Московский государственный университет имени М.В.Ломоносова
Суперкомпьютерный консорциум университетов России

Эффективность параллельных программ

А.С.Антонов
Вед. н.с. НИВЦ МГУ, к.ф.-м.н.
asa@parallel.ru

Летняя суперкомпьютерная академия
Москва, 2015

Важные сокращения

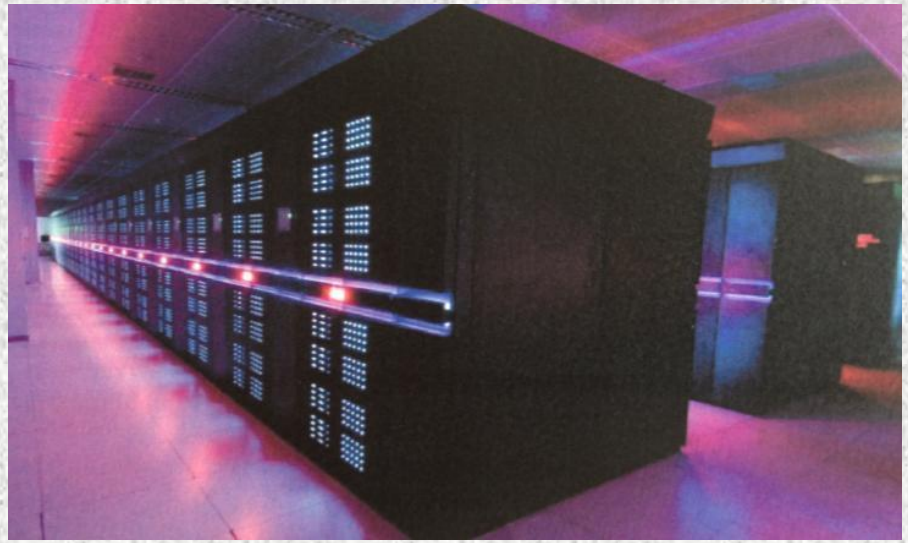
Мега (Mega)	– 10^6	(миллион)
Гига (Giga)	– 10^9	(биллион / миллиард)
Тера (Tera)	– 10^{12}	(триллион)
Пета (Peta)	– 10^{15}	(квадриллион)
Экза (Exa)	– 10^{18}	(квинтиллион)

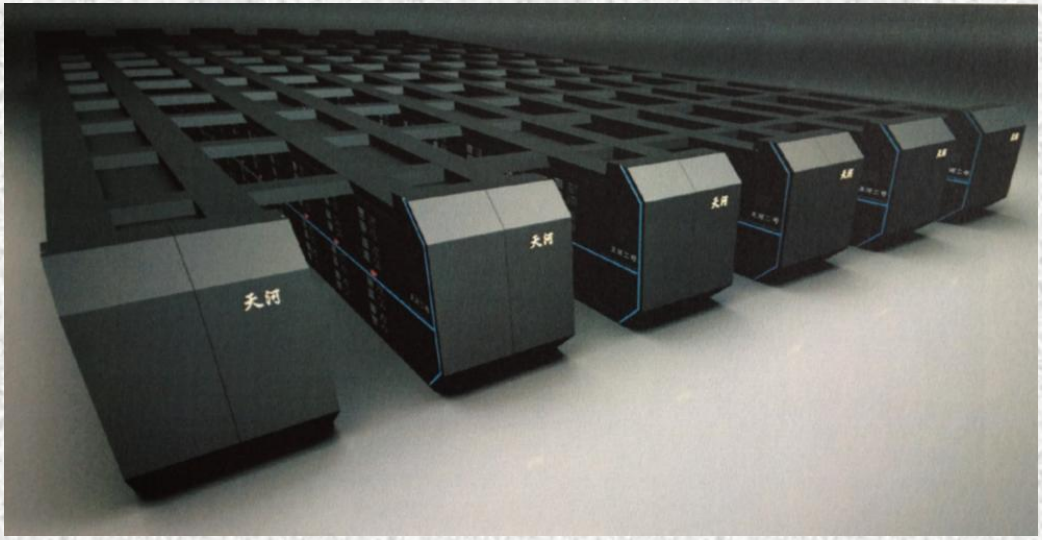
Флоп/с, Flop/s – Floating point operations per second

15 Tflop/s = $15 * 10^{12}$ арифметических операций в секунду над вещественными данными, представленными в форме с плавающей точкой.

Суперкомпьютер “Tianhe-2”, Китай

(#1 Top500 в 2013-2014 г.)





16 000 вычислительных узлов

32 000 Intel Xeon IvyBridge, 12-core
48 000 Intel Xeon Phi

Всего: 3 120 000 ядер

Производительность:
Peak: 54,9 Pflop/s
Linpack: 33,86 Pflop/s

Суперкомпьютеры и их характеристики
(2014 г.)

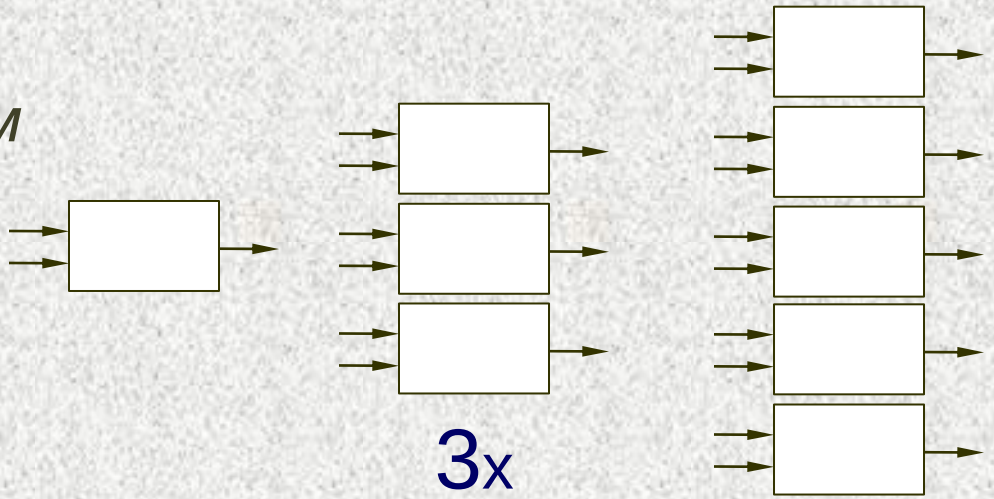
- *Tianhe-2, 16 000 вычислительных узлов:*
  2 × Intel XeonE5-2692 (12cores), 2.93GHz + 3 × Xeon Phi,
  33.86 Pflop/s, ОП = 1,375 PB + 375 TB, HDD = 12 PB
- *Cray XK7, Titan, 18 688 вычислительных узлов:*
  AMD Opteron 16-core + NVIDIA Tesla K20
  17.59 Pflop/s, ОП = 710 TB, HDD = 10 PB
- *IBM BlueGene/Q, Sequoia, 1 572 864 ядра, IBM PowerPC A2, 1.6 GHz*
  17.17 Pflop/s, ОП = 1.57 PB, HDD = 55 PB, Tape = “virtually unlimited”
- *IBM RoadRunner, 6562 AMD Opteron DC + 12240 IBM Cell,*
  1.042 Pflop/s, ОП = 98 TB
- *Т-Платформы, Ломоносов, 52 168 x86-cores, 2 130 NVIDIA X2070*
  Intel Xeon 5570 (4cores), 5670 (6cores) 2.93 GHz
  901 Tflop/s, ОП = 92 TB, HDD = 650 TB, Tape = 1.2 PB

Два вида параллельной обработки

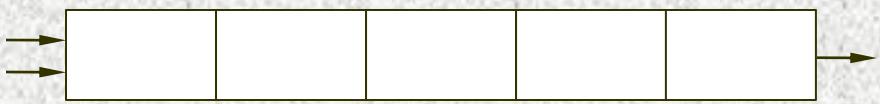
- *Параллелизм*
- *Конвейерность*

Параллелизм в архитектуре компьютеров

- *Параллелизм*



- *Конвейерность*



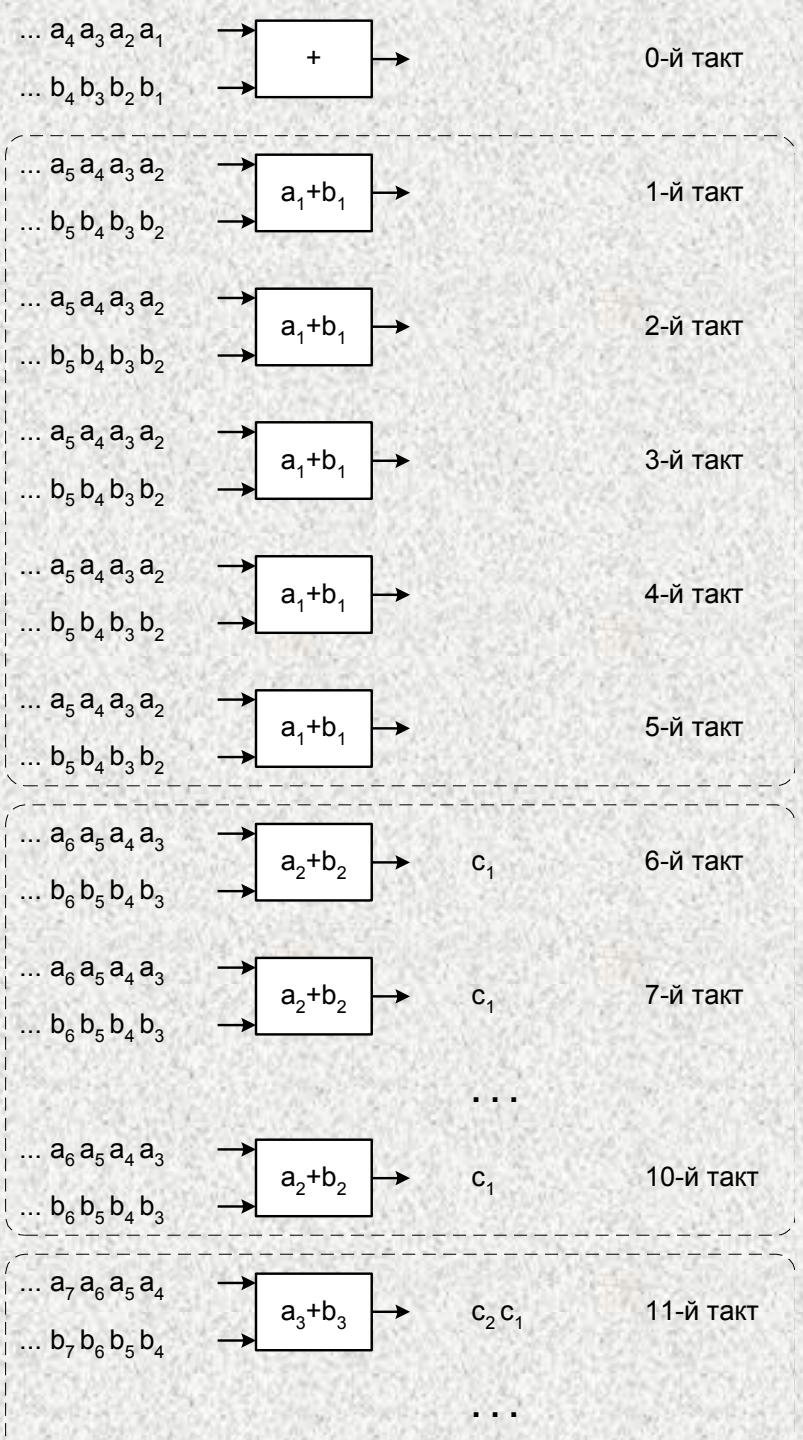
Последовательная обработка данных

Суммирование векторов C = A + B с помощью 1 последовательного устройства.

Устройство выполняет 1 операцию за 5 тактов.

Векторы A и B содержат по 100 элементов.

Время выполнения данной операции: 500 тактов.



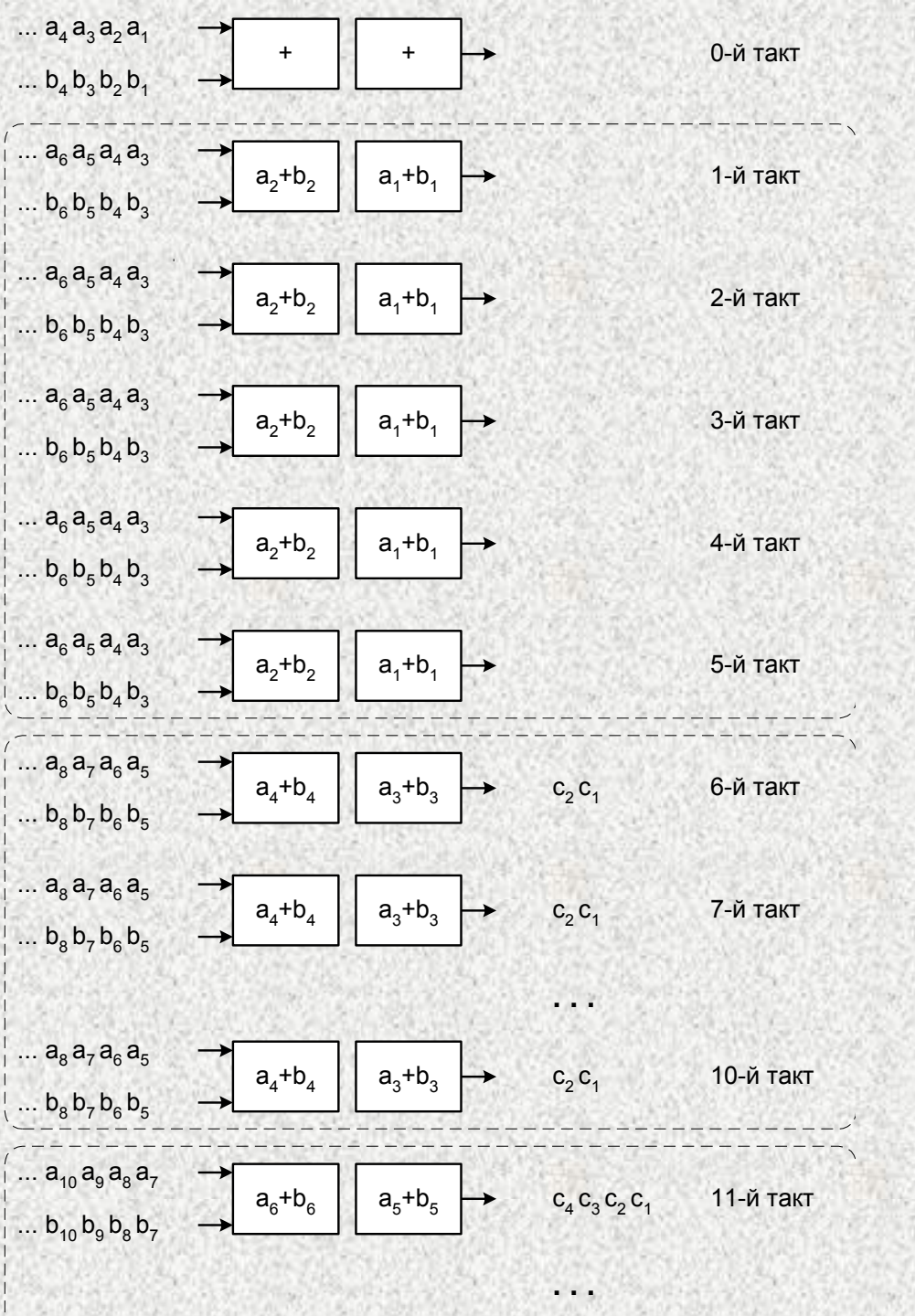
Параллельная обработка данных

Суммирование векторов C = A + B с помощью 2 одинаковых последовательных устройств.

Каждое устройство выполняет по 1 операции за 5 тактов.

Векторы A и B содержат по 100 элементов.

Время выполнения данной операции: 250 тактов.



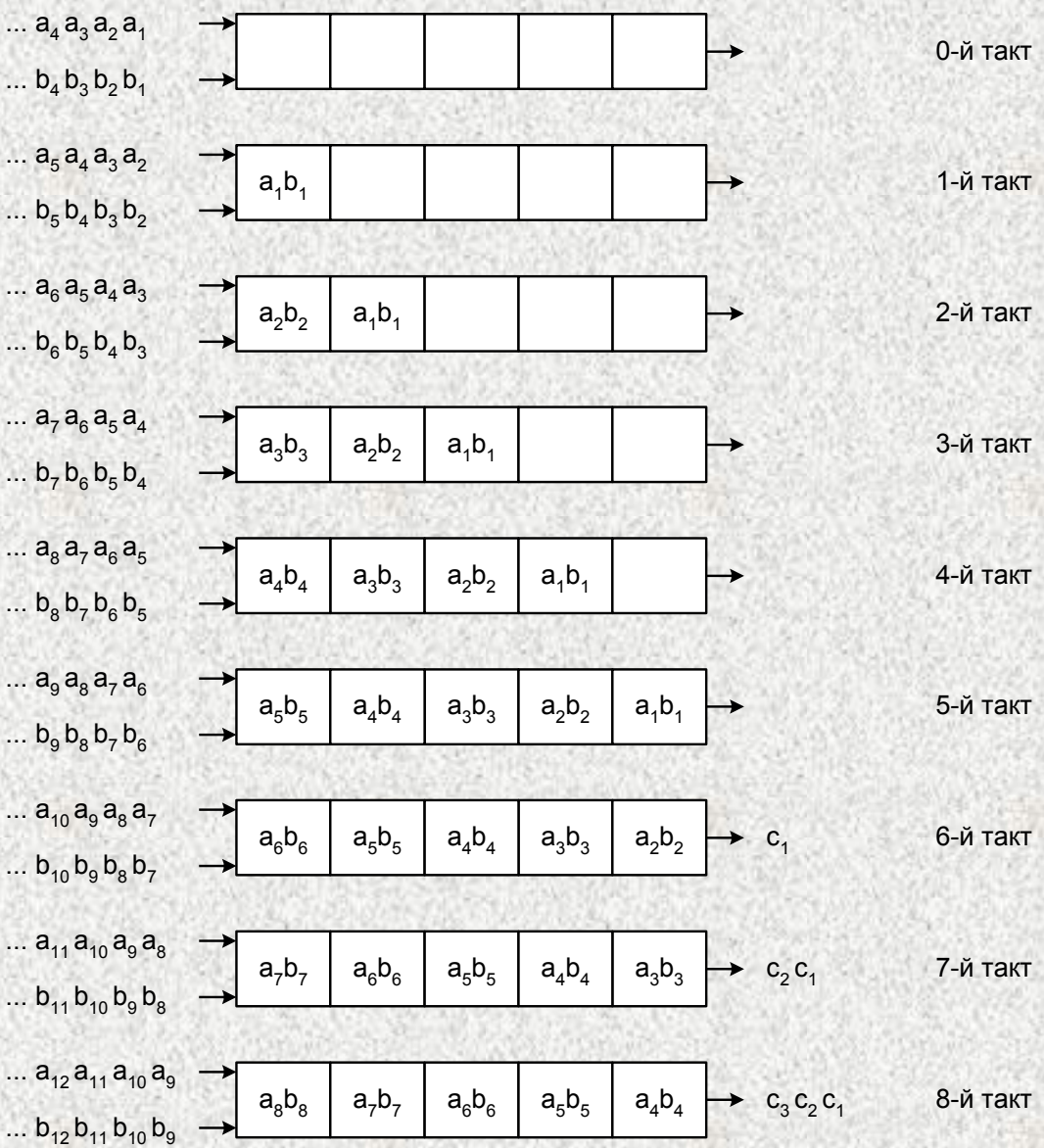
Конвейерная обработка данных

Суммирование векторов C = A + B с помощью конвейерного устройства.

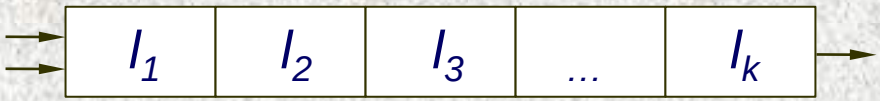
Конвейерное устройство состоит из 5 ступеней. Каждая из 5 ступеней срабатывает за 1 такт.

Векторы A и B содержат по 100 элементов.

Время выполнения данной операции: 104 такта.



Конвейерная обработка

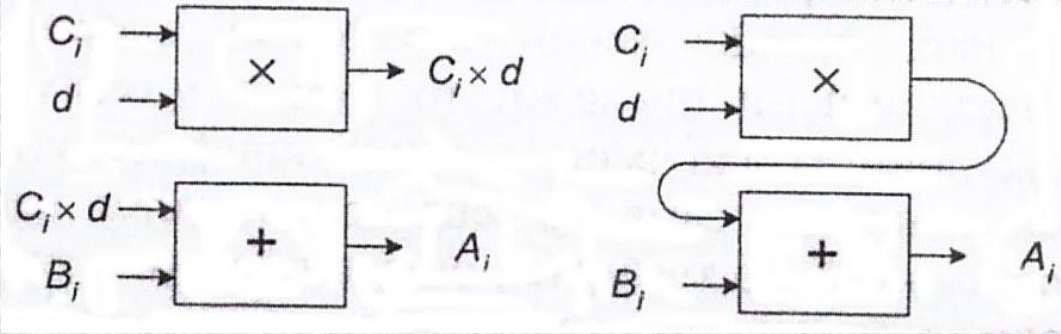


Конвейер состоит из k ступеней, время срабатывания i-ой ступени l_i.

- *Время разгона конвейера:* $\sum_{i=1}^{k} l_i$ *тактов*
- *Каждый следующий результат будет получен через* $\max_{i=1}^{k} l_i$ *тактов*
- *Время выполнения всей операции над векторами длины n:*

$$\sum_{i=1}^{k} l_i + (n-1) \times \max_{i=1}^{k} l_i$$

Зацепление конвейерных устройств



$A_i = B_i + C_i * d$
n – длина векторов

× – l_1 ступеней, срабатывающих за 1 такт
+ – l_2 ступеней, срабатывающих за 1 такт

- Без зацепления: $(l_1+n-1) + (l_2+n-1) = l_1+l_2+2*n-2$ тактов
- С зацеплением: l_1+l_2+n-1 тактов

Параллельные вычисления – есть ли здесь проблемы ?

Производительность компьютера и время решения задачи

Вскапываем огород, 10×10 м – все хорошо: если нужно ускориться, то зовем 5, 10, 50 помощников…

Копаем яму, 1×1×1 м – как ускорить процесс? Еще 10 или 50 помощников сильно время не уменьшат…

Копаем фундамент: разметку может делать только бригадир, остальное – рабочие.

- *1 час разметка, остальное - 10 часов (1 рабочий) = 11 часов*
- *1 час разметка, остальное - 1 час (10 рабочих) = 2 часа*
- *1 час разметка, остальное - 6 мин. (100 рабочих) = 1 час 6 мин.*

…бригадир становится “узким местом”.

Закон Амдала

f - доля последовательных операций ($0 \le f \le 1$)
p - число процессоров

$$\frac{T_1}{T_p} = S \le \frac{1}{f+(1-f)/p}$$

T_1 – время работы программы на одном процессоре
T_p – время работы программы на системе из p процессоров
S – ускорение работы программы

Закон Амдала. Следствие

$$S \approx \frac{1}{f}$$ *(при большом числе процессоров)*

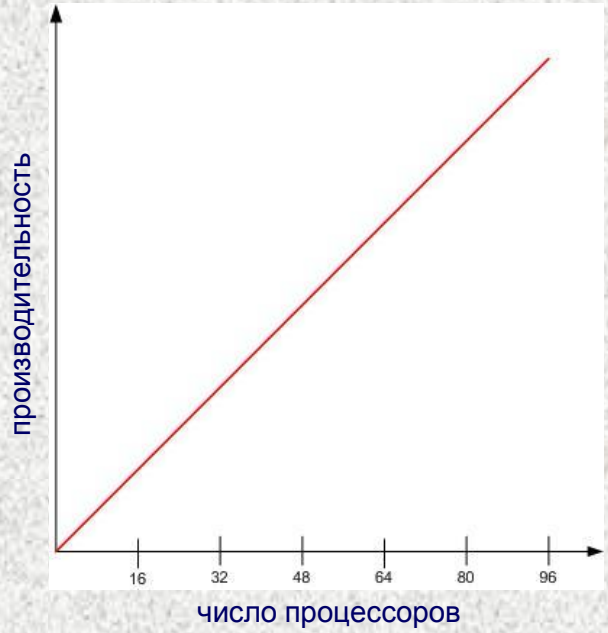
На практике. Если доля последовательных операций в некоторой программе равна 0.1, значит вне зависимости от числа используемых процессоров ускорение не превысит 10.

Закон Амдала. Следствие

Для того чтобы ускорить программу в q раз, необходимо ускорить не менее, чем в q раз не менее, чем (1-1/q)-ю часть программы.

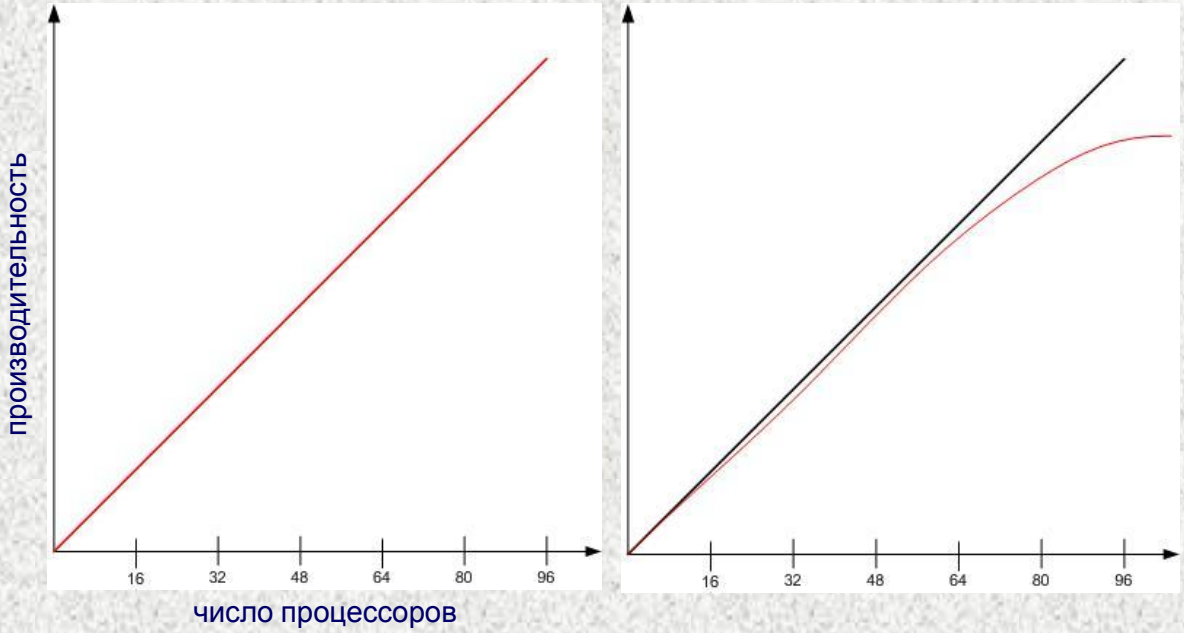
На практике. Нужно ускорить работу программы в 100 раз. Значит необходимо ускорить не менее, чем в 100 раз не менее, чем 99% этой программы.

Рост производительности параллельных вычислительных систем
(в теории)



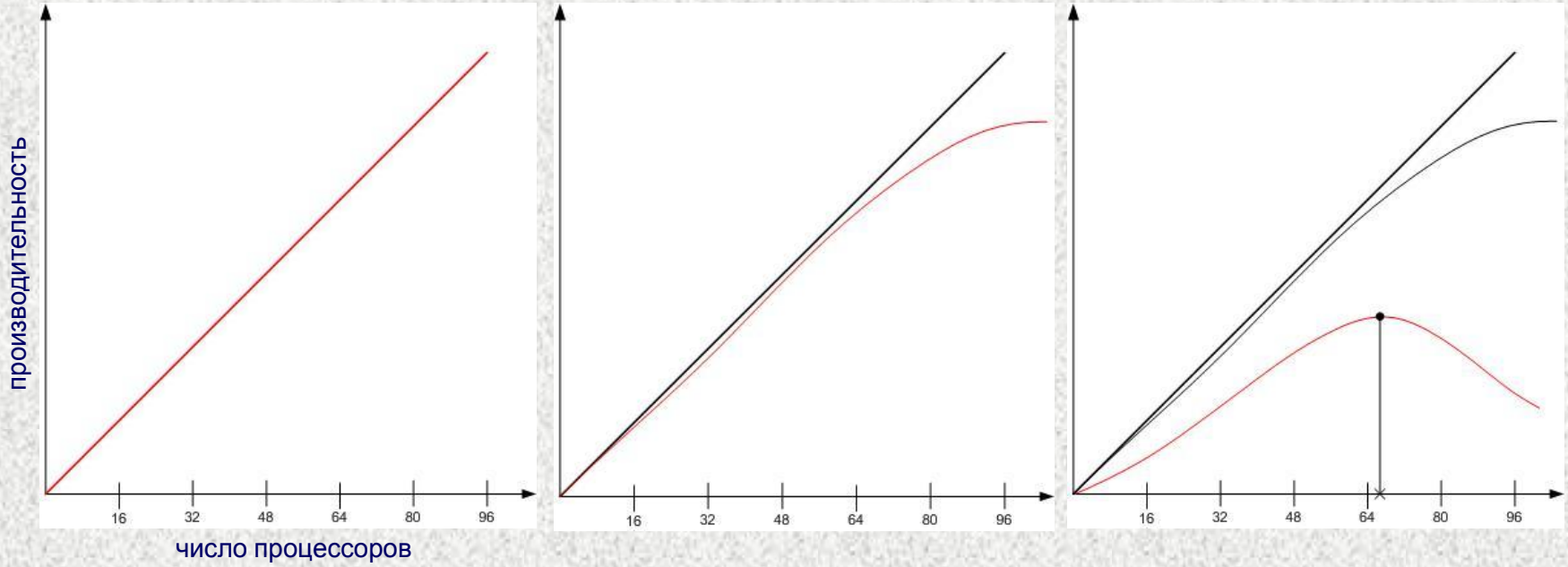
Рост производительности параллельных вычислительных систем

(в теории и на практике)



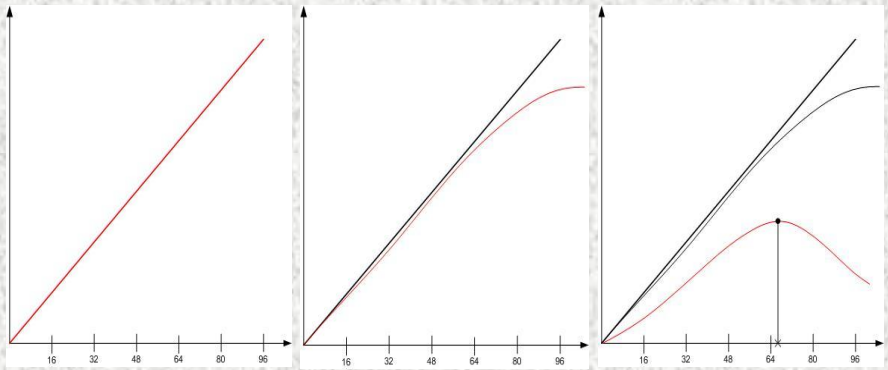
Рост производительности параллельных вычислительных систем

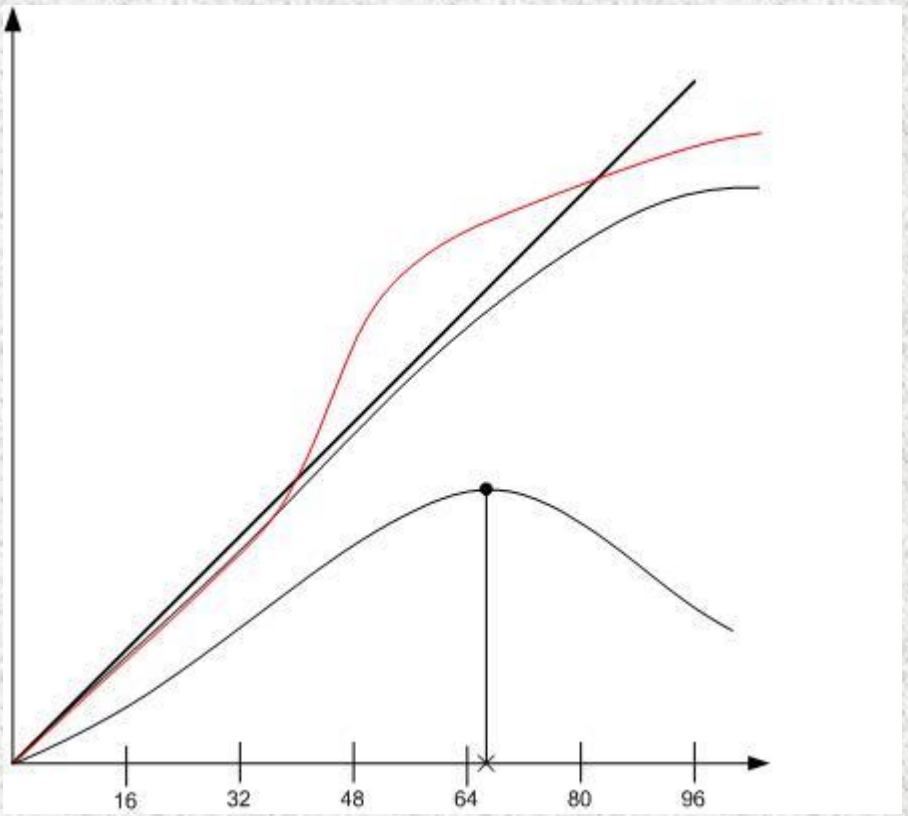
(в теории и на практике)



Рост производительности параллельных вычислительных систем

(в теории и на практике)

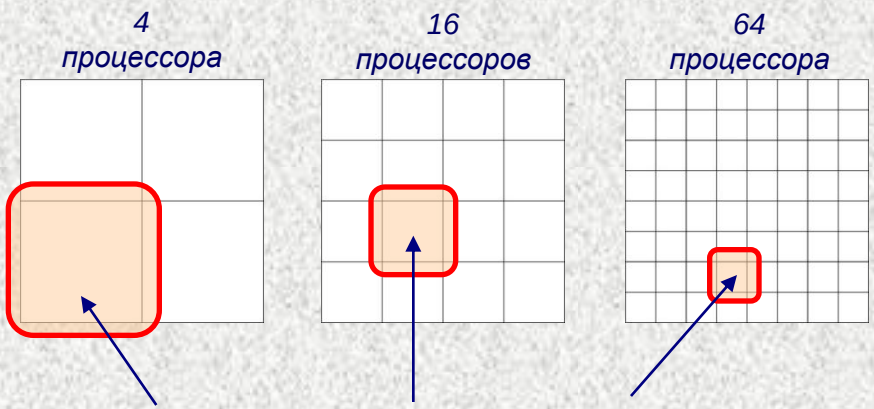




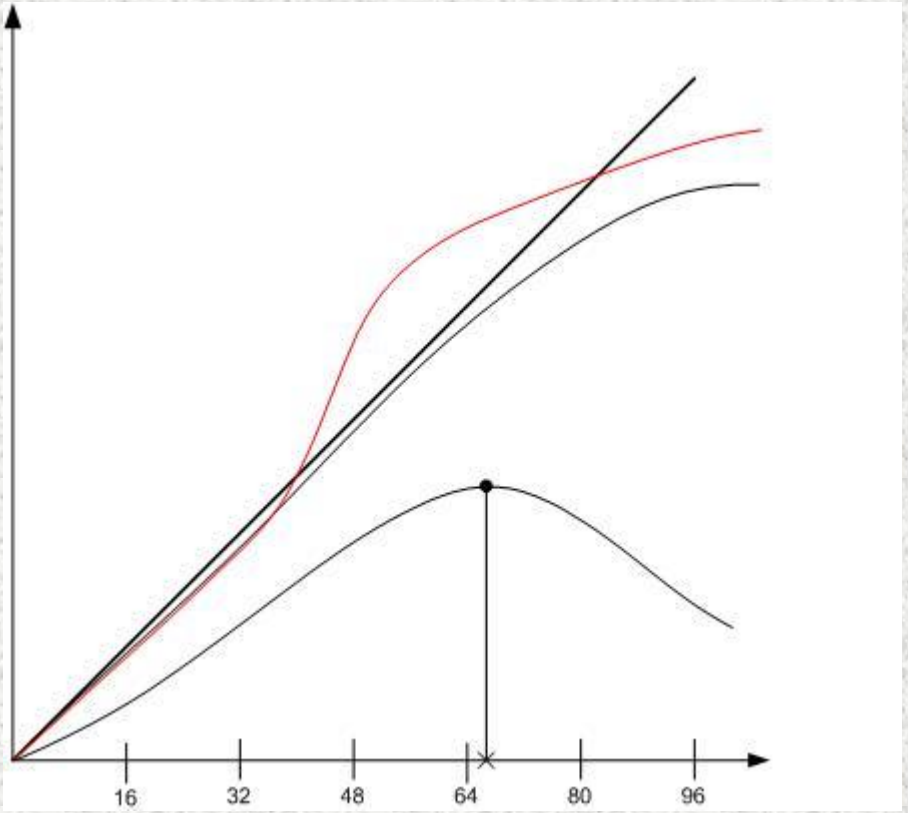
Рост производительности параллельных вычислительных систем

(в теории и на практике, суперлинейное ускорение)

Исходная расчетная область



Часть расчетной области, которая приходится на каждый процессор



Эффективность реализации

Пиковая производительность компьютера, R_{peak} – теоретический максимум производительности данного компьютера.

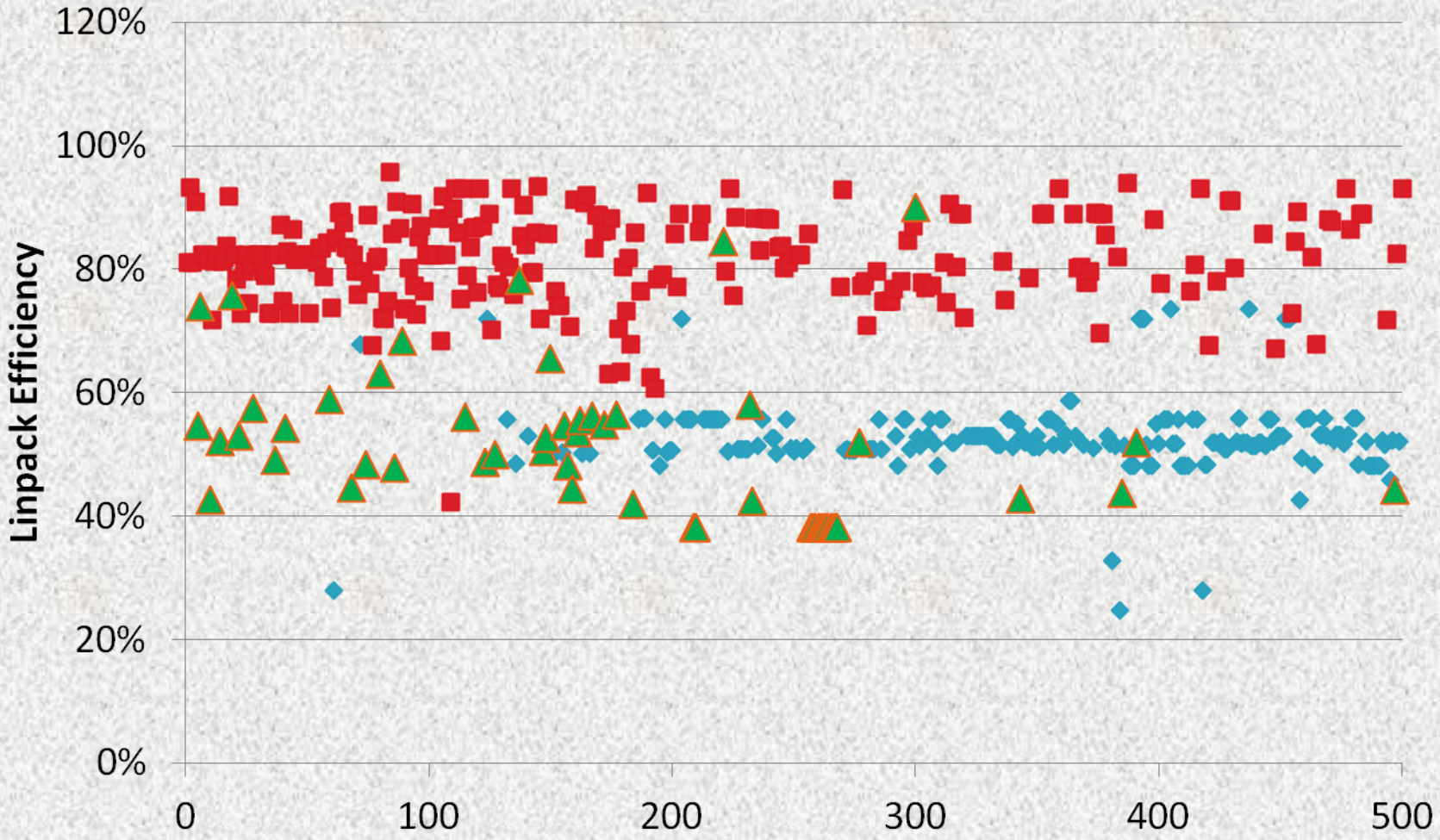
Реальная производительность, R_{max} – достигнутая производительность на некоторой программе или тесте.

$$R_{max} = \frac{\text{число операций в программе}}{\text{время работы компьютера}}$$

$$R_{max} << R_{peak}$$

Эффективность реализации $E_{real} = \frac{R_{max}}{R_{peak}}$

Top500, Linpack, Эффективность



Основные показатели эффективности и масштабируемости параллельных программ

Основные обозначения:

- p — число процессоров (процессорных ядер).
- T_1 — время работы программы на одном процессоре.
- T_p — время работы программы на p процессорах.

Основные показатели эффективности и масштабируемости параллельных программ

- *Ускорение* (*speedup*) $S = T_1/T_p$, где T_p — время исполнения распараллеленной программы на p процессорах, T_1 — время исполнения исходной программы.
- В идеальном случае (отсутствие накладных расходов на организацию параллелизма) получаем $S = p$ – *линейное ускорение*.
- *Суперлинейное ускорение* $S > p$.

Основные показатели эффективности и масштабируемости параллельных программ

- *Эффективность реализации* программы $E_{real}=R_{max}/R_{peak}$ определяется как отношение реальной производительности R_{max} к пиковой производительности R_{peak}.
- Поскольку пиковая производительность недостижима на практике, эффективность реализации программы всегда меньше единицы.
- Чем ближе этот показатель к единице, тем лучше для пользователя, поскольку говорит о том, что более эффективно задействованы ресурсы компьютера.

Основные показатели эффективности и масштабируемости параллельных программ

- *Эффективность распараллеливания* $E_{par} = S/p$ определяет среднюю долю времени выполнения параллельного алгоритма, в течение которого процессоры реально используются для решения задачи.
- Оценка качества распараллеливания предполагает получение наилучших (максимальных) значений ускорения и эффективности распараллеливания.
- Получение большого ускорения за счёт большого числа процессоров зачастую приводит к снижению эффективности распараллеливания.

Основные показатели эффективности и масштабируемости параллельных программ

- *Стоимость (cost)* вычислений $C = pT_p$.
- $T_0 = pT_p - T_1$ — суммарные *накладные расходы (total overhead)*. При увеличении p значение, как правило, возрастает.
- $T_p = (T_1 + T_0)/p$
- $S = T_1/T_p = pT_1/(T_1 + T_0)$

Основные показатели эффективности и масштабируемости параллельных программ

- $E_{par} = S/p = T_1/(T_1+T_0)=1/(1+T_0/T_1)$
- Если T_1 фиксировано, то при увеличении числа процессоров p эффективность распараллеливания E_{par}, как правило, уменьшается за счёт роста накладных расходов T_0.
- Если фиксировано число процессоров p, то эффективность распараллеливания E_{par} можно увеличить, увеличивая сложность решаемой задачи T_1.

Основные показатели эффективности и масштабируемости параллельных программ

- *Масштабируемость* (*scalability*) — способность системы увеличивать свою производительность при добавлении ресурсов (обычно аппаратных).
- Система называется *масштабируемой*, если она способна увеличивать производительность пропорционально дополнительным ресурсам.

Основные показатели эффективности и масштабируемости параллельных программ

- Масштабируемость можно оценить через отношение прироста производительности системы к приросту используемых ей ресурсов.
- Чем ближе это отношение к единице, тем масштабируемость лучше.

Основные показатели эффективности и масштабируемости параллельных программ

Масштабируемость:

- Компьютера или его компонент (например, коммуникационной сети).
- Алгоритмов безотносительно к компьютеру.
- Параллельных программ относительно данного компьютера.

Основные показатели эффективности и масштабируемости параллельных программ

Масштабируемость компьютера:

- *Вертикальная масштабируемость* (*масштабируемость вглубь, scale up*) – возможность замены платформы, в которой функционирует система, на новую, обладающую большей производительностью.
- *Горизонтальная масштабируемость* (*масштабируемость вширь, scale out*) – возможность увеличения производительности системы за счет добавления дополнительных программных или аппаратных средств.

Основные показатели эффективности и масштабируемости параллельных программ

- *Вычислительная сложность задачи* W – количество основных вычислительных шагов лучшего последовательного алгоритма, необходимых для решения задачи на одном процессоре.
- W является некоторой функцией от размера входных данных.
- Если для простоты предположить, что каждый основной вычислительный шаг выполняется за единицу времени, то получим $W = T_1$.

Основные показатели эффективности и масштабируемости параллельных программ

Примеры:

- Для сложения N чисел:

$W = N-1$

- Для скалярного произведения векторов:

$W = 2N-1$

- Для перемножения матриц:

$W = N^2(2N-1)$

Основные показатели эффективности и масштабируемости параллельных программ

Масштабируемость параллельной программы определяется относительно конкретного компьютера и показывает, как изменяются динамические характеристики данной программы при использовании бóльших вычислительных ресурсов.

Основные показатели эффективности и масштабируемости параллельных программ

Масштабируемость параллельных программ:

- *Сильная масштабируемость* (*strong scaling*) — зависимость производительности R от количества процессоров p при фиксированной вычислительной сложности задачи (W = const).

Основные показатели эффективности и масштабируемости параллельных программ

Масштабируемость параллельных программ:

- *Масштабируемость вширь (wide scaling)* – зависимость производительности R от вычислительной сложности задачи W при фиксированном числе процессоров (p = const).

Основные показатели эффективности и масштабируемости параллельных программ

Масштабируемость параллельных программ:

- *Слабая масштабируемость* (*weak scaling*) — зависимость производительности R от количества процессоров p при фиксированной вычислительной сложности задачи в пересчёте на один процессор (W/p = const).

Основные показатели эффективности и масштабируемости параллельных программ

- Нужна метрика масштабируемости.
- Для многих задач при увеличении вычислительной сложности задачи W (а, следовательно, и времени T_1) эффективность распараллеливания E_{par} растёт.
- Если при одновременном увеличении числа процессоров p и вычислительной сложности задачи W эффективность распараллеливания E_{par} остаётся прежней, данную задачу на данном компьютере можно считать *масштабируемой*.

Основные показатели эффективности и масштабируемости параллельных программ

Функция изоэффективности **(***isoefficiency function***)**:

- Определим, в какой степени должна увеличиваться вычислительная сложность задачи W в зависимости от числа процессоров p, чтобы эффективность распараллеливания E_{par} оставалась постоянной. Эта характеристика будет показывать масштабируемость конкретной вычислительной системы.
- Чем меньше необходимая степень роста W для поддержания нужного уровня эффективности распараллеливания, тем более масштабируемой является система.

Основные показатели эффективности и масштабируемости параллельных программ

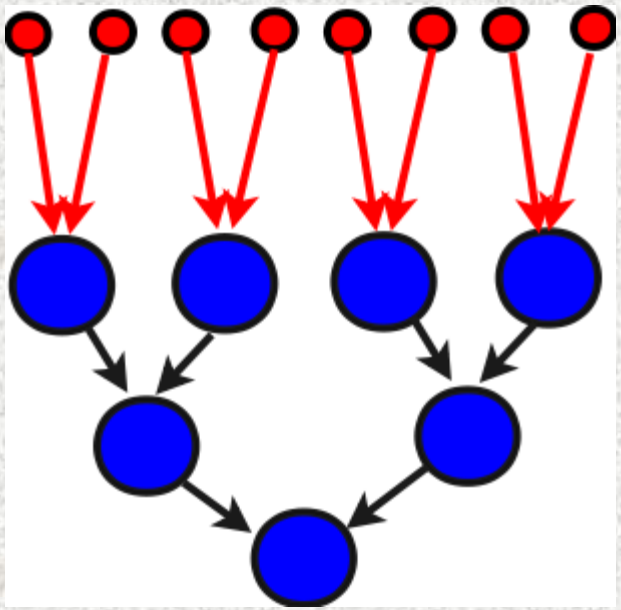
Функция изоэффективности **(***isoefficiency function***)**:

- $E_{par} = 1/(1+T_0/T_1) = 1/(1+T_0/W)$.
- $W = E_{par}/(1-E_{par})*T_0 = K*T_0$, где
  $K = E_{par}/(1-E_{par})$
- Получившаяся зависимость размера задачи, необходимой для достижения заданной эффективности распараллеливания, от числа процессоров – *функция изоэффективности (isoefficiency function)*.

Основные показатели эффективности и масштабируемости параллельных программ

Пример:

- Суммирование методом сдваивания (каскадная схема).



Основные показатели эффективности и масштабируемости параллельных программ

Пример:

- Пусть для сложения n чисел используется p процессоров
- $W = T_1 \approx n$
- $T_p \approx n/p + \mathbf{2}\log_2 p$
- $S = T_1/T_p = n/(n/p+2\log_2 p) = p/(1+2p\log_2 p/n)$
- $E_{par} = S/p = 1/(1+2p\log_2 p/n)$

Основные показатели эффективности и масштабируемости параллельных программ

Пример:

- $C = pT_p = p(n/p+2\log_2 p) = n+2p\log_2 p$
- $T_0 = pT_p - T_1 = 2p\log_2 p$
- $W = KT_0 = 2Kp\log_2 p = \Theta(p\log_2 p)$
- При увеличении числа процессоров от p до p' для поддержания постоянной эффективности распараллеливания E необходимо увеличить размер задачи в $(p'\log_2 p')/(p\log_2 p)$ раз.

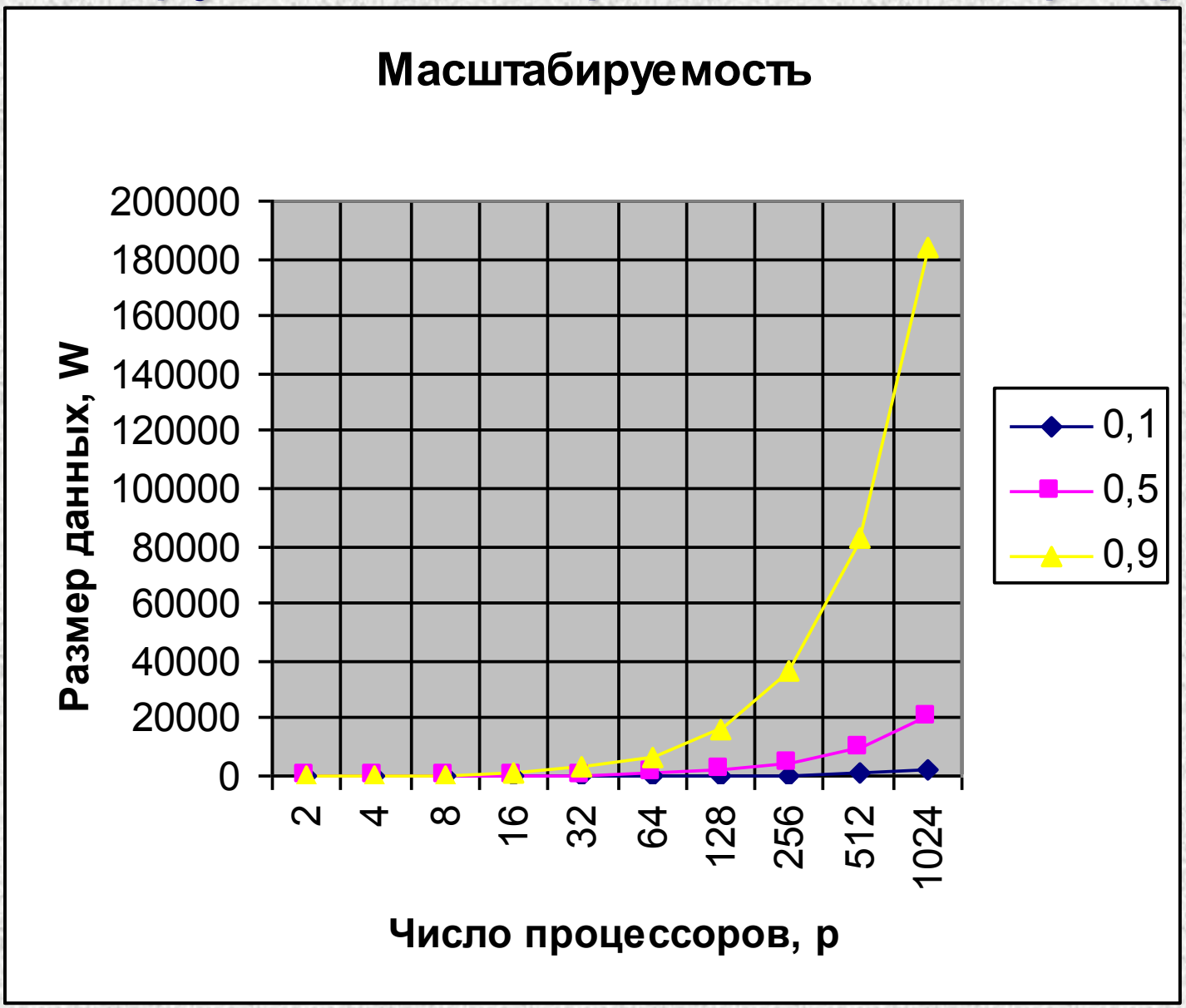
Основные показатели эффективности и масштабируемости параллельных программ

Пример:

Пусть $E_{par} = 0.5$, тогда $K = E_{par}/(1-E_{par}) = 1$.

- Пусть $p = 16$, тогда $W = 2Kp\log_2 p = 128$.
- Пусть $p = 64$, тогда $W = 2Kp\log_2 p = 768$.
- Пусть $p = 1024$, тогда $W = 2Kp\log_2 p = 20480$.

Основные показатели эффективности и масштабируемости параллельных программ



Основные показатели эффективности и масштабируемости параллельных программ

Основные помехи масштабируемости параллельных программ:

- Закон Амдала (последовательные части программы).
- Накладные расходы на коммуникации (латентность, пропускная способность).
- Неравномерность загрузки (load balancing) процессоров.
- Предел декомпозиции данных.

Методы оценки производительности

Методы оценки производительности

Какой компьютер выбрать?

В идеале было бы компьютеру однозначно сопоставить некое число.

***Пиковая производительность**: вычисляется просто и однозначно, но нет связи с реальной задачей пользователя. Даёт нижнюю оценку времени выполнения программы.*

Полезнее для пользователя оценка эффективности программно-аппаратной среды на некоторых задачах или наборе задач.

***Синтетические (или искусственные) тесты** не имеют отношения к реальным приложениям; предназначены для создания стрессовой нагрузки на отдельные подсистемы компьютера.*

***Реальные тесты** выполняют реальные задачи над реальными данными.*

Методы оценки производительности

Основные требования к тестам производительности:

- *Непротиворечивость и понятность результатов.*
- *Легкость в использовании.*
- *Масштабируемость.*
- *Переносимость.*
- *Репрезентативность.*
- *Доступность теста и его исходного кода.*
- *Воспроизводимость.*

Методы оценки производительности

Наиболее известные тесты (бенчмарки):

- *Linpack*
- *STREAM*
- *Ливерморские циклы*
- *Perfect Club Benchmarks*
- *SPEC*
- *HINT*
- *NAS Parallel Benchmarks (NPB)*
- *HPC Challenge*
- *Graph500*
- *HPCG*

Методы оценки производительности

Тест Linpack

- *Создан Джеком Донгаррой (Jack Dongarra) и его коллегами в 1979 году.*
- *Используется для формирования списков Top500 и Топ50.*
- *Решение больших систем линейных алгебраических уравнений с плотной квадратной матрицей методом LU-разложения.*
- *Число операций с плавающей точкой оценивается по формуле $2n^3/3 + 2n^2$, где n – линейный размер матрицы.*
- *Сначала Linpack 100×100, запрет изменений текста.*
- *Далее - Linpack 1000×1000, стандартная головная часть программы.*
- *Сейчас – произвольный (максимально возможный) размер матрицы, возможность вносить любые изменения в текст.*

Методы оценки производительности

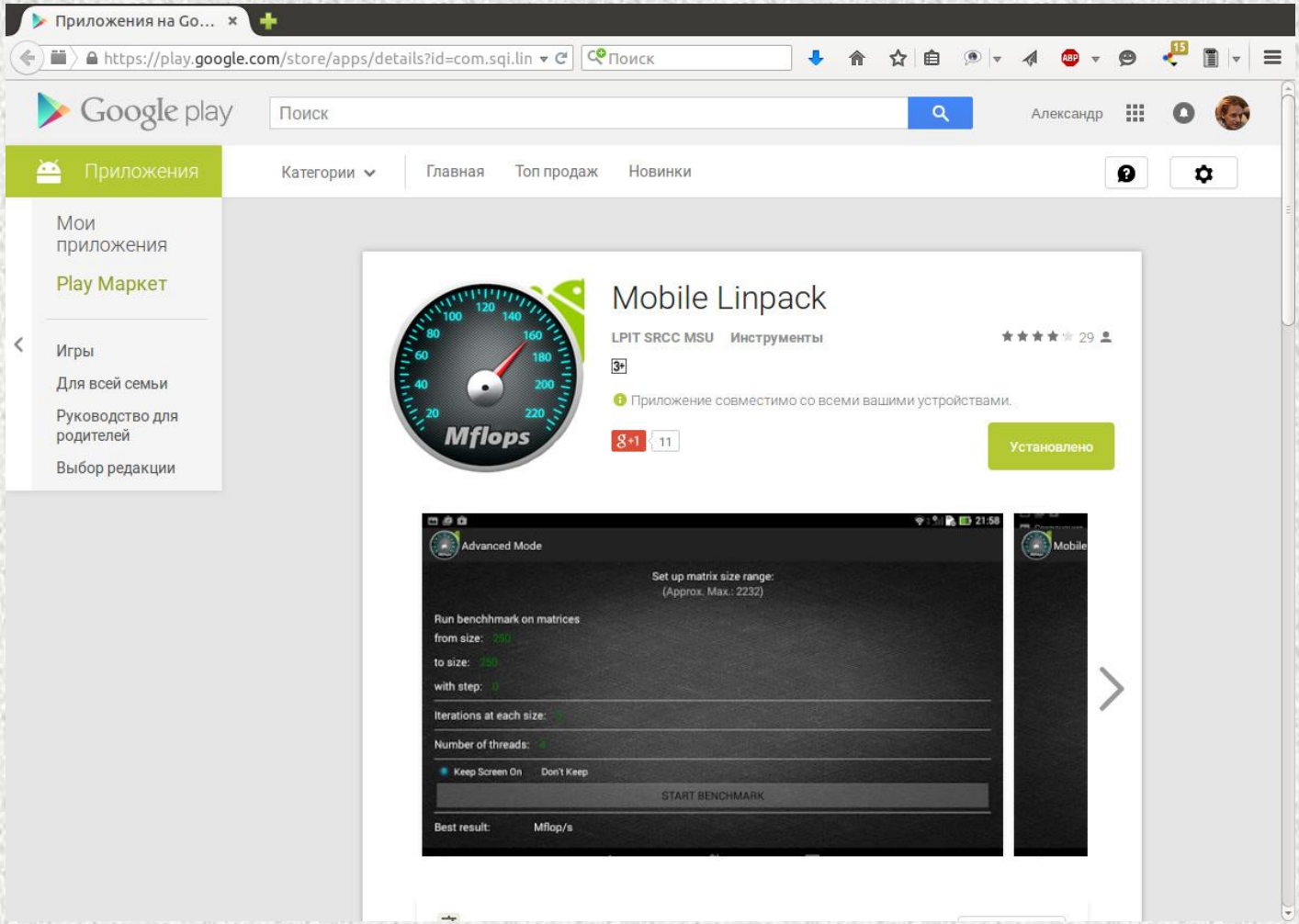
High Performance Linpack (HPL)

- http://www.netlib.org/benchmark/hpl/
- *Наиболее популярная реализация теста Linpack на языке Си.*
- *Обмены между процессорами выполняются через процедуры MPI.*
- *Вычисления на каждом процессоре основываются на низкоуровневой библиотеке базовых функций линейной алгебры BLAS (Basic Linear Algebra Subprograms) или VSIPL(Vector Signal Image Processing Library).*
- *Варианты BLAS:*
  - *ACML (AMD Core Math Library) для процессоров AMD Athlon и Opteron.*
  - *Intel MKL (Intel Math Kernel Library) для процессоров Intel.*
  - *cuBLAS в составе NVIDIA CUDA SDK для видеокарт и ускорителей.*
  - *ESSL (Engineering and Scientific Subroutine Library) для процессоров PowerPC.*
  - *ATLAS (Automatically Tuned Linear Algebra Software), реализация интерфейса BLAS с открытым исходным кодом.*
  - *uBLAS, часть библиотеки Boost.*

Методы оценки производительности

Mobile Linpack

- http://linpack.hpc.msu.ru/
- *Реализация теста Linpack для мобильных устройств. Реализованы версии для Android 1.6 и выше и для iOS 7.1 и выше.*



Контактный адрес:
ml@parallel.ru

Методы оценки производительности

STREAM (Sustainable Memory Bandwidth in High Performance Computers)

http://www.cs.virginia.edu/stream/

- *Синтетический тест, оценивающий скорость работы с памятью с простой арифметикой и без.*
- *STREAM. Производительность на операциях:*
  - *a(i)=b(i);*
  - *a(i)=q*b(i);*
  - *a(i)=b(i)+c(i);*
  - *a(i)=b(i)+q*c(i);*
  - *замер скорости передачи данных.*
- *STREAM2. Производительность на операциях:*
  - *a(i)=q;*
  - *a(i)=b(i);*
  - *a(i)=a(i)+q*b(i);*
  - *sum = sum + a(i);*

Методы оценки производительности

STREAM (Sustainable Memory Bandwidth in High Performance Computers)

• Размер массивов задаётся достаточным, чтобы выйти за размеры кэш-памяти. Отношение пиковой производительности к скорости передачи данных почти всегда больше 1.Чем отношение больше, тем больше несбалансированность компьютера.
• Параллельные версии с использованием OpenMP и MPI.

Простота и переносимость тестов вызывает недоверие, появляются более сложные тесты, что вызывает проблемы с переносимостью и т.д. Поэтому создаются наборы тестов.

Методы оценки производительности

NAS Parallel Benchmarks

- *http://www.nas.nasa.gov/Software/NPB/*
- *Набор тестов производительности, разработанных в NASA Advanced Supercomputing (NAS) Division (ранее NASA Numerical Aerodynamic Simulation Program).*
- *Существуют последовательная реализация, параллельные реализации с использованием MPI, OpenMP, MPI+OpenMP , вариант на JAVA, версия для Grid на базе Globus.*
- *Последняя на данный момент версия NPB 3.3.*
- *Размер задачи – классы:*
  - *S, W (для тестовых прогонов);*
  - *A, B, C (каждый в среднем в 4 раза больше предыдущего);*
  - *D, E, F (каждый в среднем в 16 раз больше предыдущего)*

Методы оценки производительности

NAS Parallel Benchmarks

- *5 вычислительных ядер:*
  - *IS (Integer Sort);*
  - *EP (Embarrassingly Parallel);*
  - *CG (Conjugate Gradient);*
  - *MG (MultiGrid);*
  - *FT (Fast Fourier Transform).*
- *3 модельных приложения:*
  - *BT (Block Tri-diagonal solver);*
  - *SP (Scalar Penta-diagonal solver);*
  - *LU (Lower-Upper Gauss-Seidel solver).*
- *3 дополнительных теста:*
  - *UA (Unstructured Adaptive mesh);*
  - *DC (Data Cube);*
  - *DT (Data Traffic).*

Методы оценки производительности

HPC Challenge

- *http://www.hpcchallenge.org/*
- *Включает в себя 7 тестов:*
  - *HPL (Linpack);*
  - *DGEMM (вычисляет производительность перемножения матриц);*
  - *STREAM;*
  - *PTRANS (транспонирование матрицы);*
  - *RandomAccess (вычисляет скорость случайных обращений к памяти);*
  - *FFTE (реализация одномерного дискретного преобразования Фурье);*
  - *Communication bandwidth and latency (скорость передачи данных и латентность).*

Методы оценки производительности

Graph500

- *http://www.graph500.org/*
- *Анонсирован в 2010 году, составляется свой список наиболее производительных компьютеров.*
- *На данный момент последняя версия теста 2.1.4.*
- *Включает последовательный вариант, варианты на OpenMP, MPI и вариант для Cray XMT.*
- *Реализует поиск в ширину в большом ненаправленном графе (модель графа Кронекера со средним весом вершин 16).*
- *Используется для измерения пропускной способности сетевой системы суперкомпьютеров .*
- *Производительность алгоритма BFS измеряется количеством пройденных дуг графа в секунду (Traversed Edges Per Second, TEPS). Используются обозначения ME/s и GE/s – миллионы и миллиарды пройденных дуг в секунду соответственно.*

Методы оценки производительности

HPCG

- *https://software.sandia.gov/hpcg/*
- *High Performance Conjugate Gradient – решение системы линейных алгебраических уравнений с разрежённой квадратной положительно определённой симметричной матрицей.*
- *Оценивает не только скорость самих вычислений, но и нерегулярных обращений в память.*
- *Разработчики: Jack Dongarra, Michael Heroux, Piotr Luszczek*
- *Анонсирован в 2013 году*
- *На данный момент последняя версия теста 2.4.*

Методы оценки производительности

Необходимость комплексного тестирования программно-аппаратной среды

- *базовый уровень ПО (ОС, компилятор, системы программирования);*
- *базовый уровень аппаратуры (элементарные операции, иерархия памяти);*
- *уровень операций ввода/вывода;*
- *базовый коммуникационный уровень;*
- *коммуникационный уровень приложений;*
- *уровень модельных приложений;*
- *уровень реальных приложений.*

Открытая энциклопедия свойств алгоритмов AlgoWiki

Алговики

algowiki-project.org/ru/Открытая_энциклопедия_свойств_ал

Поиск

Algowiki

Статья | Обсуждение | Читать | Править | История | Ещё | Поиск

Открытая энциклопедия свойств алгоритмов

- Заглавная страница
- Форум
- Свежие правки

Хранилище файлов
- Новые файлы
- Загрузить файл

Инструменты
- Ссылки сюда
- Связанные правки
- Загрузить файл
- Спецстраницы
- Версия для печати
- Постоянная ссылка
- Сведения о странице

На других языках
- English

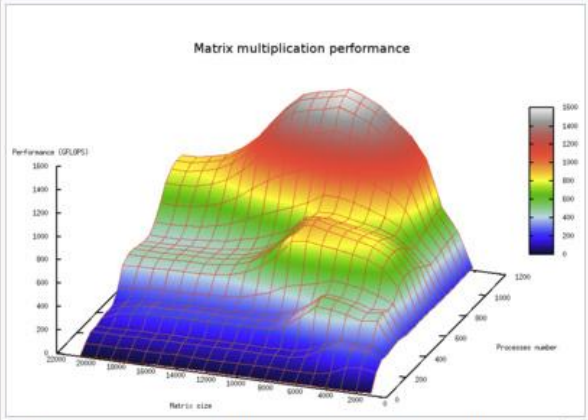
Добро пожаловать!

AlgoWiki - это открытая энциклопедия по **свойствам алгоритмов и особенностям их реализации** на различных программно-аппаратных платформах от мобильных платформ до экзафлопсных суперкомпьютерных систем с возможностью коллективной работы всего мирового вычислительного сообщества.

Цель **AlgoWiki** - дать исчерпывающее описание алгоритма, которое поможет оценить его потенциал применительно к конкретной параллельной вычислительной платформе. Кроме классических свойств алгоритмов, например, последовательной сложности, в AlgoWiki представлены дополнительные сведения, составляющие в совокупности полную картину об алгоритме: параллельная сложность, параллельная структура, детерминированность, оценки локальности данных, эффективность и масштабируемость, коммуникационный профиль конкретных реализаций и многие другие.

Читать подробнее: О проекте

Изображение дня



Производительность умножения плотных матриц

Организация работы

- Структура описания свойств алгоритмов
- Руководства по заполнению разделов описания
- Глоссарий
- Помощь в редактировании

Структура проекта

Классификация алгоритмов - основной раздел AlgoWiki, содержащий описания всех алгоритмов. Алгоритмы добавляются в подходящий раздел классификации, при необходимости классификация расширяется за счет но

http://algowiki-project.org

Московский государственный университет имени М.В.Ломоносова
Суперкомпьютерный консорциум университетов России

Эффективность параллельных программ

А.С.Антонов
Вед. н.с. НИВЦ МГУ, к.ф.-м.н.
asa@parallel.ru

Летняя суперкомпьютерная академия
Москва, 2015