Лекция №9

Оценка эффективности параллельных вычислений

Содержание

- Показатели эффективности параллельного алгоритма
  - Ускорение;
  - Эффективность;
  - Стоимость.
- Оценка максимально достижимого параллелизма
  - Закон Амдала;
  - Закон Густафсона;
- Анализ масштабируемости параллельного алгоритма.

Показатели эффективности

- *Ускорение* относительно последовательного выполнения вычислений;
- *Эффективность* использования процессоров;
- *Стоимость* вычислений.

Ускорение

- *Ускорение (speedup)*, получаемое при использовании параллельного алгоритма для p процессоров, по сравнению с последовательным вариантом выполнения вычислений:
  - $S_p(n) = \frac{T_1(n)}{T_p(n)}$
  - n – параметр вычислительной сложности решаемой задачи (например, количество входных данных задачи).

Абсолютное и относительное ускорение

- Величину ускорения называют *абсолютной*, если в качестве T_1 берется время выполнения наилучшего последовательного алгоритма.
- Величину ускорения называют *относительной*, если в качестве T_1 берется время выполнения параллельного алгоритма на одном процессоре.

Линейное и сверхлинейное ускорение

- *Линейное (linear)* или *идеальное (ideal)* ускорение имеет место при $S_p=p$.
- *Сверхлинейное (superlinear)* ускорение имеет место при $S_p>p$.
  - Неравноправность выполнения последовательной и параллельной программ (например, недостаток оперативной памяти).
  - Нелинейный характер зависимости сложности решения задачи от объема обрабатываемых данных.

- Различие вычислительных схем последовательного и параллельного методов.

Эффективность

- *Эффективность (efficiency)* – средняя доля времени выполнения параллельного алгоритма, в течение которого процессоры реально используются для решения задачи.

$$E_p(n) = \frac{T_1(n)}{p \cdot T_p(n)} = \frac{S_p(n)}{p}$$

Стоимость вычислений

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

$$C_p = p \cdot T_p$$

Можно ли достичь max параллелизма?

- Получение идеальных величин $S_p=p$ для ускорения и $E_p=1$ для эффективности может быть обеспечено не для всех вычислительно трудоемких задач.
- Достижению максимального ускорения может препятствовать существование в выполняемых вычислениях последовательных расчетов, которые не могут быть распараллелены.

Закон Амдала

- Задает связь между ожидаемым ускорением параллельных реализаций алгоритма и последовательным алгоритмом в предположении, что размер задачи остается постоянным.
- Пусть f – доля последовательных вычислений в алгоритме. Тогда

$$S_p = \frac{T_1}{T_p} = \frac{f + (1-f)}{f + \frac{1-f}{p}} = \frac{1}{f + \frac{1-f}{p}}, \quad \boxed{S_p = \frac{1}{f + \frac{1-f}{p}}}$$

$$\boxed{\lim_{p \to \infty} S_p = \frac{1}{f}}$$

Пример покраска забора (300 досок)

- *Подготовка* – 30 мин.
  НЕ распараллеливается
- *Покраска* (одной доски) – 1 мин.
  РАСПАРАЛЛЕЛИВАЕТСЯ
- *Уборка* – 30 мин.
  НЕ распараллеливается

Количество маляров	Время покраски
1	30 + 300/1 + 30 = 360
2	30 + 300/2 + 30 = 210
10	30 + 300/10 + 30 = 90
100	30 + 300/100 + 30 = 63
1000	30 + 300/1000 + 30 ≅ 60
1000000	30 + 300/1000000 + 30 ≅ 60

График ускорения

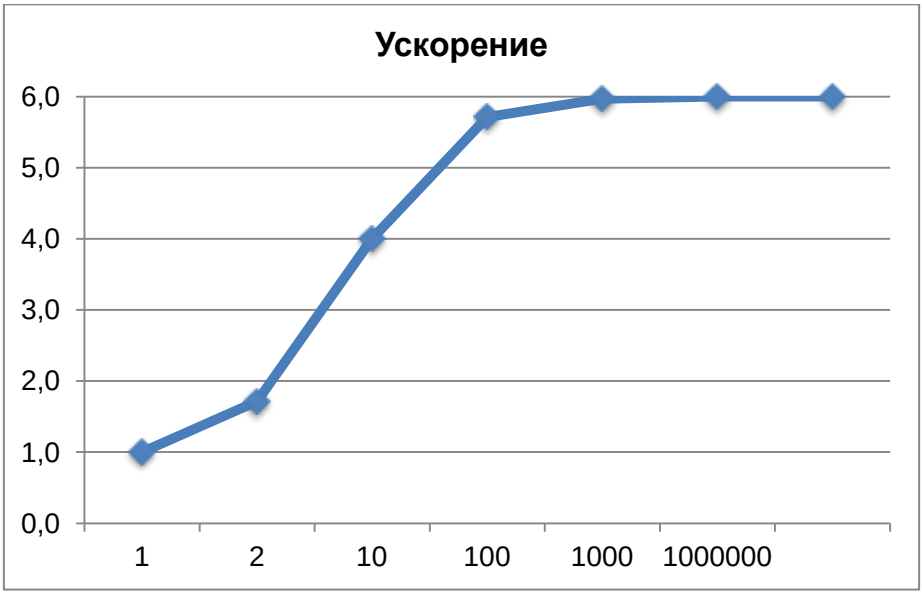
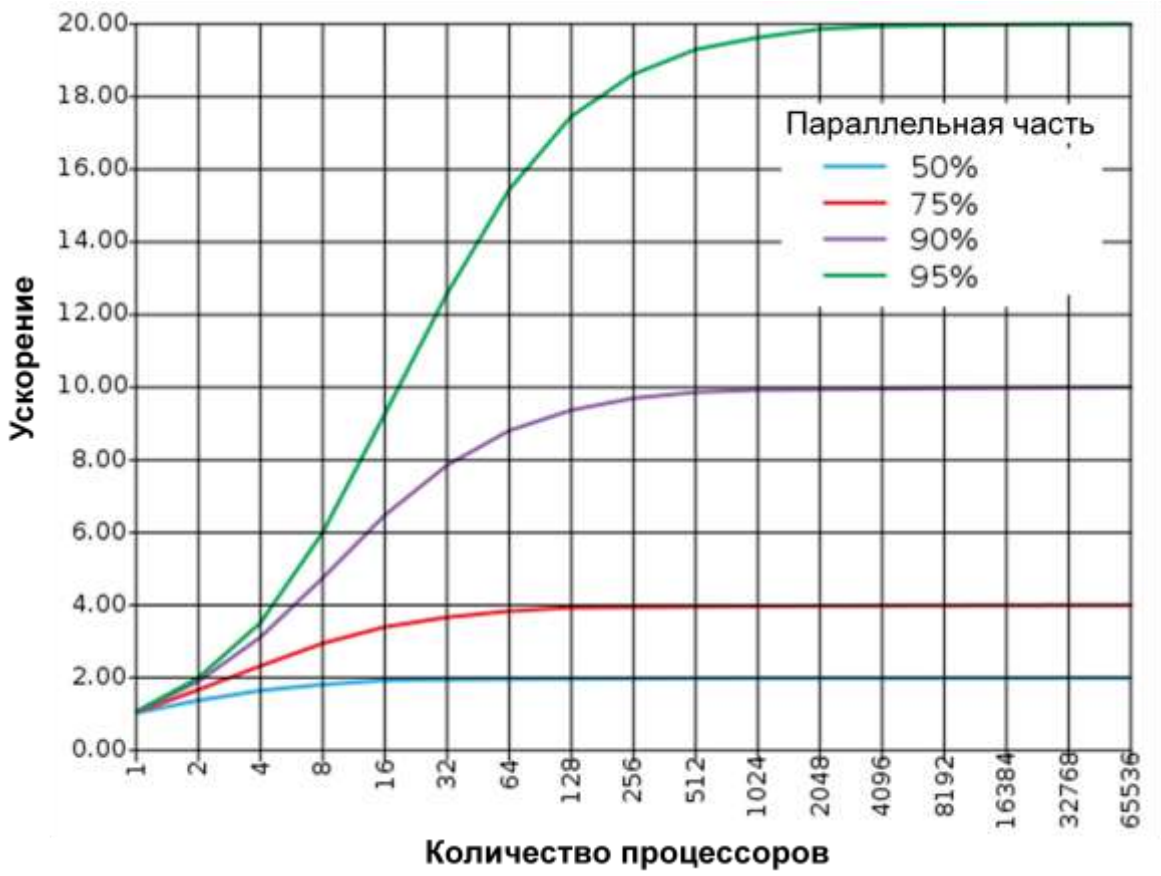
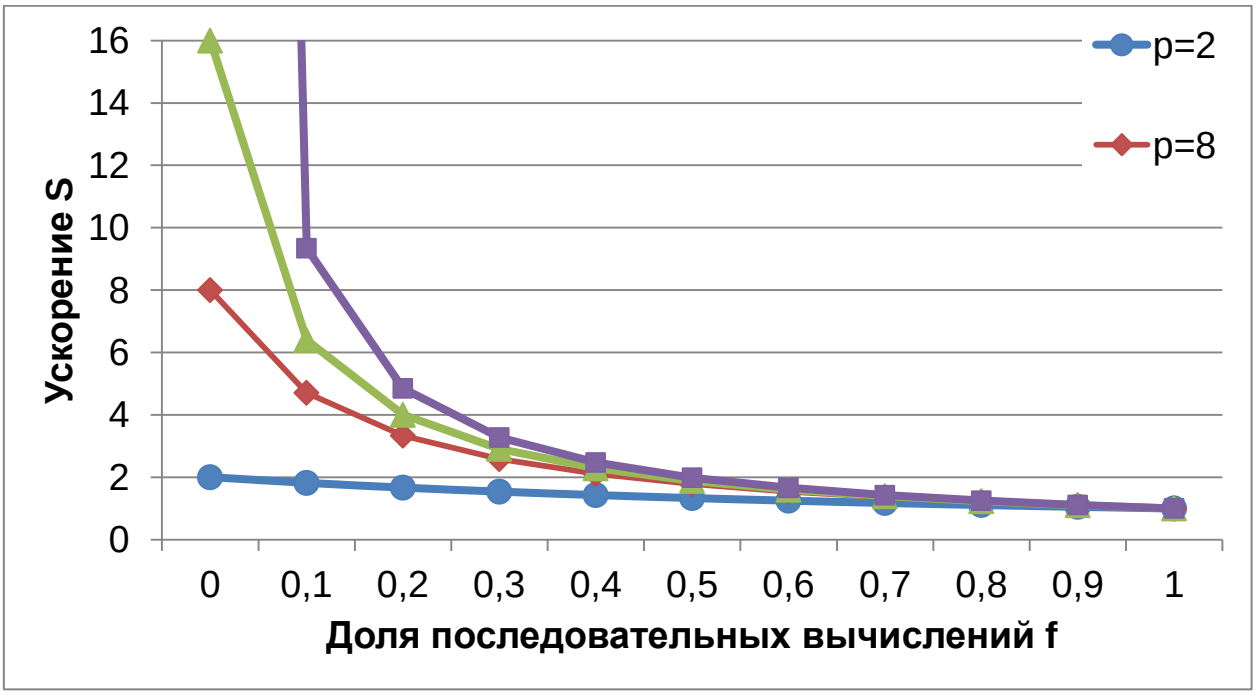


График эффективности

Ускорение параллельной программы зависит не от количества процессоров, а величины последовательной части программы.





Закон Густафсона

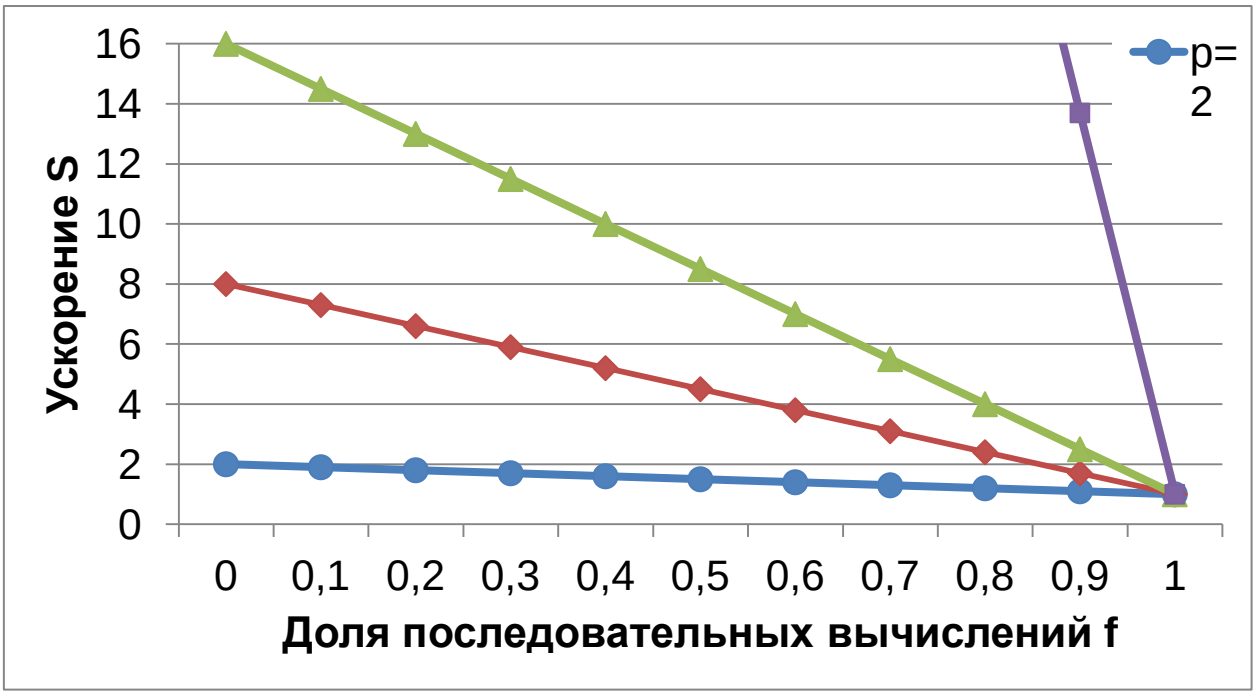
Закон Амдала предполагает, что количество процессоров и доля параллельной части программы независимы, что не совсем верно.

- Как правило, задача с фиксированным объемом данных не запускается на различном количестве процессоров (за исключением

академических исследований), а объем данных изменяется в соответствии с количеством процессоров.

- Вместо вопроса об ускорении на p процессорах рассмотрим вопрос о замедлении вычислений при переходе на один процессор.

$$S_p = \frac{f + (1-f)\cdot p}{f + (1-f)} = f + p - f \cdot p = p - f \cdot (p-1), \quad S_p = p - f \cdot (p-1)$$



Законы Амдала и Густафсона

- Уменьшение времени выполнения *vs* увеличение объема решаемой задачи;
- Увеличение объема решаемой задачи приводит к увеличению доли параллельной части, т.к. последовательная часть не изменяется.

Масштабируемость алгоритмов

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

- При анализе масштабируемости необходимо учитывать *накладные расходы (total overhead)*, на организацию взаимодействия процессоров, синхронизацию параллельных вычислений и др.

Анализ масштабируемости

- Накладные расходы $T_0 = pT_p - T_1$;
- Время решения задачи $T_p = \frac{T_1 + T_0}{p}$;
- Ускорение $S_p = \frac{T_1}{T_p} = \frac{pT_1}{T_1 + T_0}$;
- Эффективность $E_p = \frac{S_p}{p} = \frac{T_1}{T_1 + T_0} = \frac{1}{1 + \frac{T_0}{T_1}}$.
- Если сложность решаемой задачи является фиксированной (T_1=*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.

- Если сложность решаемой задачи является фиксированной (T_1=*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.