

Оценка эффективности параллельных алгоритмов

Лекция 4.

Введение

- Принципиальный момент при разработке параллельных алгоритмов - **анализ эффективности использования параллелизма:**
 - Оценка максимально возможного ускорения процесса решения рассматриваемой задачи (анализ всех возможных способов выполнения вычислений)
 - Оценка эффективности распараллеливания конкретных выбранных методов выполнения вычислений

4.1 Показатели эффективности параллельного алгоритма

■ Ускорение (*speedup*)

получаемое при использовании параллельного алгоритма для p процессоров, по сравнению с последовательным вариантом выполнения вычислений, определяется величиной

$$S_p(n) = T_1(n) / T_p(n)$$

(величина n используется для параметризации вычислительной сложности решаемой задачи и может пониматься, например, как количество входных данных задачи)

4.1 Показатели эффективности параллельного алгоритма

■ Эффективность (*efficiency*)

использования параллельным алгоритмом процессоров при решении задачи определяется соотношением:

$$E_p(n) = T_1(n) / (pT_p(n)) = S_p(n) / p$$

Величина эффективности определяет среднюю долю времени выполнения параллельного алгоритма, в течение которого процессоры реально используются для решения задачи

4.1 Показатели эффективности параллельного алгоритма

■ Замечания

- Сверхлинейное (*superlinear*) ускорение $S_p(n) > p$ может иметь место в силу следующего ряда причин:
 - неравноправность выполнения последовательной и параллельной программ (например, недостаток ОП),
 - нелинейный характер зависимости сложности решения задачи от объема обрабатываемых данных,
 - различие вычислительных схем последовательного и параллельного методов.
- Показатели качества параллельных вычислений являются противоречивыми: попытки повышения качества параллельных вычислений по одному из показателей может привести к ухудшению ситуации по другому показателю.

4.1 Показатели эффективности параллельного алгоритма

- **Стоимость (*cost*)** вычислений

$$C_p = pT_p$$

- **Стоимостно-оптимальный (*cost-optimal*)** параллельный алгоритм - метод, стоимость которого является пропорциональной времени выполнения наилучшего последовательного алгоритма.

4.2 Оценка максимально достижимого параллелизма

- Оценка качества параллельных вычислений предполагает знание *наилучших (максимально достижимых)* значений показателей ускорения и эффективности
- Получение идеальных величин $S_p=p$ для ускорения и $E_p=1$ для эффективности может быть обеспечено не для всех вычислительно трудоемких задач

4.2 Оценка максимально достижимого параллелизма

■ Закон Амдаля (Amdahl)

Достижению максимального ускорения может препятствовать существование в выполняемых вычислениях последовательных расчетов, которые не могут быть распараллелены.

Пусть f есть доля последовательных вычислений в применяемом алгоритме обработки данных. Ускорение процесса вычислений при использовании p процессоров ограничивается величиной:

$$S_p \leq \frac{1}{f + (1 - f) / p} \leq S^* = \frac{1}{f}$$

4.2 Оценка максимально достижимого параллелизма

■ Закон Амдаля. Следствия

- Доля последовательных вычислений может быть существенно снижена при выборе более подходящих для распараллеливания методов
- Эффект Амдаля:

Для большого ряда задач доля последовательных вычислений $f=f(n)$ является убывающей функцией от n , и в этом случае ускорение для фиксированного числа процессоров может быть увеличено за счет увеличения вычислительной сложности решаемой задачи. В этом случае, ускорение $S_p = S_p(n)$ является возрастающей функцией от параметра n .

4.2 Оценка максимально достижимого параллелизма

■ Закон Густавсона – Барсиса

Оценим максимально достижимое ускорение исходя из имеющейся доли последовательных расчетов в выполняемых параллельных вычислениях:

$$g = \frac{\tau(n)}{\tau(n) + \pi(n) / p}$$

где $\tau(n)$ и $\pi(n)$ есть времена последовательной и параллельной частей

$$T_1 = \tau(n) + \pi(n), \quad T_p = \tau(n) + \pi(n) / p$$

С учетом введенной величины g можно получить

$$\tau(n) = g \cdot (\tau(n) + \pi(n) / p), \quad \pi(n) = (1 - g)p \cdot (\tau(n) + \pi(n) / p),$$

что позволяет построить оценку для ускорения

$$S_p = \frac{T_1}{T_p} = \frac{\tau(n) + \pi(n)}{\tau(n) + \pi(n) / p} = \frac{(\tau(n) + \pi(n) / p)(g + (1 - g)p)}{\tau(n) + \pi(n) / p}$$

4.2 Оценка максимально достижимого параллелизма

■ Закон Густавсона - Барсиса

Упрощение последней оценки для ускорения

$$S_p = g + (1 - g)p = p + (1 - p)g$$

Оценку ускорения, получаемую в соответствии с законом Густавсона-Барсиса, еще называют **ускорением масштабирования** (scaled speedup), поскольку данная характеристика может показать, насколько эффективно могут быть организованы параллельные вычисления при увеличении сложности решаемых задач

4.3 Анализ масштабируемости параллельных вычислений

Параллельный алгоритм называют **масштабируемым (scalable)**, если при росте числа процессоров он обеспечивает увеличение ускорения при сохранении постоянного уровня эффективности использования процессоров

4.3 Анализ масштабируемости параллельных вычислений

Накладные расходы (*total overhead*) появляются за счет необходимости организации взаимодействия процессоров, выполнения некоторых дополнительных действий, синхронизации параллельных вычислений и т.п.

$$T_0 = pT_p - T_1$$

Новые выражения для времени параллельного решения задачи и получаемого при этом ускорения:

$$T_p = \frac{T_1 + T_0}{p}, \quad S_p = \frac{T_1}{T_p} = \frac{pT_1}{T_1 + T_0}$$

Тогда эффективность использования процессоров можно выразить:

$$E_p = \frac{S_p}{p} = \frac{T_1}{T_1 + T_0} = \frac{1}{1 + T_0 / T_1}$$

4.3 Анализ масштабируемости параллельных вычислений

- Если сложность решаемой задачи является фиксированной ($T_1 = \text{const}$), то при росте числа процессоров эффективность, как правило, будет убывать за счет роста накладных расходов T_0
- При фиксации числа процессоров эффективность использования процессоров можно улучшить путем повышения сложности решаемой задачи T_1
- При увеличении числа процессоров в большинстве случаев можно обеспечить определенный уровень эффективности при помощи соответствующего повышения сложности решаемых задач

4.3 Анализ масштабируемости параллельных вычислений

- Пусть $E=const$ есть желаемый уровень эффективности выполняемых вычислений. Из выражения для эффективности можно получить

$$\frac{T_0}{T_1} = \frac{1-E}{E}, \text{ или } T_1 = KT_0, K = E/(1-E)$$

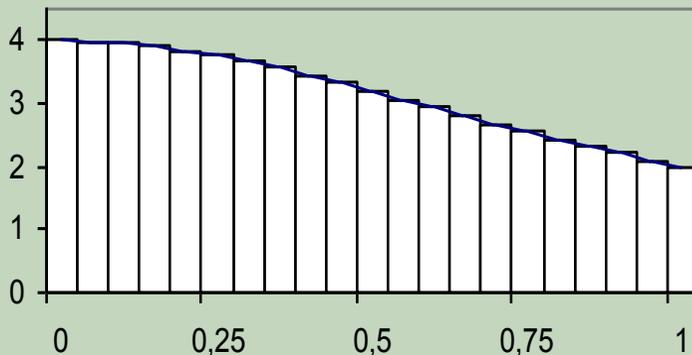
- Порождаемую последним соотношением зависимость $n=F(p)$ между сложностью решаемой задачи и числом процессоров обычно называют **функцией изоэффективности** (isoefficiency function).

4.4 Пример: *Вычисление числа π*

- ❑ Значение числа π может быть получено при помощи интеграла

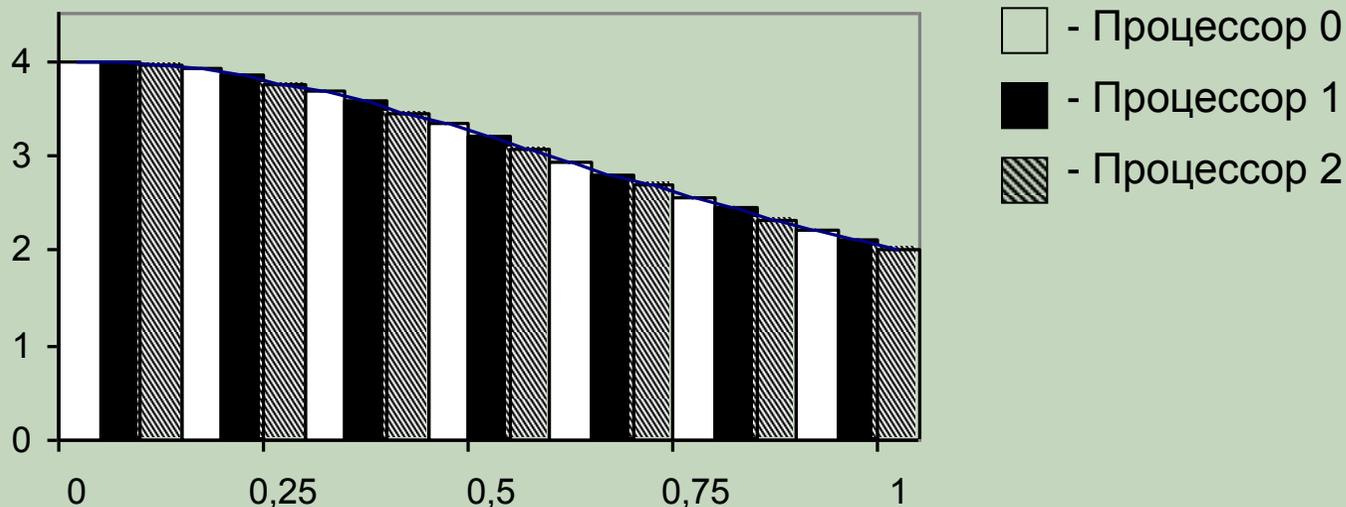
$$\pi = \int_0^1 \frac{4}{1+x^2} dx$$

- ❑ Для численного интегрирования применим метод прямоугольников



4.4 Пример: *Вычисление числа π*

- ❑ Распределим вычисления между p процессорами (циклическая схема)
- ❑ Получаемые на отдельных процессорах частные суммы должны быть просуммированы



4.4 Пример: *Вычисление числа π*

Анализ эффективности

□ n – количество разбиений отрезка $[0, 1]$

□ Вычислительная сложность задачи

$$W = T_1 = 6n$$

□ Количество узлов сетки на отдельном процессоре

$$m = \lceil n/p \rceil \leq n/p + 1$$

□ Объем вычислений на отдельном процессоре

$$W_p = 6m = 6n/p + 6.$$

4.4 Пример: *Вычисление числа π*

Анализ эффективности

- Время параллельного решения задачи

$$T_p = 6n/p + 6 + \log_2 p$$

- Ускорение $S_p = T_1 / T_p = 6n / (6n/p + 6 + \log_2 p)$

- Эффективность $E_p = 6n / (6n + 6p + p \log_2 p)$

- Функция изоэффективности

$$W = K(pT_p - W) = K(6p + p \log_2 p)$$

$$\Rightarrow n = [K(6p + p \log_2 p)]/6, \quad (K = E/(1-E))$$

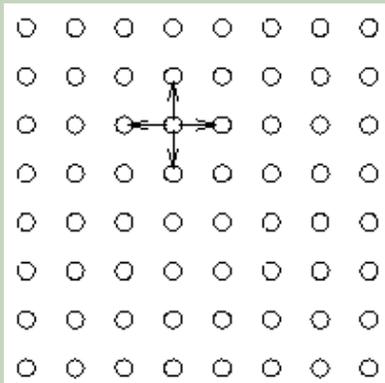
$$\text{Ex.: } E=0.5 \text{ при } p=8 \rightarrow n = 12$$

$$\text{при } p=64 \rightarrow n = 128$$

4.5 Пример: *Метод конечных разностей*

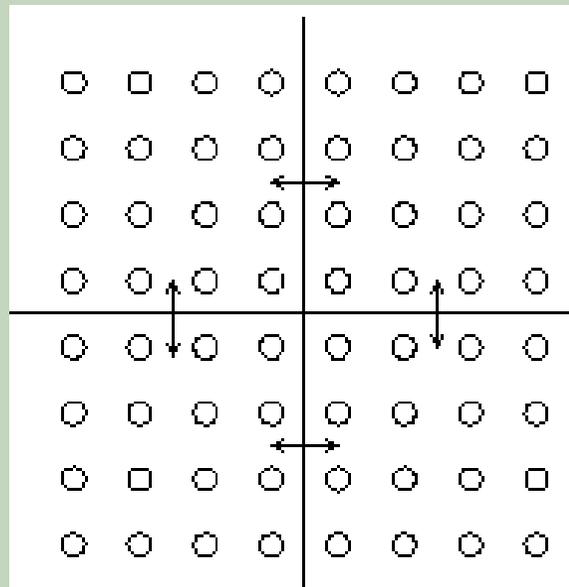
- Метод конечных разностей широко применяется для численного решения уравнений в частных производных
- Рассмотрим схему $N = 2$

$$X_{i,j}^{t+1} = w(X_{i,j-1}^t + X_{i,j+1}^t + X_{i-1,j}^t + X_{i+1,j}^t) + (1-w) X_{i,j}^t$$



4.5 Пример: *Метод конечных разностей*

- Каждый процессор проводит вычисления на прямоугольной подобласти с $(n/\sqrt{p}) * (n/\sqrt{p})$ точками
- После выполнения каждой итерации расчета необходима синхронизация расчета



4.5 Пример: *Метод конечных разностей*

Анализ эффективности

- $W = T_1 = 6n^2M$ (M – количество итераций)
- $T_p = 6n^2M/p + 4M$
- Ускорение $S_p = T_1 / T_p = 6n^2 / (6n^2/p + 4)$
- Функция изоэффективности
$$W = K(pT_p - W) = K(4p)$$
$$\Rightarrow n^2 = [K(4p)]/6, \quad (K=E/(1-E))$$

Метод конечных разностей является более масштабируемым, чем метод прямоугольников

4.6 Способы вычислений времени выполнения алгоритма

- T_1 - вычислительная сложность последовательного алгоритма
- $T_p = T_{\text{выч.}} + T_{\text{комм.}} + T_{\text{синхр.}}$

4.6 Способы вычислений времени выполнения алгоритма

- Для оценки **времени передачи сообщений** используется модель, предложенная Хокни (*Hockney*), в которой трудоемкость операции коммуникации между узлами вычислительной системы оценивается :

$$t_{n\partial}(n) = \alpha + n / \beta,$$

где α - латентность сети передачи данных

n - размер передаваемого сообщения в байтах

β - пропускная способность сети