M)$; последующее увеличение числа транспьютеров в подсистеме до $n^k_{\max}(M)$ сопровождается несущественным ростом коэффициента ускорения и снижением эффективности параллельной обработки $n^E_{\max}(M) < n^k_{\max}(M)$;

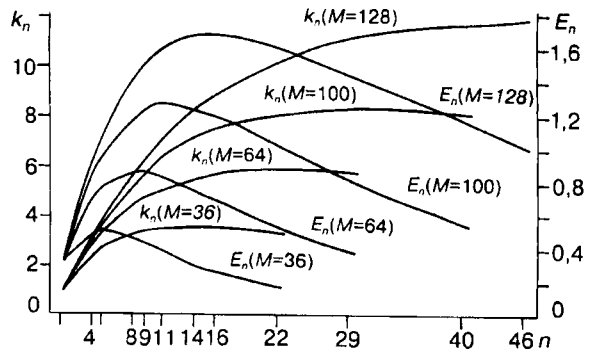

Рис. 3

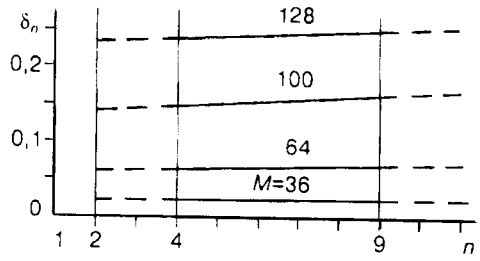

Рис. 4

2) показатели качества распараллеливания обработки (коэффициент ускорения и эффективность) возрастают с увеличением размеров перемножаемых матриц: при $M_1 < M_2$ $k_n(M_1) < k_n(M_2)$ и $E_n(M_1) < E_n(M_2)$;

3) границы эффективного наращивания подсистемы при увеличении размеров перемножаемых матриц отодвигаются в сторону возрастания n, т. е. $n^E_{max}(M_1) < n^E_{max}(M_2)$, $n^k_{max}(M_1) < n^k_{max}(M_2)$.

Оценивая семейство $\delta_n = f(n)$ (рис. 4) для различных значений M, можно убедиться в том, что их характер приближенно может быть описан уравнением прямой $\delta_n = (n-1)x + my$, $n \geq 2$. Здесь $x = \Delta$ — среднее время установления соединения между модулями — определяется быстродействием используемого протокола и реализующего его сетевого драйвера, алгоритмом маршрутизации и физической структурой ВС. Величина my — суммарное среднее время обмена (в данном случае среднее время передачи информационного массива). Вследствие недостатка информации об алгоритме обработки невозможно сделать более точные предположения, тем более учесть дублирование данных и другие факторы. Однако и используемые предположения достаточно близко отражают характер изменения δ_n. Полученная при этом величина $y = 0{,}0189$ с не противоречит значению быстродействия транспьютерных линков.

Исходя из вышеизложенного, можно сделать вывод о том, что быстродействие линков в значительной степени определяет показатели эффективности распараллеливания исследуемой программы. Проиллюстрируем это утверждение гипотетическим увеличением быстродействия линков в 10 раз. Полученные при этом результаты приведены в табл. 3.

Сравнение полученных величин с приведенными в табл. 2 показывает, что увеличение быстродействия линков (или уменьшение объемов обмениваемой информации) позволило бы приблизить значение n^E_{max} к величине n^k_{max} и добиться значительного увеличения абсолютных значений k_n и E_n, k_{max} и E_{max}. При этом возможности максимального наращивания числа транспьютеров в ВС остаются прежними, т. е. $n^k_{max} = \text{const}$.

Итак, анализ результатов хронометража параллельной программы перемножения матриц на транспьютерной ВС показал, что:

1) программная реализация алгоритма перемножения довольно эффективна, коэффициент использования транспьютеров достаточно высок;

Таблица 3

M	n^k_{max}	k_{max}	E_{kmax}	n^E_{max}	k^E_{max}	E_{max}
36	14	6,558	1,024	8	5,721	1,364
64	22	10,218	1,582	12	8,72	2,112
100	29	13,621	2,133	16	11,643	2,824
128	46	21,158	3,224	25	17,975	4,308

2) использование операционных средств в программном комплексе или отсутствует, или настолько незначительно, что практически не сказывается на эффективности параллельной реализации, $\alpha = 0$;

использования обрабатывающих модулей. Использование показателя позволило определить соотношение между коэффициентом ускорения и числом элементарных машин в подсистеме, ограничивающее снизу область эффективного распараллеливания алгоритмов.

Определена структура временных затрат на реализацию параллельного алгоритма, показана ее зависимость от числа ЭМ в подсистеме и от объема обрабатываемых данных. Выделены факторы, ограничивающие возможности эффективного распараллеливания, показаны пути оптимизации параллельных программных систем.

Проведен анализ параллельной реализации на транспьютерной системе программы перемножения матриц, демонстрирующий возможности изложенной методики.

Предлагаемые критерии эффективности, структура и соотношения временных затрат, методики анализа и оптимизации могут быть использованы как в их прямом назначении, сформулированном в названии статьи, так и на этапе проектирования при определении параметров системы (подсистемы), ориентированной на решение вполне определенных задач или их набора.

СПИСОК ЛИТЕРАТУРЫ

1. **Делвис Л. М., Браун Н. Г.** Численные библиотеки для транспьютерных сетей // Транспьютеры. Архитектура и программное обеспечение: Пер. с англ. /Под ред. Г. Харпа. М.: Радио и связь, 1993.

Поступила в редакцию 1 июля 1997 г.