ОЦЕНКА ЭФФЕКТИВНОСТИ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ

В наш век передовой техники неэффективность и непроизводительность есть грех перед Святым Духом.

О. Хаксли

Суперкомпьютеры и их применение

Содержание

- Показатели эффективности параллельного алгоритма
  - Ускорение
  - Эффективность
  - Стоимость
- Оценка максимально достижимого параллелизма
  - Закон Амдала
  - Закон Густавсона
- 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.
  - $$E_p(n) = \frac{T_1(n)}{p \cdot T_p(n)} = \frac{S_p(n)}{p}$$

Ускорение vs эффективность

- Ускорение и эффективность – 2 стороны одной медали: попытки повышения качества параллельных вычислений по одному из показателей может привести к ухудшению качества по другому показателю.

Стоимость вычислений

- *Стоимость (cost)* параллельных вычислений

$$C_p = p \cdot T_p$$

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

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.

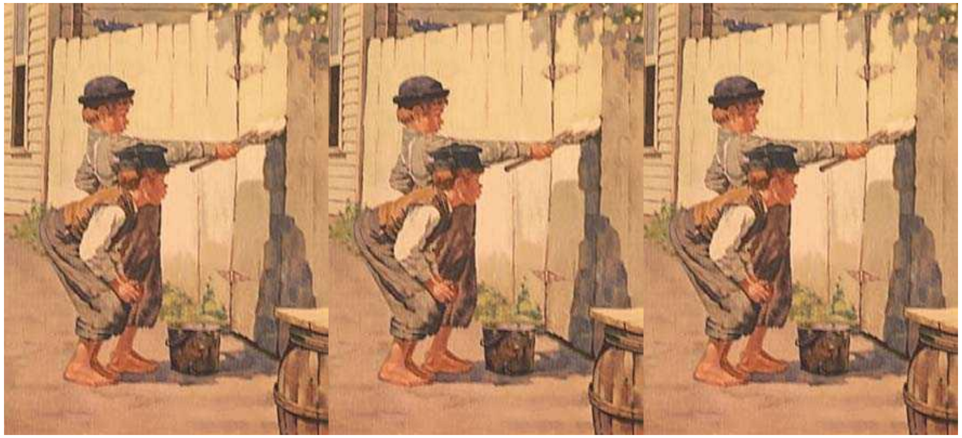
Закон Амдала

Джин Амдал
(р. 1922)

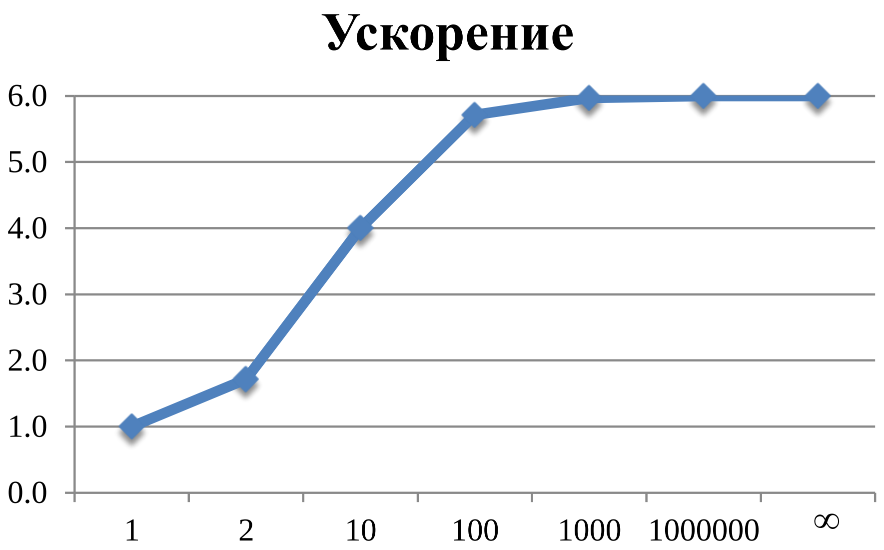
- Задает связь между ожидаемым ускорением параллельных реализаций алгоритма и последовательным алгоритмом в предположении, что размер задачи остается постоянным.
- Пусть f – доля последовательных вычислений в алгоритме. Тогда
  - $S_p = \frac{T_1}{T_p} = \frac{f + (1-f)}{f + \frac{1-f}{p}} = \frac{1}{f + \frac{1-f}{p}}$ т.е. $\boxed{S_p = \frac{1}{f + \frac{1-f}{p}}}$
  - $\boxed{\lim_{p\to\infty} S_p = \frac{1}{f}}$

Закон Амдала

- Покраска забора (300 досок)
  - *Подготовка* – 30 мин.
    НЕ распараллеливается
  - *Покраска* (одной доски) – 1 мин.
    РАСПАРАЛЛЕЛИВАЕТСЯ
  - *Уборка* – 30 мин.
    НЕ распараллеливается

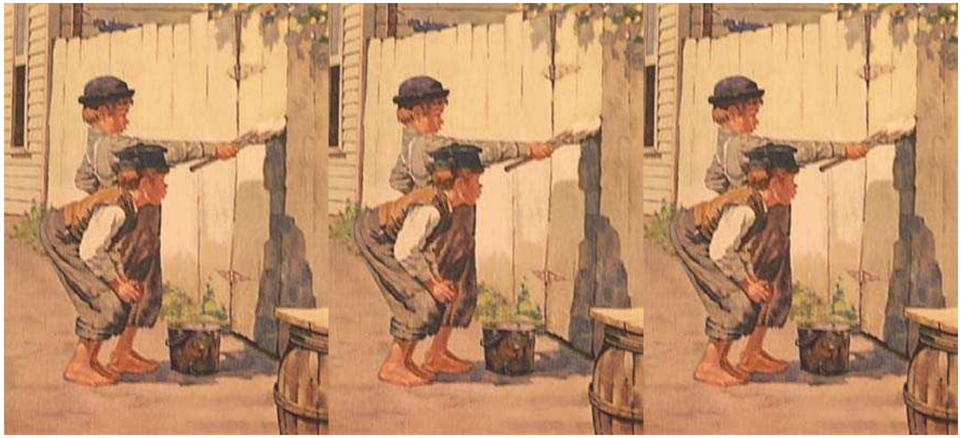


Количество маляров	Время покраски	
1	30 + 300/1	+ 30 = 360
2	30 + 300/2	+ 30 = 210
10	30 + 300/10	+ 30 = 90
100	30 + 300/100	+ 30 = 63
1000	30 + 300/1000	+ 30 ≅ 60
1000000	30 + 300/1000000	+ 30 ≅ 60



Закон Амдала

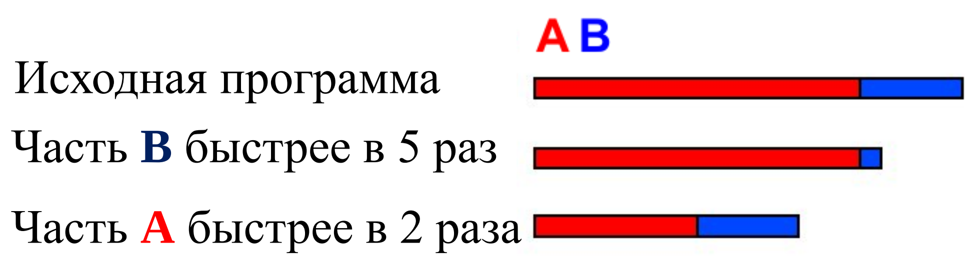
- Покраска забора (300 досок)
  - *Подготовка* – 30 мин.
    НЕ распараллеливается
  - *Покраска* (одной доски) – 1 мин.
    РАСПАРАЛЛЕЛИВАЕТСЯ
  - *Уборка* – 30 мин.
    НЕ распараллеливается



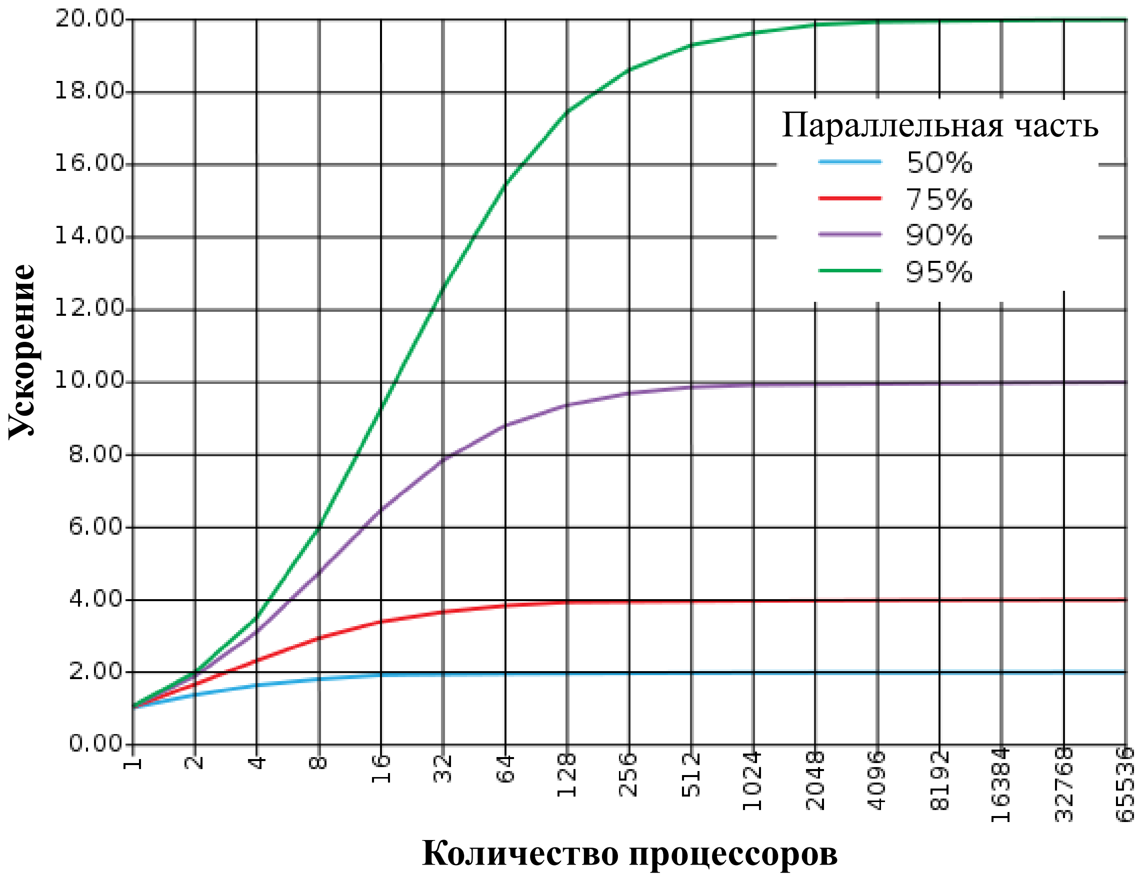
Количество маляров	Время покраски
1	30 + 300/1 + 30 = 360
2	30 + 300/2 + 30 = 210
10	30 + 300/10 + 30 = 90
100	30 + 300/100 + 30 = 63
1000	30 + 300/1000 + 30 ≅ 60
1000000	30 + 300/1000000 + 30 ≅ 60

Ускорение последовательной программы

- Ускорение улучшенной последовательной программы (быстродействие некоторой ее части увеличено в *p* раз) $\frac{p}{1+f*(p-1)}$, где *f* – доля времени работы не улучшенной части программы (до улучшения).
  - Пусть имеется последовательная программа, состоящая из двух независимых частей: **A** (75%) и **B** (25%).
  - Увеличение быстродействия части **B** *в 5 раз* даст ускорение $\frac{5}{1+0.75*(5-1)} = 1.25$
  - Увеличение быстродействия части **A** *в 2 раза* даст ускорение $\frac{2}{1+0.25*(2-1)} = 1.60$

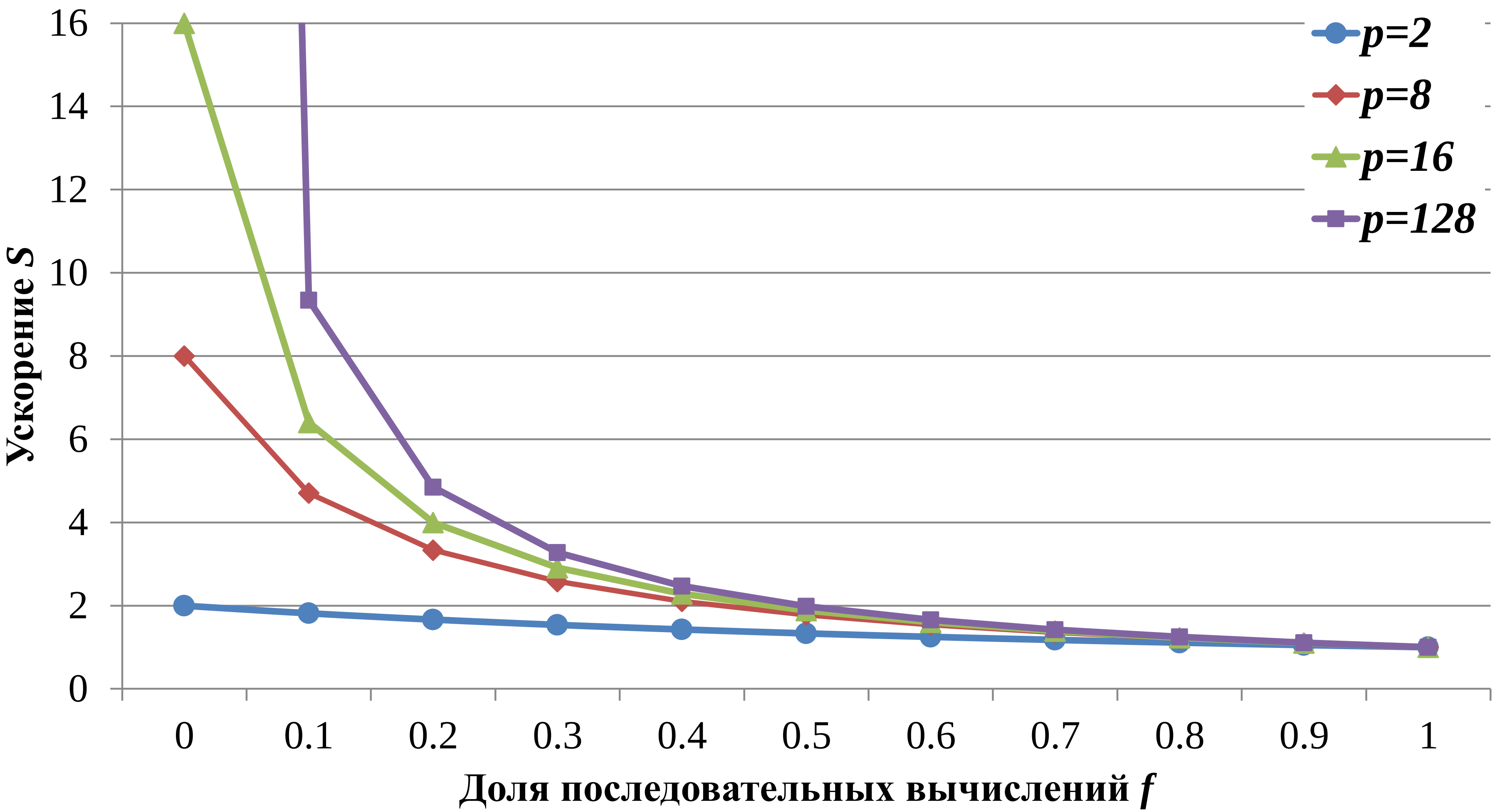


Закон Амдала



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

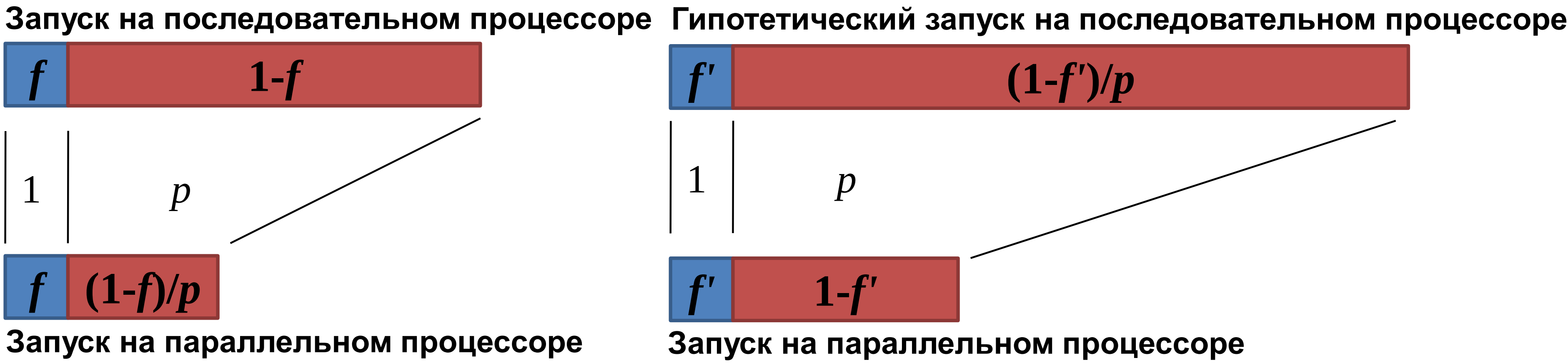
Закон Амдала



Закон Густавсона

- 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.

Дж. Густавсон
(р. 1955)



Закон Густавсона

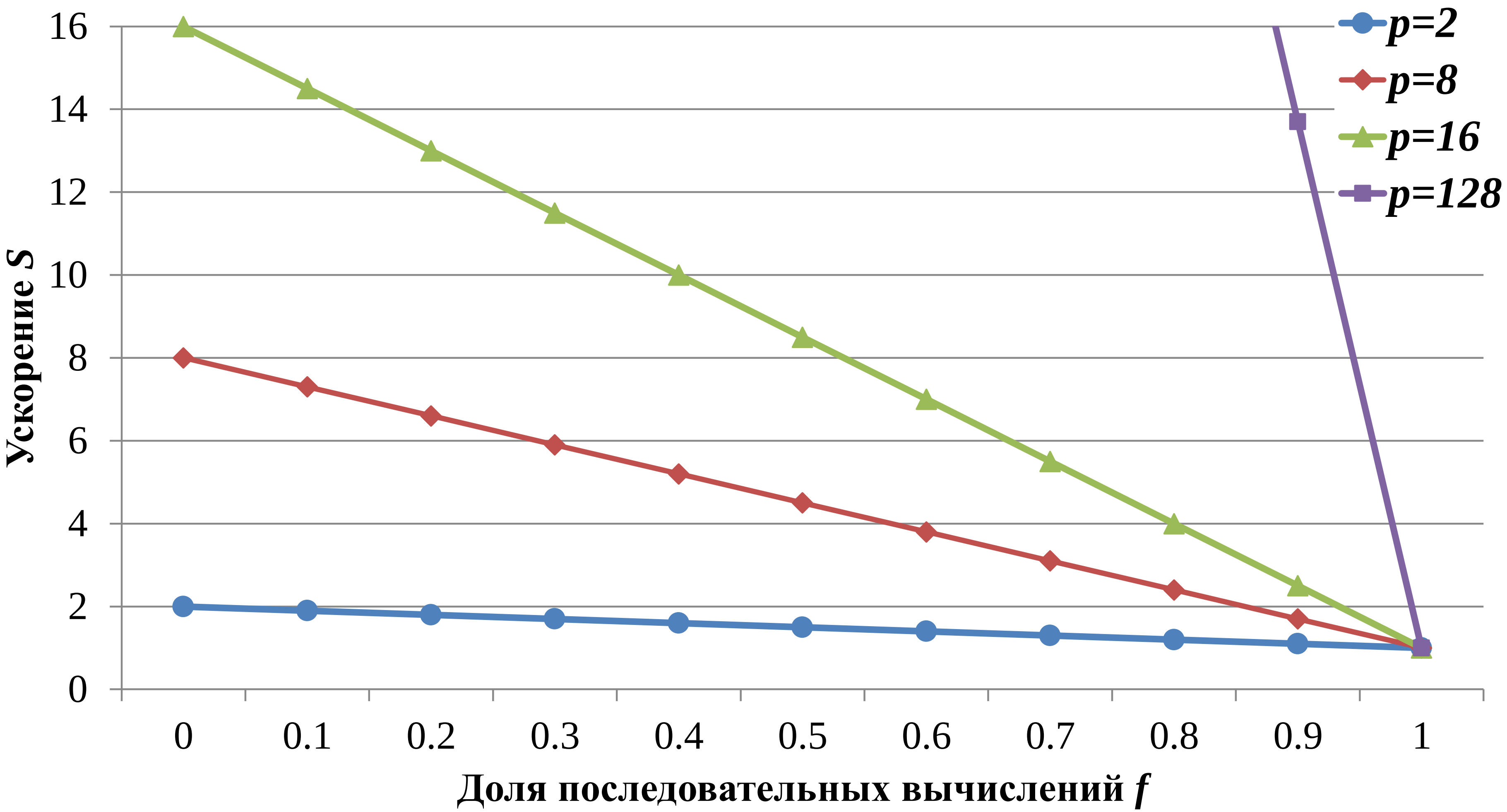
- Закон Амдала предполагает, что количество процессоров и доля параллельной части программы независимы, что не совсем верно.
  - Как правило, задача с фиксированным объемом данных не запускается на различном количестве процессоров (за исключением академических исследований), а объем данных изменяется в соответствии с количеством процессоров.
  - Вместо вопроса об ускорении на *p* процессорах рассмотрим вопрос о замедлении вычислений при переходе на один процессор.

Дж. Густавсон
(р. 1955)

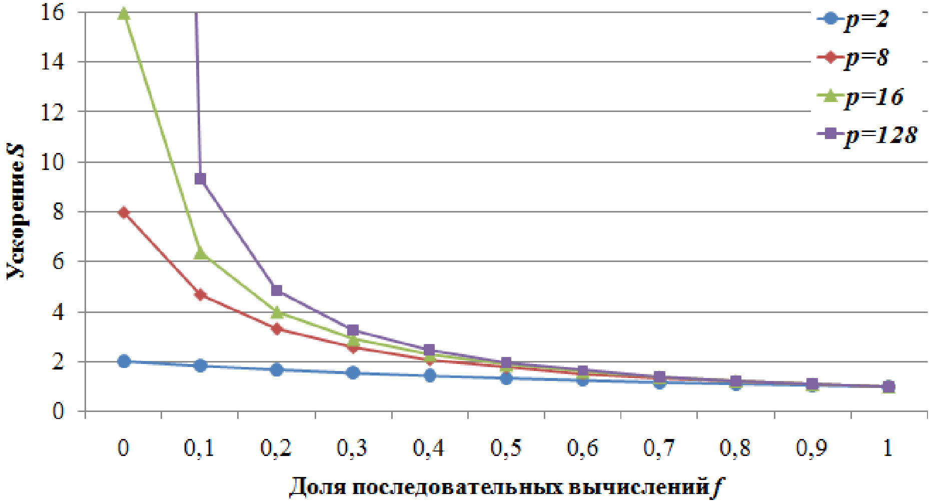
$$S_p = \frac{f + (1-f)\cdot p}{f + (1-f)} = f + p - f\cdot p = p - f\cdot(p-1)$$

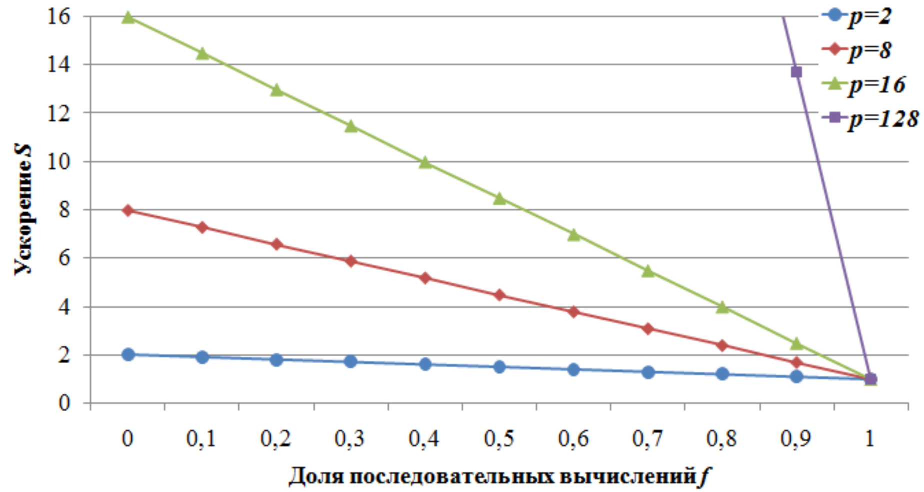
$$\boxed{S_p = p - f\cdot(p-1)}$$

Закон Густавсона



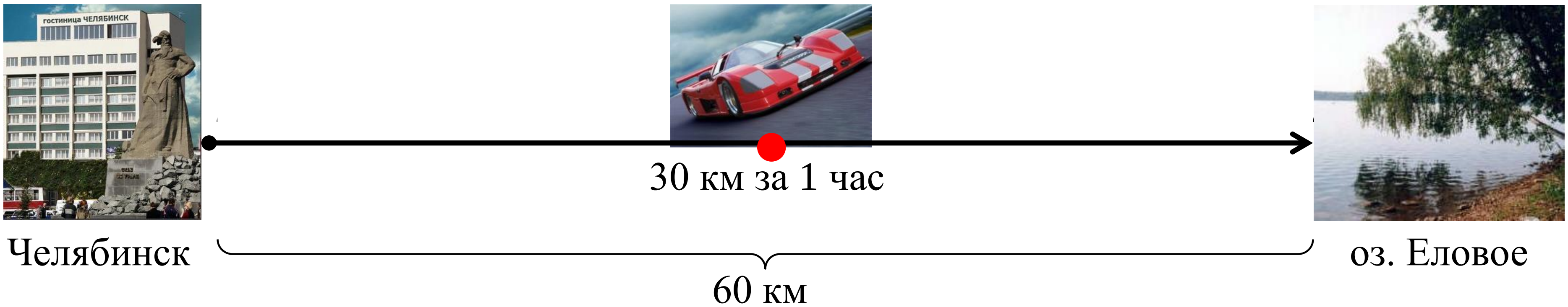
Законы Амдала и Густавсона





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

Законы Амдала и Густавсона



- Закон Амдала
  - Независимо от того, как быстро будет ехать машина вторую половину пути, невозможно достигнуть средней скорости 90 км/ч до приезда в пункт назначения
    - Например, до приезда даже очень быстрая езда позволит достигнуть лишь средней скорости 60 км/ч.
- Закон Густавсона
  - Независимо от того, как долго или как медленно двигалась машина первую половину пути, при наличии достаточного количества времени и протяженности дороги средняя скорость машины в конечном итоге всегда достигнет значения 90 км/ч
    - Например, при скорости 150 км/ч на второй половине пути по приезде в пункт назначения средняя скорость равна 90 км/ч .

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 $T_p = \dfrac{T_1 + T_0}{p}$
- Ускорение $S_p = \dfrac{T_1}{T_p} = \dfrac{pT_1}{T_1 + T_0}$
- Эффективность $E_p = \dfrac{S_p}{p} = \dfrac{T_1}{T_1 + T_0} = \dfrac{1}{1 + \dfrac{T_0}{T_1}}$

Анализ масштабируемости

- 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.

Анализ масштабируемости

- Пусть *E=const* – это желаемый уровень эффективности выполняемых вычислений. Тогда

$$\frac{T_0}{T_1} = \frac{1-E}{E}, \text{ или } T_1 = kT_0, \text{ где } k = \frac{E}{1-E}$$

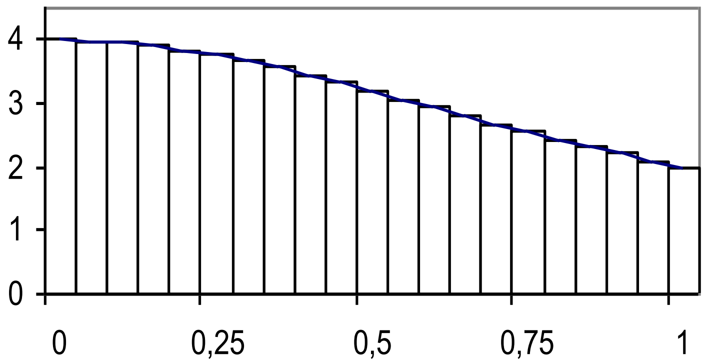
- Данную зависимость *n=F(p)* между сложностью решаемой задачи и числом процессоров называют *функцией изоэффективности* (*isoefficiency function*).

Пример

- Значение числа π может быть получено при помощи интеграла

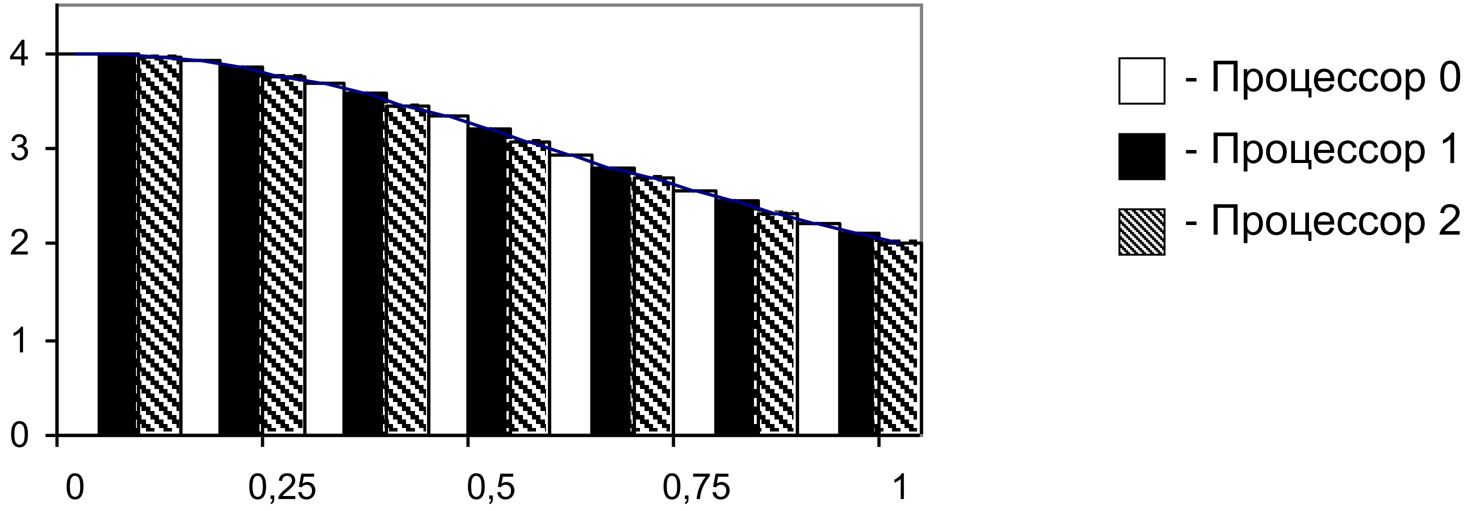
$$\pi = \int_0^1 \frac{4}{1+x^2}\,dx$$

- Численное интегрирование выполняется методом прямоугольников



Пример

- Распределяем вычисления циклически между *p* процессорами.
- Суммируем получаемые на отдельных процессорах частные суммы.



Пример

- n – количество разбиений отрезка [0;1]
- Вычислительная сложность задачи $W=T_1=6n$
- Количество узлов сетки на отдельном процессоре $m=\lceil n/p \rceil \le n/p + 1$
- Объем вычислений на отдельном процессоре $W_p=6m=6n/p+6$

Пример

- Время параллельного решения задачи $T_p\text{=}6n/p\text{+}6\text{+}log_2 p$
- Ускорение $S_p\text{=}T_1/T_{\mathrm{p}}\text{=}6n/(6n/p\ \text{+}6\text{+}log_2 p)$
- Эффективность $E_p\text{=}6n/(6n\text{+}6p\text{+}plog2p)$
- Функция изоэффективности $W\text{=}k(pTp\text{-}W)\text{=}k(6p + plog2p)$

Заключение

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