



# АНАЛИЗ СЛОЖНОСТИ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ

Сложность – это сумма простых трудностей.

*Г. Александров*

# Содержание

2

- Модель вычислений "операции-операнды"
- Схема параллельного выполнения алгоритма
- Оценки времени выполнения параллельного алгоритма

# Модель "операции-операнды"

3

- Допущения
  - ▣ время выполнения любых вычислительных операций является одинаковым и равно 1
  - ▣ передача данных между вычислительными устройствами выполняется мгновенно.
- Алгоритм представляется в виде ациклического ориентированного графа  $G(V,R)$ 
  - ▣ Вершины  $V=\{1,\dots,|V|\}$  – операции алгоритма.
  - ▣ Дуги  $R=\{r(i,j)\}$  – информационные связи между операциями (операция  $j$  использует результат операции  $i$ ).
  - ▣ Вершины без входных дуг – операции ввода, вершины без выходных дуг – операции вывода.
  - ▣  $\bar{V}$  – множество вершин графа без вершин ввода.
  - ▣  $d(G)$  – диаметр графа, длина максимального пути.

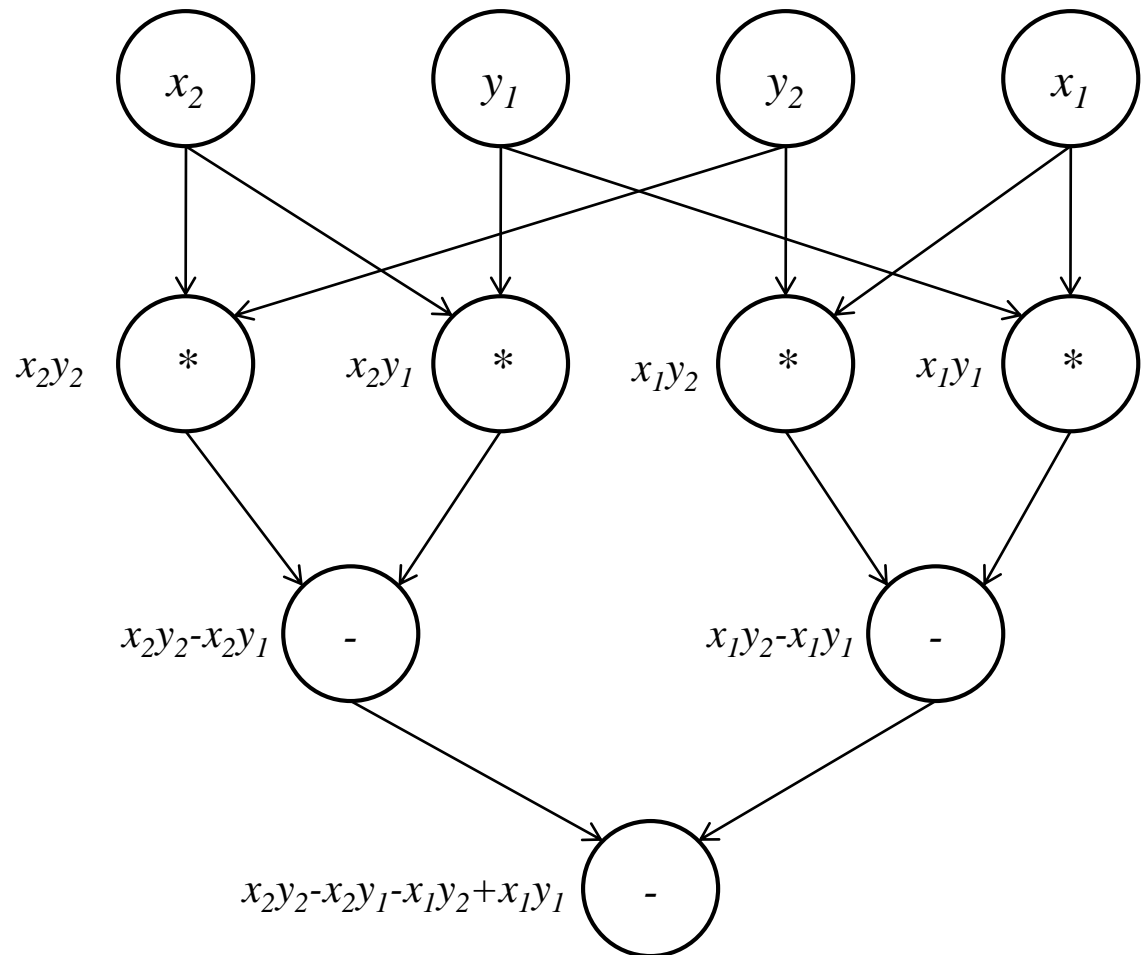
# Граф "операции-операнды"

4



$$S = (x_2 - x_1)(y_2 - y_1) = x_2y_2 - x_2y_1 - x_1y_2 + x_1y_1$$

Если между некоторыми вершинами графа не существует соединяющий их путь, то соответствующие операции могут быть выполнены параллельно.



# Схема параллельного выполнения

5

- Расписание параллельного выполнения алгоритма – множество  $H_p = \{ (i, P_i, t_i), \text{ где } i \in V \}$ 
  - ▣  $p$  – количество процессоров для выполнения алгоритма
  - ▣  $i$  – номер операции
  - ▣  $P_i$  – номер процессора
  - ▣  $t_i$  – время начала  $i$ -й операции
- Необходимые условия
  - ▣ Один и тот же процессор не должен назначаться разным операциям в один и тот же момент времени  
 $\forall i, j \in V : t_i = t_j \Rightarrow P_i \neq P_j$
  - ▣ к назначаемому моменту выполнения операции все необходимые ей данные должны быть вычислены  
 $\forall (i, j) \in R \Rightarrow t_j \geq t_i + 1$

# Время выполнения последовательного алгоритма

6

- Время выполнения последовательного алгоритма для заданной вычислительной схемы

$$T_1(G) = |\bar{V}|$$

- Время выполнения последовательного алгоритма

$$T_1 = \min_G T_1(G)$$

- Время последовательного решения задачи

$$T_1^* = \min T_1$$

# Время выполнения параллельного алгоритма

7

- Модель параллельного алгоритма

$$A_p(G, H_p)$$

- Время выполнения параллельного алгоритма с заданным расписанием

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

- Время выполнения параллельного алгоритма с оптимальным расписанием

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

# Время выполнения параллельного алгоритма

8

- Минимально возможное время решения задачи при заданном количестве процессоров (определение *наилучшей вычислительной схемы*)

$$T_p = \min_G T_p(G)$$

- Оценка наиболее быстрого исполнения алгоритма (при использовании *паракомпьютера* – гипотетической системы с неограниченным числом процессоров)

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



# Оценки времени выполнения параллельного алгоритма

9

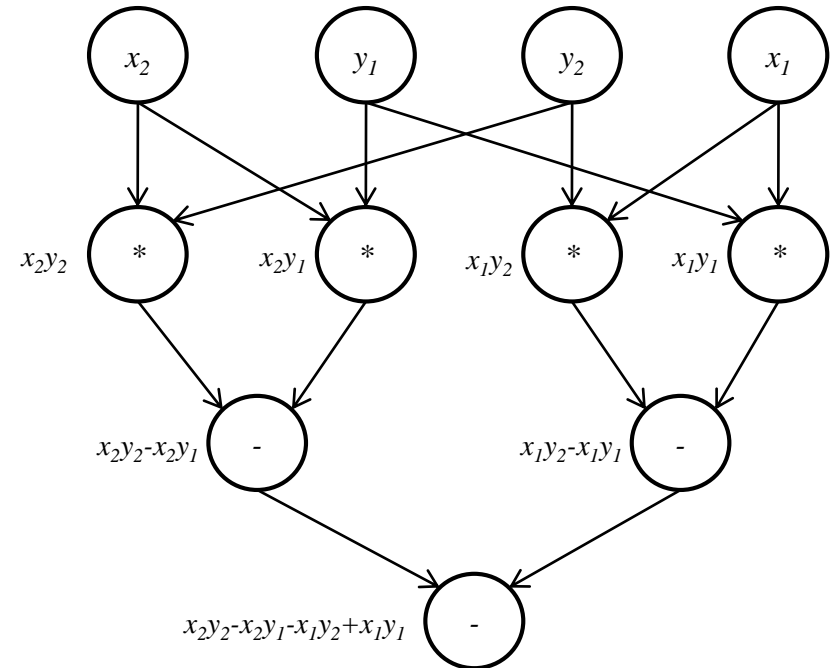
- Каково минимальное время выполнения?
- Каково максимальное время выполнения?
- Как изменяется время выполнения при уменьшении количества процессоров?
- Сколько нужно процессоров, чтобы достичь минимального времени?

# Теорема 1

10

- Минимально возможное время выполнения параллельного алгоритма определяется длиной максимального пути вычислительной схемы алгоритма:

$$T_{\infty}(G) = d(G)$$



# Теорема 2

11

- Пусть
  - для некоторой вершины вывода в вычислительной схеме алгоритма существует путь из каждой вершины ввода
  - входная степень вершин схемы (количество входящих дуг) не превышает 2.
- Тогда минимально возможное время выполнения параллельного алгоритма ограничено снизу значением  $\log_2 n$ , где  $n$  – количество вершин ввода в схеме алгоритма.

$$T_{\infty}(G) = \log_2 n$$

# Теорема 3

12

- При уменьшении количества используемых процессоров время выполнения алгоритма увеличивается пропорционально величине уменьшения количества процессоров

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

# Теорема 4

13

- Для любого количества используемых процессоров справедлива следующая верхняя оценка для времени выполнения параллельного алгоритма

$$\forall p \quad T_p < T_\infty + \frac{T_1}{p}$$

# Теорема 5

14

- Время выполнения алгоритма, сопоставимое с минимально возможным временем  $T_\infty$ , достигается при количестве процессоров порядка  $\frac{T_1}{T_\infty}$ , а именно:

$$p \geq \frac{T_1}{T_\infty} \Rightarrow T_p \leq 2T_\infty$$

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

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

# Выбор параллельного алгоритма

15

- Граф вычислительной схемы алгоритма должен иметь минимально возможный диаметр (Теорема 1).
- Целесообразное количество процессоров для параллельного выполнения определяется величиной  $\frac{T_1}{T_\infty}$  (Теорема 5).
- Время выполнения параллельного алгоритма ограничивается сверху величинами  $T_\infty + \frac{T_1}{p}$  (теорема 4) и  $2\frac{T_1}{p}$  (Теорема 5).

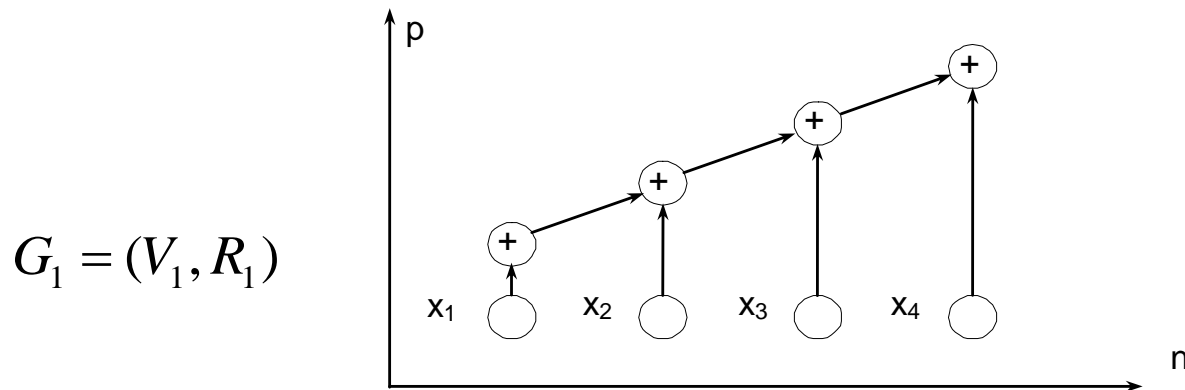
# Пример: вычисление суммы

16

- Пусть требуется вычислить сумму набора значений

$$S = \sum_{i=1}^n x_i$$

- Последовательный алгоритм



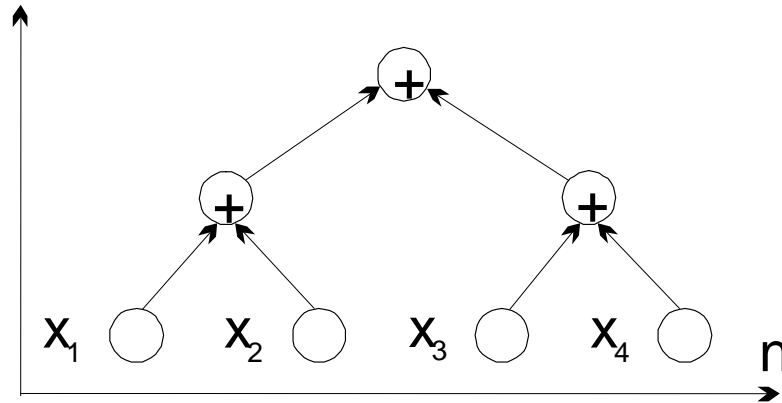
допускает только строго последовательное исполнение и не может быть распараллелен.



# Каскадная схема суммирования

17

$$G_2 = (V_2, R_2)$$



- Количество итераций  
 $k = \log_2 n$
- Общее количество операций суммирования
  - ▣ Последовательное исполнение  
 $K_{\text{послед}} = n/2 + n/4 + \dots + 1 = n - 1$
  - ▣ Параллельное исполнение  
 $K_{\text{пар}} = \log_2 n$

# Оценка эффективности каскадной схемы

18

- Ускорение

$$S_p = T_1 / T_p = (n-1) / \log_2 n$$

- Эффективность

$$E_p = T_1 / pT_p = (n-1) / (p \log_2 n) = (n-1) / ((n/2) \log_2 n)$$

- Необходимое количество процессоров

$$p = n/2$$

- Наблюдения

- Время параллельного выполнения каскадной схемы совпадает с оценкой для паракомпьютера (Теорема 2).

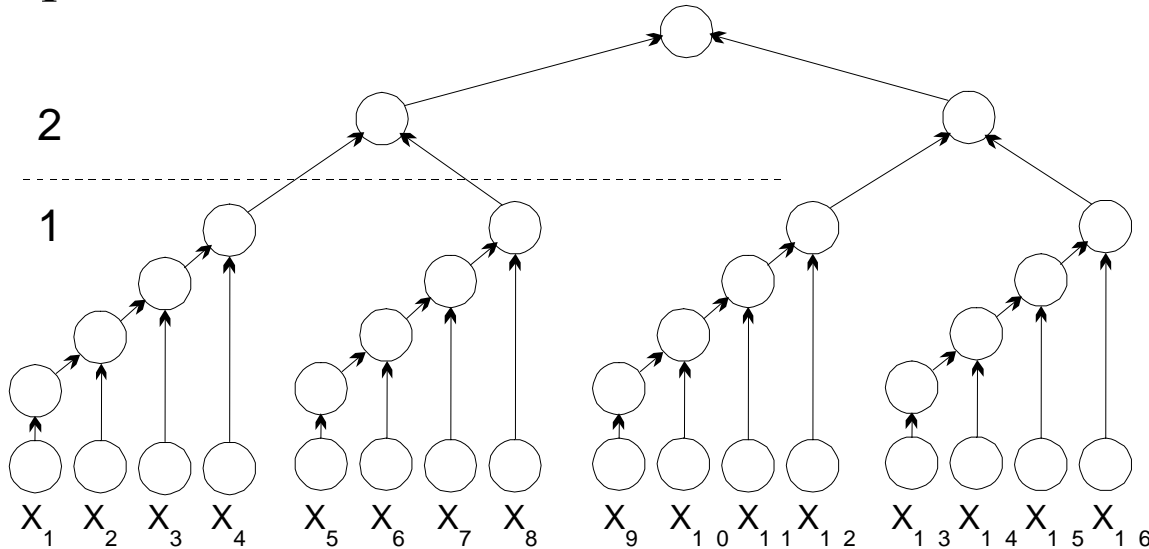
- Эффективность использования процессоров уменьшается при увеличении количества суммируемых значений:

$$\lim_{n \rightarrow \infty} E_p = 0$$

# Улучшение каскадной схемы

19

- Все суммируемые значения подразделяются на  $n/\log_2 n$  групп, в каждой из которых содержится  $\log_2 n$  элементов; для каждой группы вычисляется сумма значений при помощи последовательного алгоритма суммирования;
- На втором этапе для полученных  $n/\log_2 n$  сумм отдельных групп применяется обычная каскадная схема



# Оценка эффективности улучшенной каскадной схемы

20

- Для выполнения первого этапа требуется  $\log_2 n$  выполнений параллельных операций при использовании  $p = n / \log_2 n$  процессоров.
- Для выполнения второго этапа необходимо  $\log_2(n / \log_2 n) \leq \log_2 n$  параллельных операций для  $p = (n / \log_2 n) / 2$  процессоров.
- Время выполнения параллельного алгоритма составляет  $T_p = 2 \log_2 n$  для  $p = (n / \log_2 n)$  процессоров.

# Оценка эффективности улучшенной каскадной схемы

21

## □ Ускорение

$$S_p = T_1 / T_p = (n-1) / 2 \log_2 n$$

## □ Эффективность

$$E_p = T_1 / p T_p = (n-1) / (2(n / \log_2 n) \log_2 n) = (n-1) / 2n$$

## □ Наблюдения

- По сравнению с обычной каскадной схемой ускорение уменьшилось в 2 раза.
- Для эффективности нового метода суммирования можно получить асимптотически ненулевую оценку снизу

$$E_p = (n-1) / 2n \geq 0,25 \qquad \lim_{n \rightarrow \infty} E_p = 0,5$$

- Каскадный алгоритм является стоимостно-оптимальным (стоимость вычислений пропорциональна времени выполнения последовательного алгоритма)

$$C_p = p T_p = (n / \log_2 n) (2 \log_2 n) = 2n$$

# Оценка эффективности улучшенной каскадной схемы

22

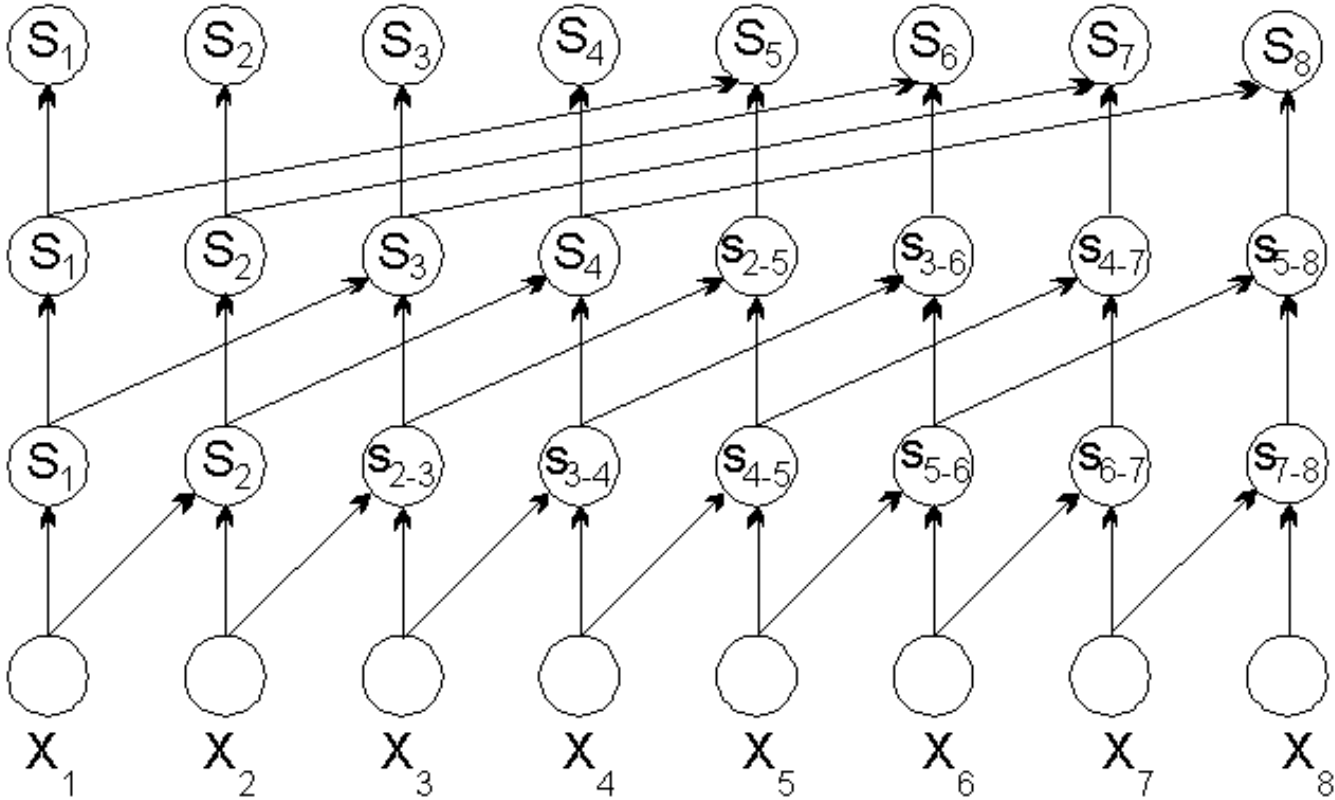
- Вычисление суммы на скалярном компьютере может быть получено при помощи обычного последовательного алгоритма суммирования при том же количестве операций:  $T_1 = n$ .
- При параллельном исполнении применение каскадной схемы в явном виде не приводит к желаемым результатам.
- Достижение эффективного распараллеливания требует привлечения новых подходов (возможно, не имеющих аналогов в последовательном программировании).

# Эффективный параллельный алгоритм

23

- Алгоритм, обеспечивающий получение результатов за  $\log_2 n$  параллельных операций:
  - Перед началом вычислений создается копия  $S$  вектора суммируемых значений ( $S=x$ ),
  - Далее на каждой итерации суммирования  $i$  ( $1 \leq i \leq \log_2 n$ ) формируется вспомогательный вектор  $Q$  путем сдвига вправо вектора  $S$  на  $2^{i-1}$  позиций. Освобождающиеся при сдвиге позиции слева устанавливаются в нулевые значения.
  - Итерация алгоритма завершается параллельной операцией суммирования векторов  $S$  и  $Q$ .

# Эффективный параллельный алгоритм





# Оценка эффективности

25

- Общее количество выполняемых скалярных операций

$$K_{\text{носл}} = n \log_2 n$$

- Необходимое количество процессоров (определяется количеством суммируемых значений)

$$p = n$$

- Ускорение

$$S_p = T_1 / T_p = n / \log_2 n$$

- Эффективность

$$E_p = T_1 / pT_p = n / (p \log_2 n) = n / n \log_2 n = 1 / \log_2 n$$

# Заключение

26

- Модель вычислений "операции-операнды"
- Схема параллельного выполнения алгоритма
- Оценки времени выполнения параллельного алгоритма