Л4. Анализ и оценка времени выполнения параллельных алгоритмов

1. Гергель В. П. Теория и практика параллельных вычислений. – М.: Интернет-Университет Информационных Технологий; БИНОМ. Лаборатория Базовых Знаний, 2007. – 427 с.
2. С.Немнюгин, О.Стесик. Параллельное программирование для многопроцессорных вычислительных систем. – СПб.: БХВ-Петербург, 2002.
3. В.В.Воеводин, Вл.В.Воеводин. Параллельные вычисления СПб.: БХВ-Петербург, 2002.
4. Bertsekas D.P., Tsitsiklis J.N. Parallel and Distribution Computation. Numerical Methods. – Prentice Hall, Englewood Cliffs, New Jersey, 1989

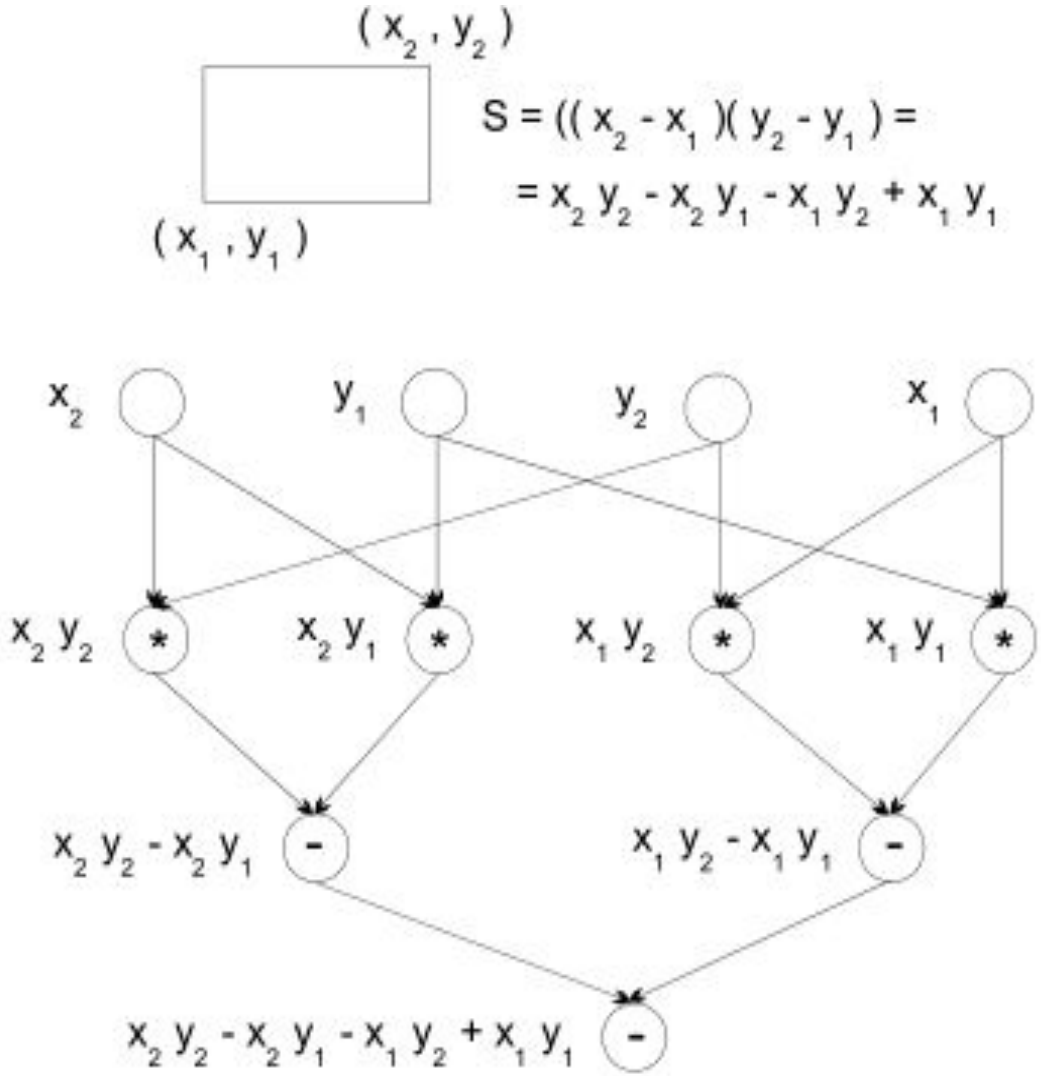
1. Анализ параллельных вычислений

1. При разработке параллельных алгоритмов решения сложных научно-технических задач принципиальным моментом является анализ эффективности использования параллелизма, состоящий обычно в оценке получаемого ускорения процесса вычислений (сокращения времени решения задачи).
2. Формирование подобных оценок ускорения может осуществляться применительно к выбранному вычислительному алгоритму (оценка эффективности распараллеливания конкретного алгоритма).
3. Другой важный подход может состоять в построении оценок максимально возможного ускорения процесса решения задачи конкретного типа (оценка эффективности параллельного способа решения задачи).

2. Модель в виде графа "операции-операнды"

1. Для уменьшения сложности излагаемого материала при построении модели предполагается, что время выполнения любых вычислительных операций является одинаковым и равняется 1 (в тех или иных единицах измерения).
2. Кроме того, принимается, что передача данных между вычислительными устройствами выполняется мгновенно без каких-либо затрат времени (что может быть справедливо, например, при наличии общей разделяемой памяти в параллельной вычислительной системе).
3. Множество операций, выполняемых в исследуемом алгоритме решения вычислительной задачи, и существующие между операциями информационные зависимости в виде ациклического ориентированного графа $G=(V, R)$, где $V=\{1, 2, \ldots |V|\}$ есть множество вершин графа, представляющих выполняемые операции алгоритма, а R есть множество дуг графа. При этом дуга $r =(i, j)$ принадлежит графу тогда и только тогда, когда операция j использует результат выполнения операции i. Вершины без входных дуг моделируют операции ввода, а вершины без выходных дуг – операции вывода.
5. Обозначим через $d(G)$ диаметр (длину максимального пути) графа.
6. Операции алгоритма, между которыми нет пути в $G=(V, R)$, могут быть выполнены параллельно.

3. Пример модели алгоритма в виде графа "операции-операнды"



4. Выбор вычислительной схемы

1. Для примера на слайде 3 показан граф алгоритма вычисления площади прямоугольника, заданного координатами двух противолежащих углов. Согласно этой модели для выполнения вычислений потребуется 4 процессора, при этом 4 операции умножения и 2 операции сложения будут выполняться парал-лельно.
2. Временная сложность вычислительной схемы в условных единицах равна длине максимального пути, т. е. равна 2.
3. Как можно заметить по приведенному примеру, для решения
задачи могут быть использованы разные схемы вычислений и построены соответственно разные вычислительные модели.
4. Как будет показано далее, разные схемы вычислений обладают различными возможностями для распараллеливания и,
тем самым, при построении модели вычислений может быть поставлена задача выбора наиболее подходящей для парал-лельного исполнения вычислительной схемы алгоритма.
5. Возможный способ описания параллельного

5. Расписание

Пусть p есть количество процессоров, используемых для выполнения алгоритма. Тогда для параллельного выполнения вычислений необходимо задать расписание в виде множества: $H_p = \{(i, P_i, t_i): i \in V\}$, в котором для каждой операции $i \in V$ указывается номер используемого для выполнения операции процессора P_i и время начала выполнения операции t_i. Для того, чтобы расписание было реализуемым, необходимо выполнение следующих требований при задании множества H_p:

1) $\forall\ i, j \in V: t_i = t_j \rightarrow P_i \neq P_j$ т.е. один и тот же процессор не должен назначаться разным операциям в один и тот же момент времени,

2) $\forall (i, j) \in R \rightarrow t_j \geq t_i + 1$, т.е. к назначаемому моменту выполнения операции все необходимые данные уже должны быть вычислены.

6. Время выполнения параллельного алгоритма

1. Вычислительная схема алгоритма G совместно с расписанием H_p может рассматриваться как модель параллельного алгоритма $Ap(G, H_p)$, исполняемого с использованием p процессоров. Время выполнения параллельного алгоритма определяется максимальным значением времени, используемым в расписании.

$$T_p(G, H_p) = \max_{i \in V}(t_i + 1).$$

2. Для выбранной схемы вычислений желательно использование расписания, обеспечивающего минимальное время исполнения алгоритма

$$T_p(G) = \min_{H_p} T_p(G, H_p).$$

3. Уменьшение времени выполнения может быть обеспечено и путем подбора наилучшей вычислительной схемы

$$T_p = \min_{G} T_p(G).$$

7. Минимально возможное время выполнения

Оценки $T_p(G,H_p)$, $T_p(G)$ и T_p могут быть использованы в качестве показателей времени выполнения параллельного алгоритма. Кроме того, для анализа максимально возможного параллелизма можно определить оценку наиболее быстрого исполнения алгоритма

$$T_\infty = \min_{p\geq 1} T_p .$$

Оценку T_∞ можно рассматривать как минимально возможное время выполнения параллельного алгоритма при использовании неограниченного количества процессоров (концепция вычислительной системы с бесконечным количеством процессоров, обычно называемой *паракомпьютером*, широко используется при теоретическом анализе параллельных вычислений).

8. Оценка времени выполнения при p=1

Оценка T_1 определяет время выполнения алгоритма при использовании одного процессора. Очевидно, что

$T_1(G) = |\overline{V}|$, т. е. есть количество вершин вычислительной схемы **G** без вершин ввода. Важно отметить, что если при определении оценки T_1 рассмотреть только один алгоритм решения задачи и использовать величину

$T_1 = \min_G T_1(G)$, то оценки ускорения будут характеризовать эффективность распараллеливания выбранного алгоритма. Для оценки эффективности параллельного решения исследуемой задачи время T_1 следует определять с учетом различных последовательных алгоритмов, т.е. использовать величину

$T_1^* = \min T_1$, где операция минимума берется по множеству всех возможных последовательных алгоритмов решения данной задачи.

9. Теоретические оценки времени выполнения параллельного алгоритма [4]

Теорема 1. Минимально возможное время выполнения параллельного алгоритма определяется длиной максимального пути ***G***, т.е.

$$T_\infty(G) = d(G)\,.$$

Теорема 2. Пусть для некоторой вершины вывода ***G*** существует путь из каждой вершины ввода, и входная степень вершин схемы не превышает 2. Тогда минимально время выполнения параллельного алгоритма ограничено снизу :

$T_\infty(G) = \log_2 n$, где ***n*** количество вершин ввода в ***G***.

Теорема 3. При уменьшении числа ***p*** время выполнения алгоритма увеличивается пропорционально величине уменьшения ***p***:

$$\forall q = cp, \quad 0 < c < 1 \Rightarrow T_p \le cT_q\,.$$

10. Теоретические оценки времени выполнения параллельного алгоритма (продолжение)

Теорема 4. Для любого количества используемых процессоров справедлива верхняя оценка для времени выполнения параллельного алгоритма:

$$\forall p \Rightarrow T_p < T_\infty + T_1 / p .$$

Теорема 5. Времени выполнения алгоритма, которое сопоставимо с минимально возможным временем T_∞, можно достичь при p порядка T_1 / T_∞:

$$p \geq T_1 / T_\infty \Rightarrow T_p \leq 2T_\infty .$$

При меньшем p время выполнения алгоритма не может превышать более чем в 2 раза, наилучшее время вычислений при имеющемся числе процессоров, т.е.

$$p < T_1 / T_\infty \Rightarrow \frac{T_1}{p} \leq T_p \leq 2\frac{T_1}{p} .$$

11. Рекомендации по правилам разработке параллельных алгоритмов

1. При выборе вычислительной схемы алгоритма должен использоваться граф с минимально возможным диаметром (см. теорему 1).
2. Для параллельного выполнения целесообразное количество процессоров определяется величиной p порядка $T1/T\infty$ (см. теорему 5);
3. Время выполнения параллельного алгоритма ограничивается сверху величинами, приведенными в теоремах 4 и 5.

12. Доказательство теоремы 4 (составление расписания).

Пусть H_∞ - расписание для достижения времени выполнения T_∞. Для каждой итерации τ, $0 \le \tau \le T_\infty$, расписания H_∞ выполняется n_τ операций. Расписание выполнения алгоритма с использованием p процессоров может быть построено следующим образом. Выполнение алгоритма разделим на T_∞ шагов; на каждом шаге τ следует выполнить все n_τ операций, которые выполнялись согласно H_∞. Выполнение этих операций может быть выполнено не более чем за $(n_\tau / p + 1)$ итераций при использовании p процессоров. Как результат, время выполнения алгоритма T_p может быть оценено следующим образом:

$$T_p = \sum_{\tau=1}^{T_\infty} \left\lceil \frac{n_\tau}{p} \right\rceil < \sum_{\tau=1}^{T_\infty} \left(\frac{n_\tau}{p} + 1 \right) = \frac{T_1}{p} + T_\infty .$$

13. Показатели эффективности параллельного алгоритма

1. Ускорение (speedup), получаемое при использовании параллельного алгоритма для p процессоров, по сравнению с последовательным вариантом выполнения вычислений определяется величиной $S_p(n)=T1(n)/T_p(n)$.

2. Эффективность (efficiency) использования параллельным алгоритмом процессоров при решении задачи определяется соотношением $E_p(n)=T1(n)/(pT_p(n)) = S_p(n)/p$.

Из приведенных соотношений видно, что в лучшем случае $S_p(n)= p$ и $E_p(n)=1$.

14. Практическое применение оценок эффективности параллельных вычислений

1. При определенных обстоятельствах ускорение может оказаться больше числа используемых процессоров $S_p(n) > p$ в этом случае место сверхлинейное (superlinear) ускорение.
2. Попытки повышения качества параллельных вычислений по одному из показателей (ускорению или эффективности) может привести к ухудшению ситуации по другому показателю, разработка методов параллельных вычислений предполагает выбор компромиссного варианта с учетом желаемых показателей ускорения и эффективности.
3. При выборе надлежащего параллельного способа решения задачи может оказаться полезной оценка стоимости (*cost*) вычислений $C_p = p T_p$.
4. Стоимостно-оптимальным (*cost-optimal*) параллельным алгоритмом является алгоритм, стоимость которого оказывается пропорциональной времени выполнения наилучшего последовательного алгоритма.