Суперкомпьютерные системы и особенности строения алгоритмов

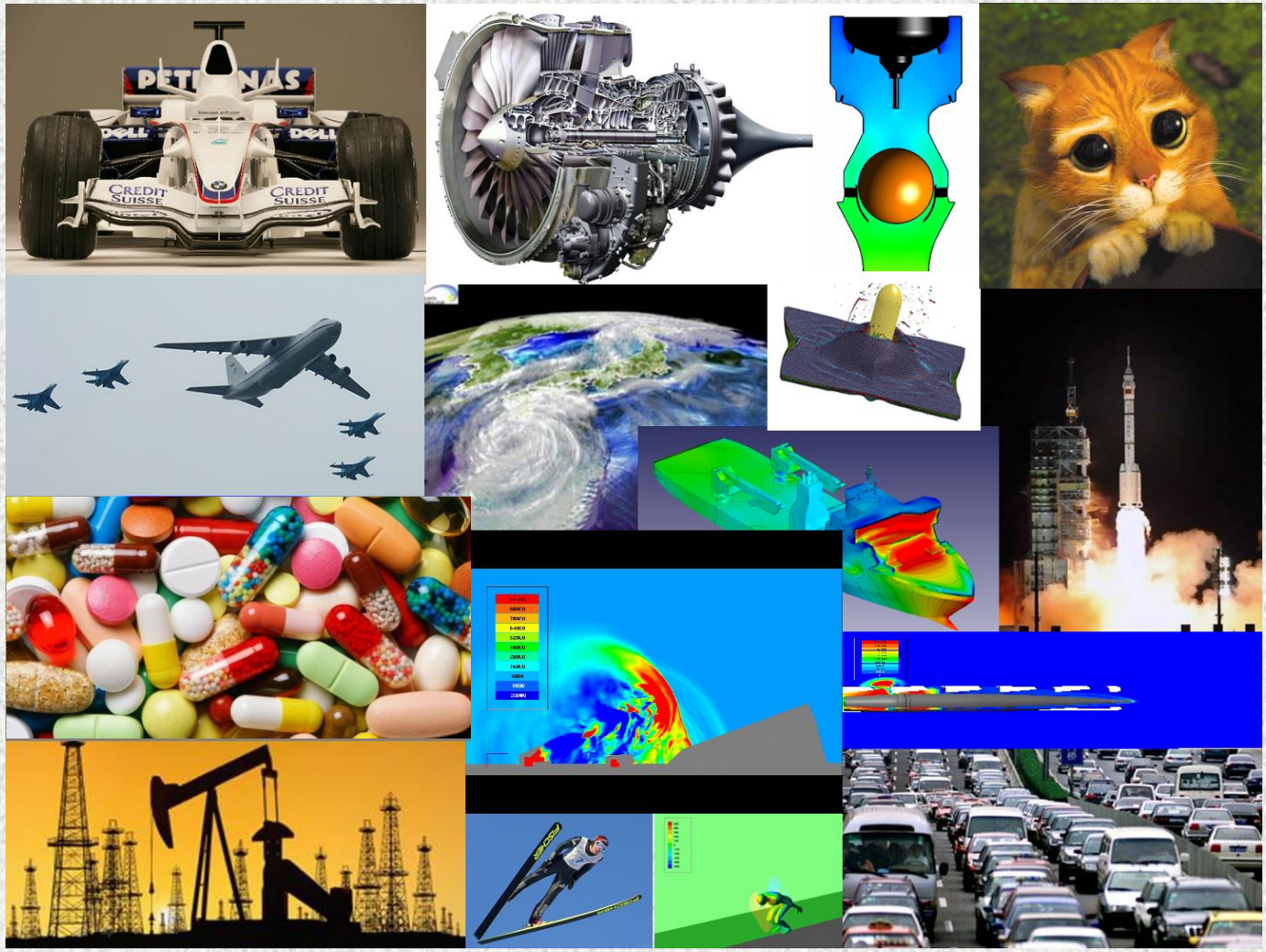
Вл.В.Воеводин

Директор НИВЦ МГУ,
Директор Филиала МГУ в г.Сарове

voevodin@parallel.ru

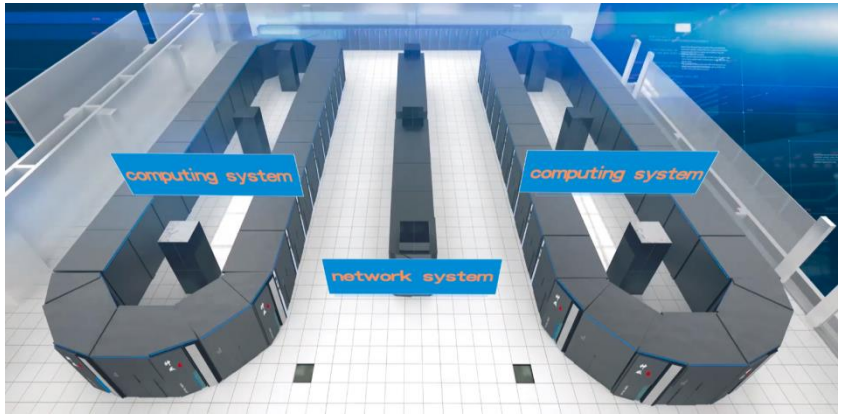
14 ноября 2022, ОИЯИ, Дубна

Суперкомпьютерные технологии сегодня везде



Суперкомпьютерные технологии – это инструмент **обеспечения конкурентоспособности** науки, компаний, промышленности, экономики, страны.
Сделать быстрее, дешевле, качественнее, быстрее вывести на рынок…

Суперкомпьютер Sunway TaihuLight, Китай
(#1 Top500 в 2016-2017 г.)



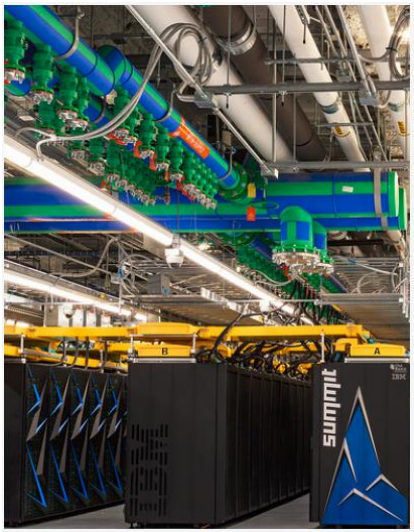
40 960 вычислительных узлов
40 960 CPUs (SW26010, 260 ядер)

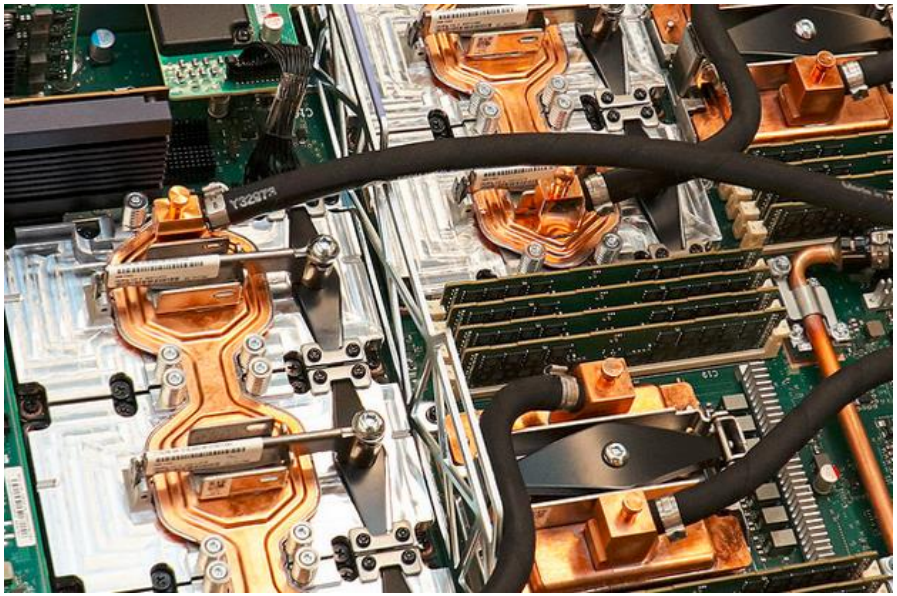
Всего: 10 649 600 процессорных ядер

Производительность:
Пик (теория): 125.4 Pflop/s
Тест Linpack: 93 Pflop/s (74%)

Оперативная память = 1.31 Пбайт
HDD = 20 ПБайт

Суперкомпьютер IBM Summit, США
(#1 Top500 в 2018-2019 г.)





4 608 вычислительных узлов,
в каждом узле:
2 x CPUs (IBM Power9, 22 ядра)
6 x GPU (NVIDIA Tesla V100)

Производительность:
Пик (теория): 200.8 Pflop/s
Тест Linpack: 148.6 Pflop/s (74%)

Оперативная память = 10 Пбайт
HDD = 250 ПБайт

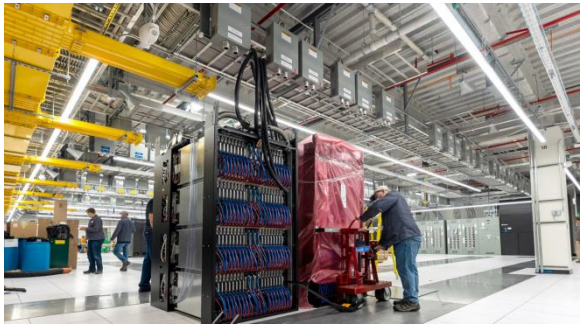
Суперкомпьютер Fujitsu Fugaku, Япония
(#1 Top500 в 2020-2021 г.)

158 976 выч.узлов, 432 стойки
7 630 848 ядер

CPU: A64FX ARM v8.2-A, 48/52 cores, 2.2 GHz
Векторные инструкции (SVE), аппаратная поддержка барьеров…
Int: 1,2,4,8 байт; Float: 16,32,64 бит

Производительность:
Пик (теория): 537 Pflop/s
Тест Linpack: 442 Pflop/s (81%)
Оперативная память = 5.09 Пбайт
Энергопотребление = 29.9 МВт
Площадь = 1920 м^2

Суперкомпьютер Frontier, США
(#1 Top500 в 2022 г.)



#1 – Top500
#1 – Green500
#1 – HPL-AI

9 408 вычислительных узлов,
в каждом узле:
1 x CPUs (AMD “Trento”, 64 ядра, 2GHz)
4 x GPU (AMD Radeon MI250X)
8 730 112 ядер

Производительность:
Пик (теория): 1.68 Eflop/s
Тест Linpack: 1.1 Eflop/s (65%)

Оперативная память = 9.2 PBytes
HDD = 716 PB (+37 PB на узлах)

21 MW (всего 29 MW) – 52.2 Gflops/Watt
(1 стойка – 62.68 Gflops/Watt)

Так ли важно, кому доступен этот “Экзафлопс”?

РИА НОВОСТИ

ВАШИНГТОН, 7 окт – РИА Новости. США вводят ограничения на поставки 28 китайским компаниям продукции для выпуска суперкомпьютеров и полупроводников, сообщило американское министерство торговли.

HPC wire

Since 1987 - Covering the Fastest Computers in the World and the People Who Run Them

- Home
- Topics
- Sectors
- Exascale
- Specials
- Resource Library
- Podcast
- Events
- Solution Channels
- Job Bank
- About
- Subscribe

US Bars Nvidia and AMD from Selling Top GPUs into China

By Tiffany Trader

September 1, 2022

New trade restrictions levied by the United States against China limit the sale of cutting-edge HPC and AI technologies from Nvidia and AMD to the world’s second-largest economy.

In an SEC filing, Nvidia revealed it was prohibited from exporting its A100 and forthcoming H100 GPUs to China and Russia, effective immediately. The stated purpose of the licensing requirements is to prevent “military end use” by these nations.

AMD reported it had also received instructions from U.S. authorities to halt sales of its top GPU chip, the Instinct MI250, to China and Russia. A variant of that chip, the MI250X, powers the U.S. Department of Energy’s Frontier supercomputer, which became the first officially ranked exascale supercomputer earlier this year.

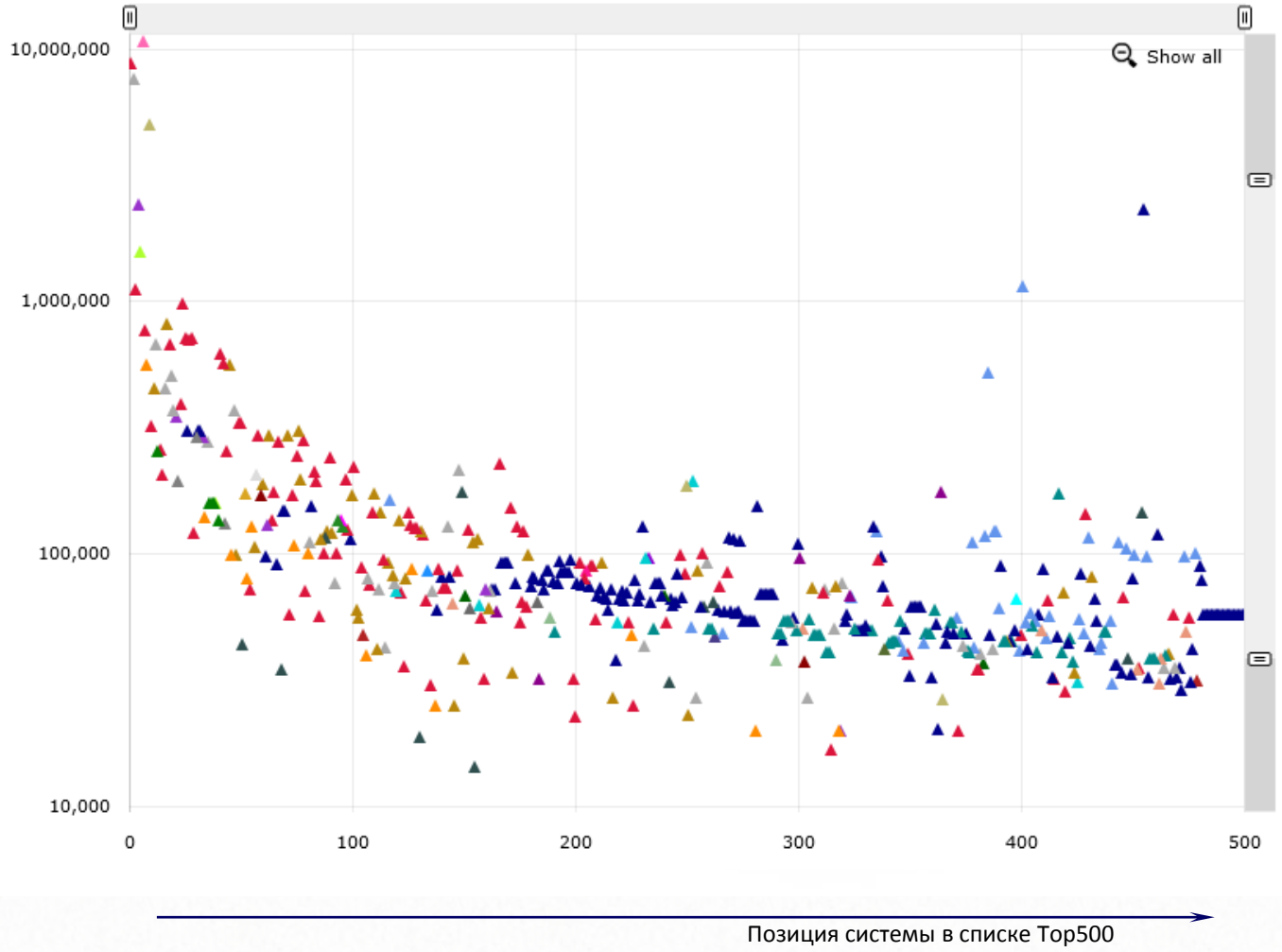
Годы, флопсы и степень параллелизма

(когда и как был достигнут очередной ‘X’flops)

10^6	Mflops	1964 г.	CDC 6600	10 MHz	1 CPUs
10^9	Gflops	1985 г.	Cray 2	125 MHz	8 CPUs
10^{12}	Tflops	1997 г.	ASCI Red	200 MHz	9152 CPUs
10^{15}	Pflops	2008 г.	Roadrunner	3,2 GHz	122400 Cores
10^{18}	Eflops	2022 г.	Frontier	2,0 GHz	8730112 Cores

Число ядер в системах списка Top500

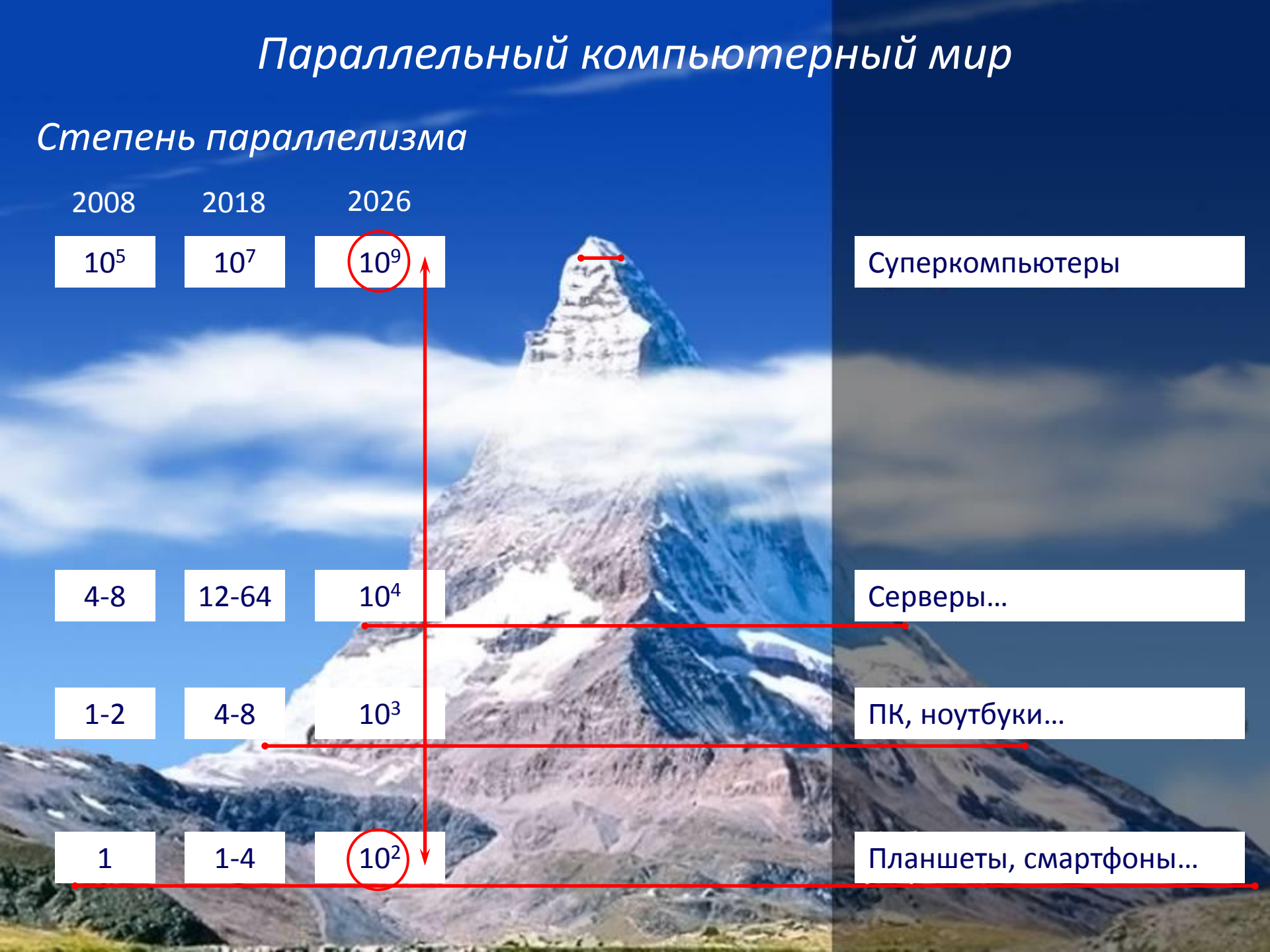
(http://top500.org, июнь, 2022 г.)



Параллельный компьютерный мир

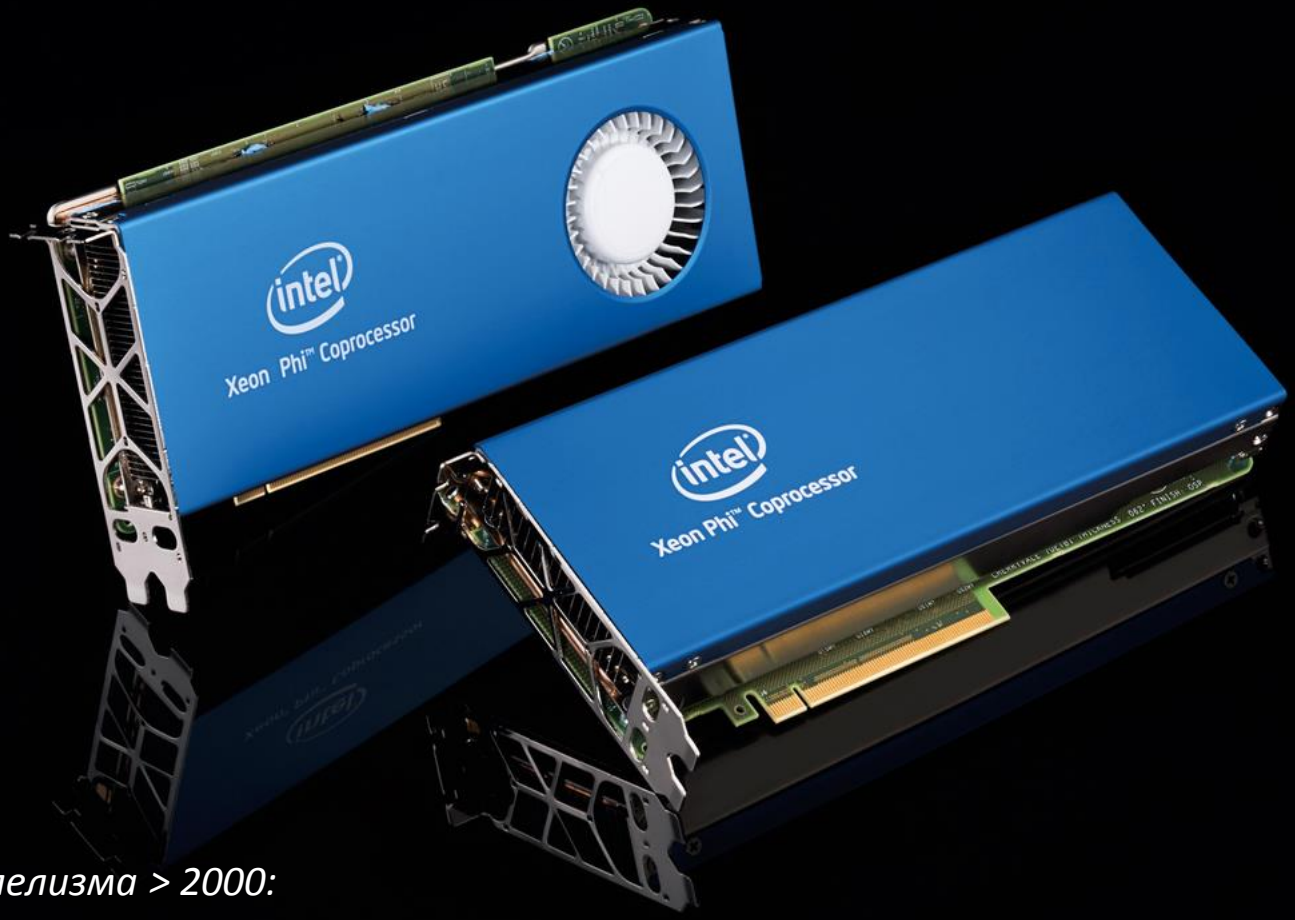
Степень параллелизма

2008	2018	2026	
10^5	10^7	10^9	Суперкомпьютеры
4-8	12-64	10^4	Серверы…
1-2	4-8	10^3	ПК, ноутбуки…
1	1-4	10^2	Планшеты, смартфоны…



Доступный параллелизм: Intel Phi

(Knights Landing)



Степень параллелизма > 2000:

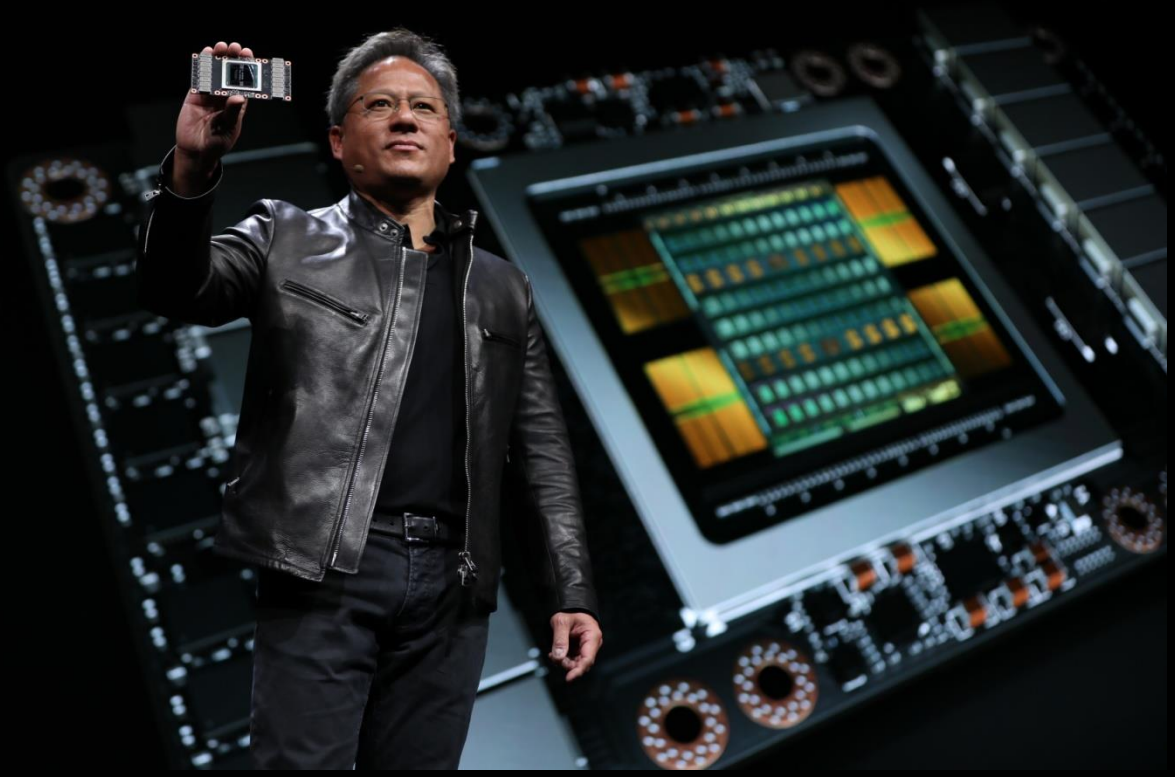
- 72 ядра,

- 4 нити на ядро,

*- AVX-512 инструкции (8 операций * 64 разряда)*

Доступный параллелизм: NVIDIA GPU

(Tesla V100, Volta)



- 5120 ядер,
- 7.5 Тфлопс на двойной точности,
- 15 Тфлопс на одинарной точности,
- 120 Тфлопс (tensor flops, FP16)

Вычислительные платформы МГУ

Технологическая основа:
Intel Xeon 4/6/10/12… ядер
SMP-узлы
Intel Xeon Phi (KNL)
NVIDIA 2070 / 2090 / K40 / P100 / V100
NVIDIA DGX-2 (V100)
IBM Power 8 / IBM Blue Gene/P
NEC SX-Aurora Tsubasa
Памяти на узел: от 12GB до 2TB

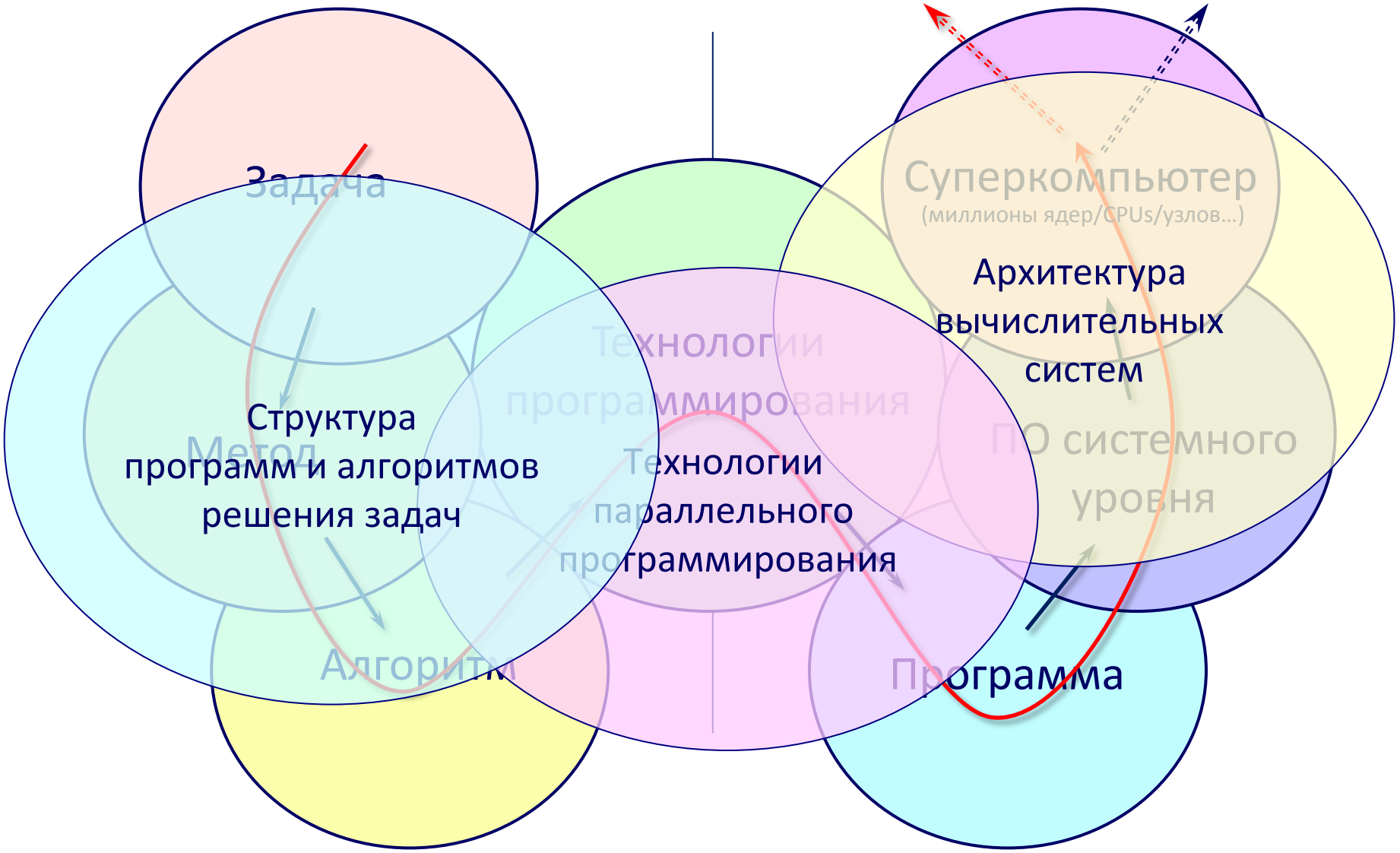
Разнообразие велико,
параллелизм – везде…

Решение задач и вычислительные системы

(Реальность)
Реальная производительность

Огромный разрыв !

(Ожидания)
Пиковая производительность

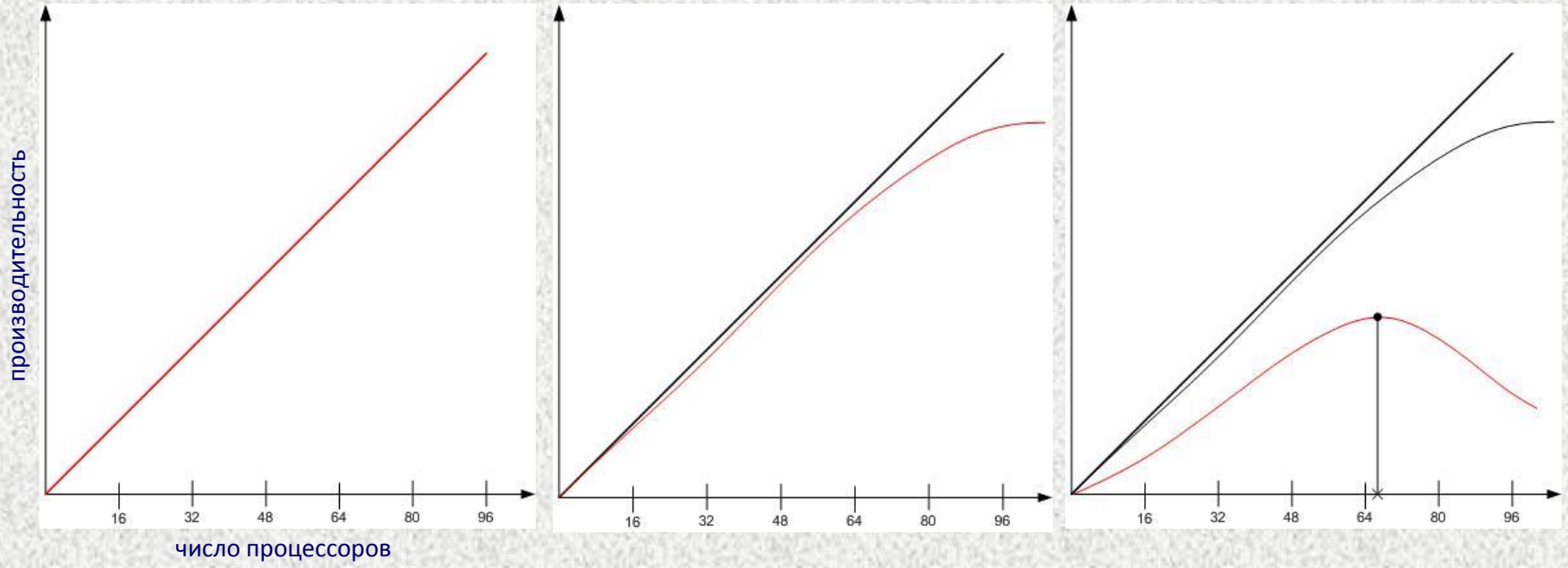


Алгоритмическая сторона

Компьютерная сторона

Масштабируемость параллельных программ и параллельных вычислительных систем

(влияют все этапы!)



Возможные причины снижения масштабируемости:

- *Многократное порождение/уничтожение нитей.*
- *Излишние барьеры для синхронизации работы нитей.*
- *Неравномерное распределение вычислительной нагрузки.*
- *Множественные конфликты по адресатам при передаче данных.*
- *Конечный параллелизм.*
- *Шум в операционной системе.*
- *Плохая организация областей вычислений и коммуникаций.*
- *Излишне мелкая гранулярность при передаче данных.*
- *Отсутствие привязки нитей к ядрам.*
- *и т.д., и т.п. …*

Тонкий анализ суперкомпьютерных приложений: динамика исполнения

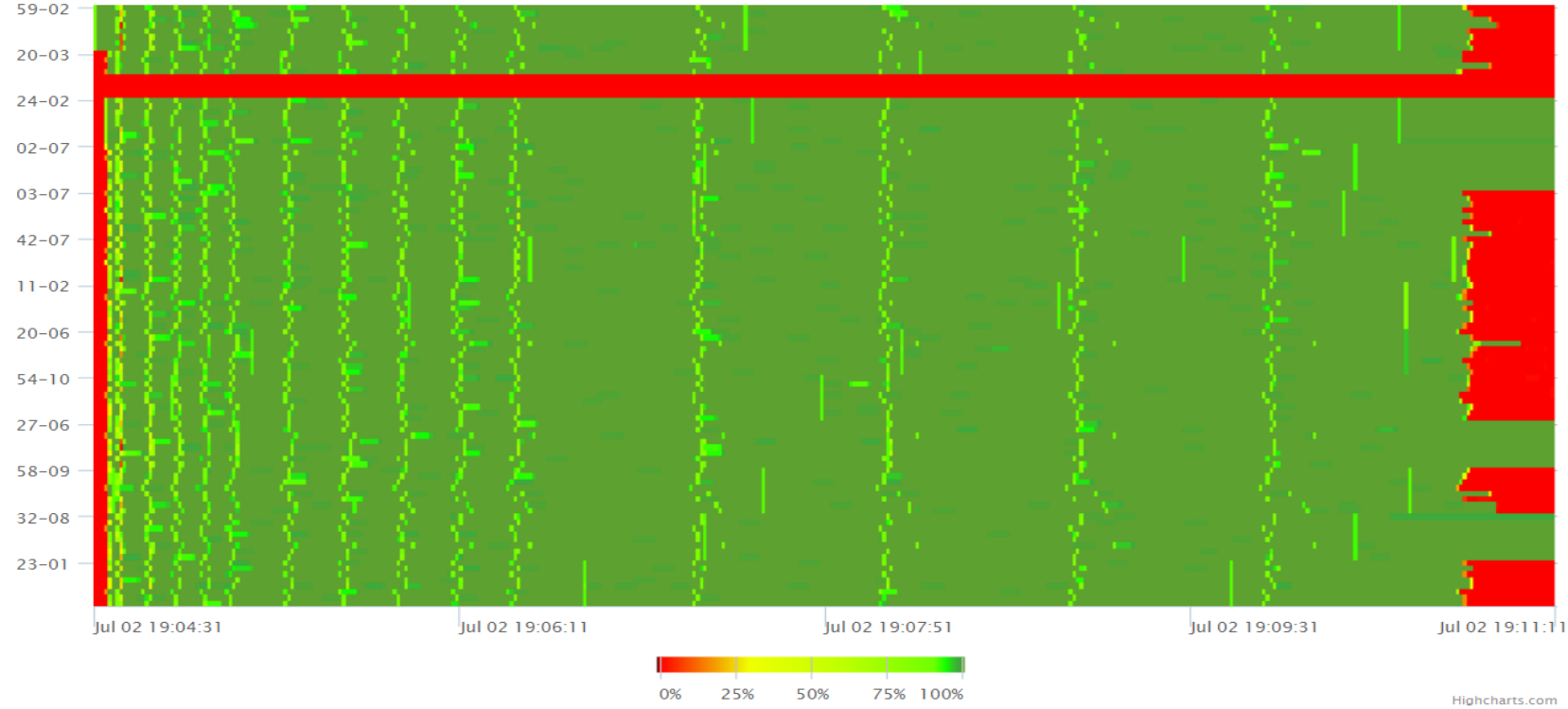
Информация о задаче № regular-1404302831-496016 пользователя

expand all

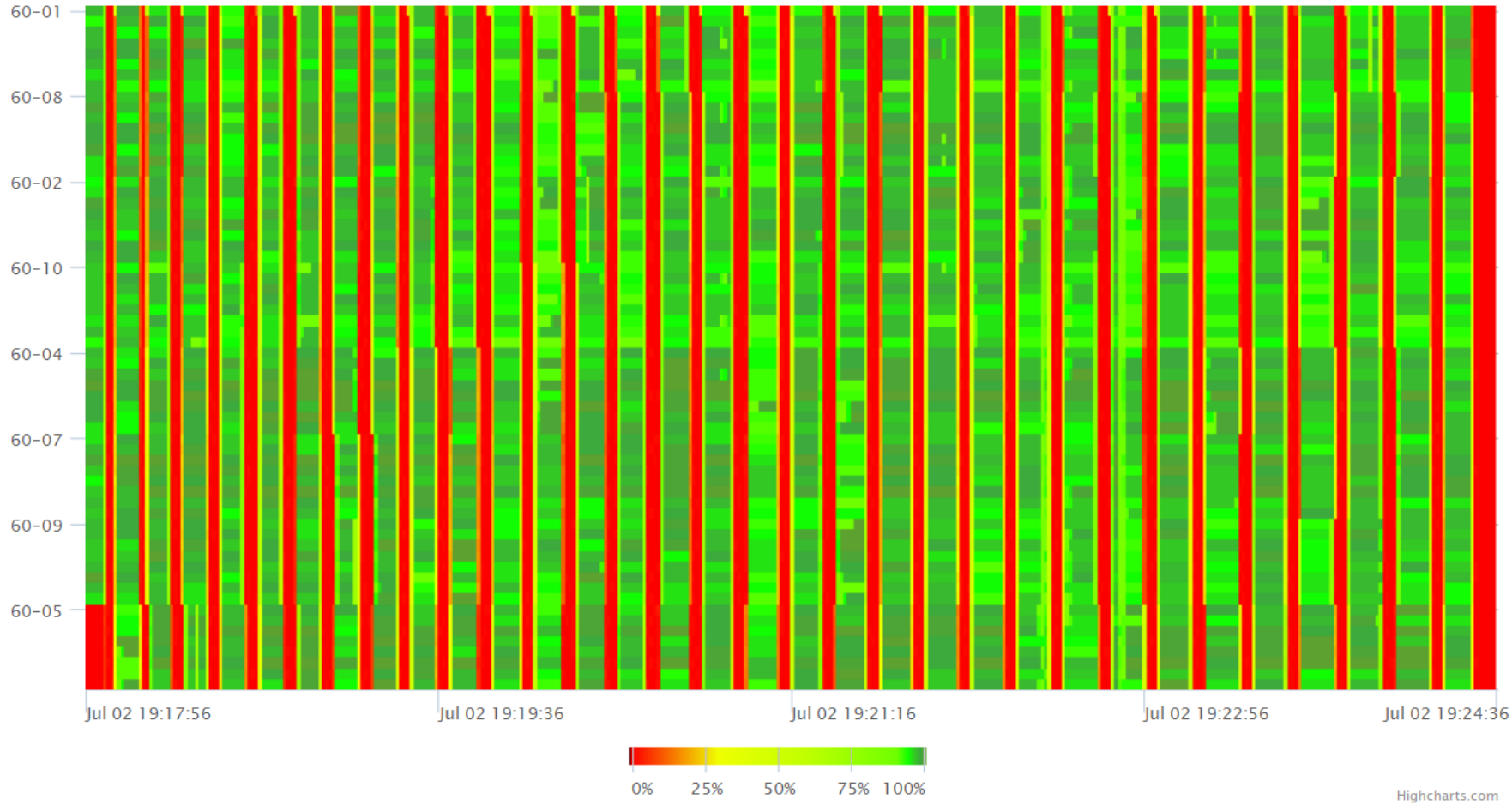
Информация о запуске

Строка запуска:	./2dxy_m01_p08.ex
Число ядер:	100
Номера узлов:	node-02-07,node-03-07,node-11-02,node-20-03,node-20-06,node-23-01,node-24-02,node-27-06,node-32-08,node-42-07,node-54-10,node-58-09,node-59-02
Дата постановки в очередь:	Wed, 02 Jul 2014 16:07:11 (1404302831)
Дата запуска:	Wed, 02 Jul 2014 19:04:29 (1404313469)
Дата окончания счета:	Wed, 02 Jul 2014 19:46:13 (1404315973)
Время счета:	0 days 0 hours 41 minutes 44 seconds
Время ожидания:	0 days 2 hours 57 minutes 18 seconds

Количе
процес



Тонкий анализ суперкомпьютерных приложений: динамика исполнения



Хорошо ли мы знаем свойства и особенности параллельных алгоритмов ?

Должны ли мы думать о параллельных алгоритмах?

Да… К сожалению, да…

Хорошо ли мы знаем статические и динамические свойства параллельных программ?

Должны ли мы думать о свойствах параллельных программ?

Да… К сожалению, да…

Хорошо ли мы знаем архитектуры параллельных компьютеров ?

Должны ли мы думать об архитектурах?

Да… К сожалению, да…

Поколения архитектур и парадигмы программирования
(или как часто мы были вынуждены полностью переписывать приложения?)

С 1976 года до наших дней:

70-е – Векторизация циклов
80-е – Распараллеливание циклов (внешних) + Векторизация (внутренних)
90-е - MPI
середина 90-х - OpenMP
середина 2000-х - MPI+OpenMP
2010-е - CUDA, OpenCL, MPI+OpenMP+ускорители (GPU, Xeon Phi)

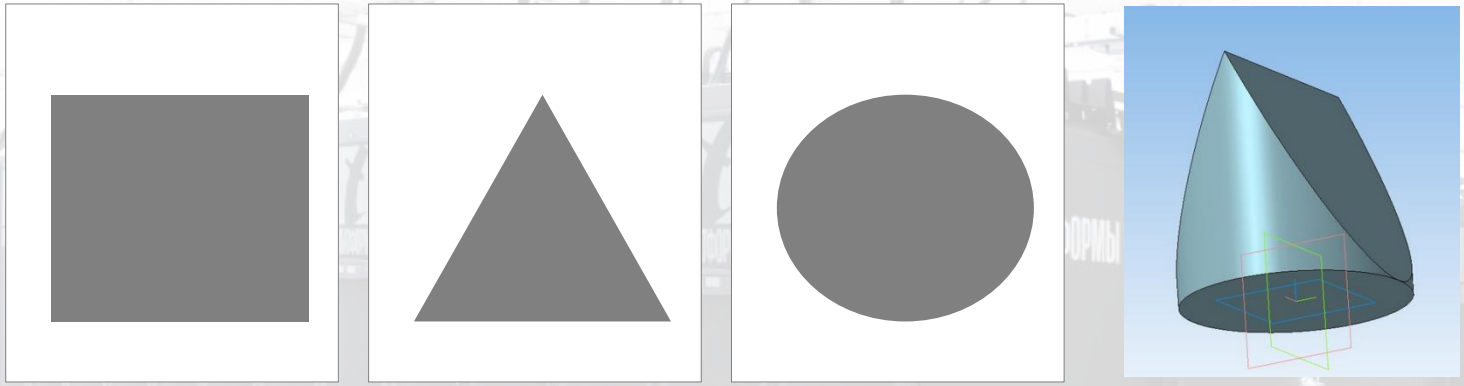
…

Изменения в архитектуре компьютеров не меняют алгоритмов! Но:

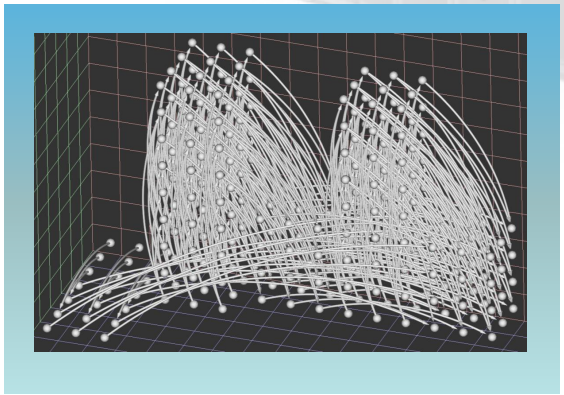
Для каждого поколения компьютеров мы вынуждены:
- Анализировать алгоритмы, чтобы понять, как их приспособить под новую компьютерную платформу ;
- Описывать найденные свойства, чтобы получить эффективную реализацию для новой платформы.

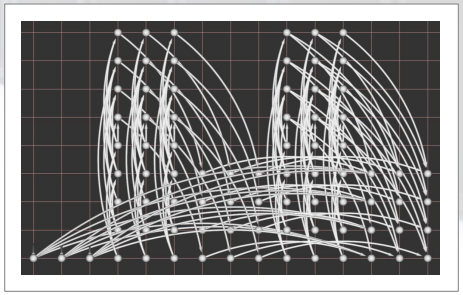
Изменения в архитектуре компьютеров не меняют алгоритмов!

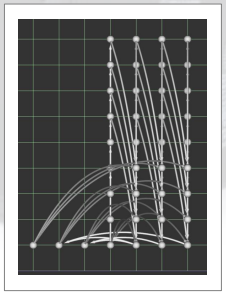
Эти фигуры разные?

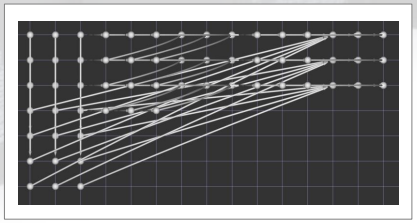


Каковы способы представления этого алгоритма?









…

Поколения архитектур и парадигмы программирования
(или как часто мы были вынуждены полностью переписывать приложения?)

С 1976 года до наших дней:

70-е – Векторизация циклов
80-е – Распараллеливание циклов (внешних) + Векторизация (внутренних)
90-е - MPI
середина 90-х - OpenMP
середина 2000-х - MPI+OpenMP
2010-е - CUDA, OpenCL, MPI+OpenMP+ускорители (GPU, Xeon Phi)

…

Можно ли выполнить такой анализ “раз и навсегда” ?

Изменения в архитектуре компьютеров не меняют алгоритмов! Но:

Для каждого поколения компьютеров мы вынуждены:
- Анализировать алгоритмы, чтобы понять, как их приспособить под новую компьютерную платформу ;
- Описывать найденные свойства, чтобы получить эффективную реализацию для новой платформы.

Что значит “выполнить анализ алгоритма”?

Что мы должны найти в алгоритмах?

“…выполнить анализ раз и навсегда…“ – как записать результаты?

Что представляет “единое” / “универсальное” описание алгоритмов?

Какие свойства алгоритмов нужно исследовать и описать чтобы получать эффективные реализации в будущем для будущих платформ?

Слишком много “простых” вопросов…

Описание алгоритмов
(на примере разложения Холецкого)

Кратко о важном

Свойства алгоритма:

- Последовательная сложность алгоритма: $O(n^3)$
- Высота ярусно-параллельной формы: $O(n)$
- Ширина ярусно-параллельной формы: $O(n^2)$
- Объём входных данных: $\frac{n(n+1)}{2}$
- Объём выходных данных: $\frac{n(n+1)}{2}$

1 Свойства и структура алгоритма

1.1 Общее описание алгоритма

Разложение Холецкого впервые предложено французским офицером и математиком Андре-Луи Холецким в конце Первой Мировой войны, незадолго до его гибели в бою в августе 1918 г. Идея этого разложения была опубликована в 1924 г. его сослуживцем. Потом оно было использовано поляком Т. Банашевичем в 1938 г. В советской математической литературе называется также методом квадратного корня [1-3]; название связано с характерными операциями, отсутствующими в родственном разложении Гаусса.

Первоначально разложение Холецкого использовалось исключительно для плотных симметричных положительно определенных матриц. В настоящее время его использование гораздо шире. Оно может быть применено также, например, к эрмитовым матрицам. Для повышения производительности вычислений часто применяется блочная версия разложения.

Для разреженных матриц разложение Холецкого также широко применяется в качестве основного этапа прямого метода решения линейных систем. В этом случае используют специальные упорядочивания для уменьшения ширины профиля исключения, а следовательно и уменьшения количества арифметических операций. Другие упорядочивания используются для выделения независимых блоков вычислений при работе на системах с параллельной организацией.

1.2 Математическое описание алгоритма

Исходные данные: положительно определённая симметрическая матрица A (элементы a_{ij}).

Вычисляемые данные: нижняя треугольная матрица L (элементы l_{ij}).

Формулы метода:

$l_{11} = \sqrt{a_{11}}$

Описание алгоритмов
(на примере разложения Холецкого)

Общее описание

For positive definite Hermitian matrices (*symmetric matrices in the real case*), we use the decomposition $A = LL^*$, where L is the lower triangular matrix, or the decomposition $A = U^*U$, where U is the upper triangular matrix. These forms of the Cholesky decomposition are equivalent in the sense of the amount of arithmetic operations and are different in the sense of data represntation. The essence of this decomposition consists in the implementation of formulas obtained uniquely for the elements of the matrix L from the above equality. The Cholesky decomposition is widely used due to the following features.

Математическое описание

Input data: a symmetric positive definite matrix A whose elements are denoted by a_{ij}).

Output data: the lower triangular matrix L whose elements are denoted by l_{ij}).

The Cholesky algorithm can be represented in the form

$$l_{11} = \sqrt{a_{11}},$$

$$l_{j1} = \frac{a_{j1}}{l_{11}}, \quad j \in [2, n],$$

$$l_{ii} = \sqrt{a_{ii} - \sum_{p=1}^{i-1} l_{ip}^2}, \quad i \in [2, n],$$

$$l_{ji} = \left(a_{ji} - \sum_{p=1}^{i-1} l_{ip} l_{jp} \right) / l_{ii}, \quad i \in [2, n-1], j \in [i+1, n].$$

Комментарии к алгоритму

The Cholesky decomposition allows one to use the so-called *accumulation mode* due to the fact that the significant part of computation involves *dot product operations*. Hence, these dot products can be accumulated in double precision for additional accuracy. In this mode, the Cholesky method has the least *equivalent perturbation*. During the process of decomposition, no growth of the matrix elements can occur, since the matrix is symmetric and positive definite. Thus, the Cholesky algorithm is unconditionally stable.

Описание алгоритмов
(на примере разложения Холецкого)

Вычислительное ядро

A computational kernel of its serial version can be composed of $\frac{n(n-1)}{2}$ dot products of the matrix rows:

$$\sum_{p=1}^{i-1} l_{ip} l_{jp}.$$

Операции чтения/записи крайне важны!

Последовательная сложность

The following number of operations should be performed to decompose a matrix of order n using a serial version of the Cholesky algorithm:

- n square roots,
- $\frac{n(n-1)}{2}$ divisiona,
- $\frac{n^3-n}{6}$ multiplications and $\frac{n^3-n}{6}$ additions (subtractions): the main amount of computational work.

Операции чтения/записи крайне важны!

Базовая схема реализации

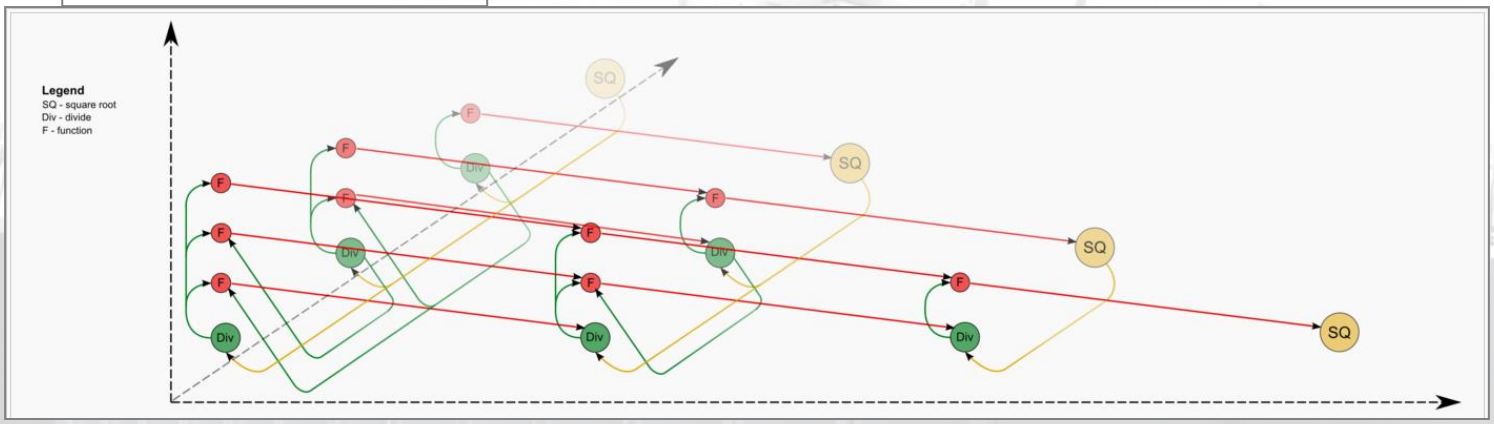
```
DO  I = 1, N
        S = A(I,I)
        DO  IP=1, I-1
                S = S - DPROD(A(I,IP), A(I,IP))
        END DO
        A(I,I) = SQRT (S)
        DO  J = I+1, N
                S = A(J,I)
                DO  IP=1, I-1
                        S = S - DPROD(A(I,IP), A(J,IP))
                END DO
                A(J,I) = S/A(I,I)
        END DO
END DO
```

Дополнительная информация

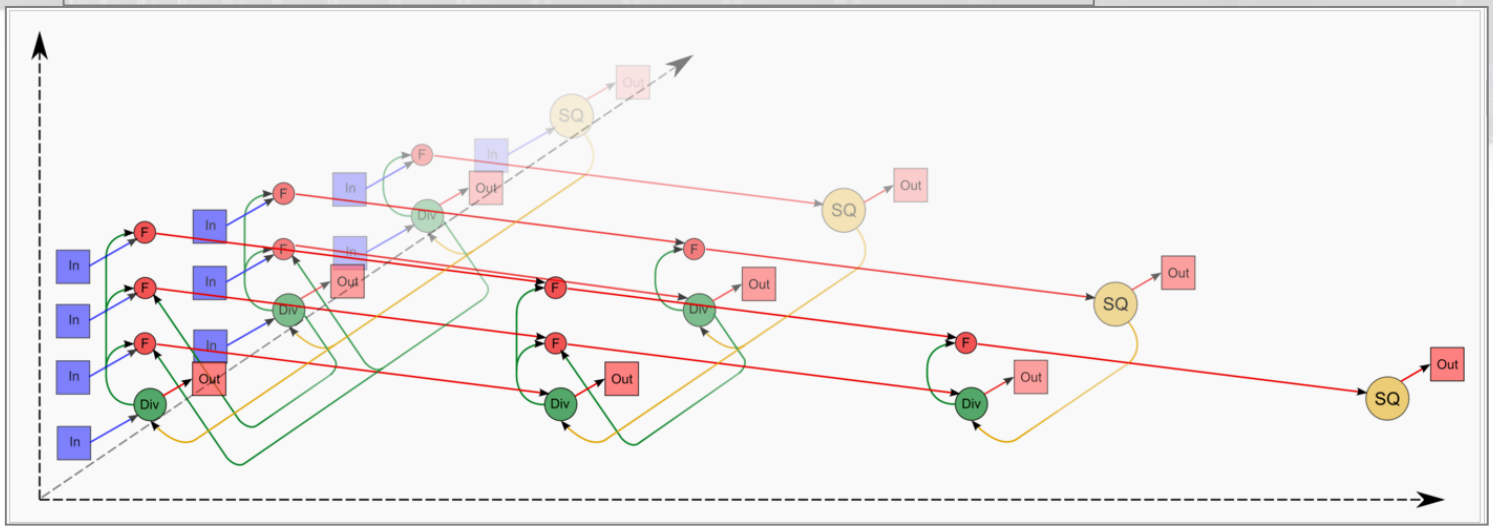
There exist block versions of this algorithm;

Описание алгоритмов
(на примере разложения Холецкого)

Информационная структура



Информационная структура с указанием входных и выходных данных



Описание алгоритмов
(на примере разложения Холецкого)

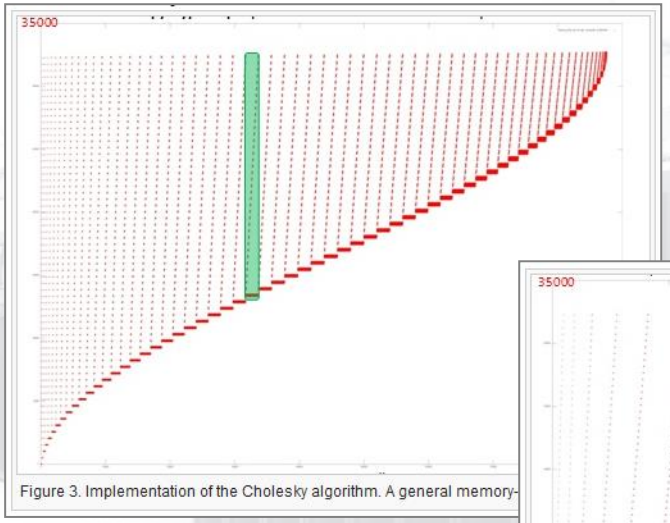


Figure 3. Implementation of the Cholesky algorithm. A general memory-

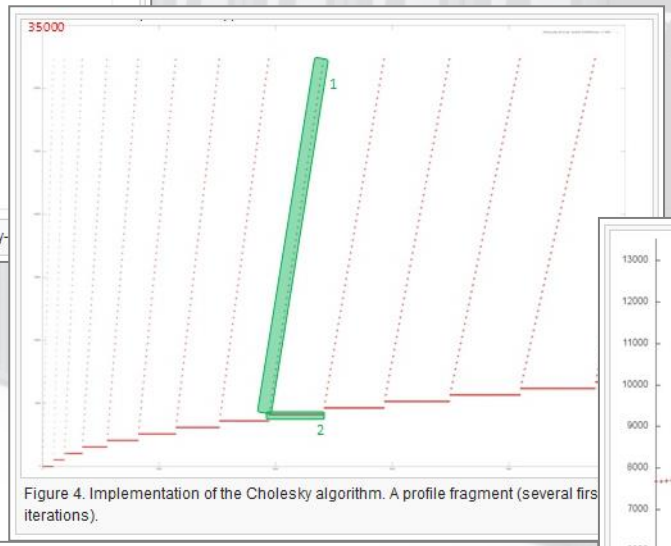


Figure 4. Implementation of the Cholesky algorithm. A profile fragment (several firs
iterations).

Локальность данных (профиль работы с памятью)

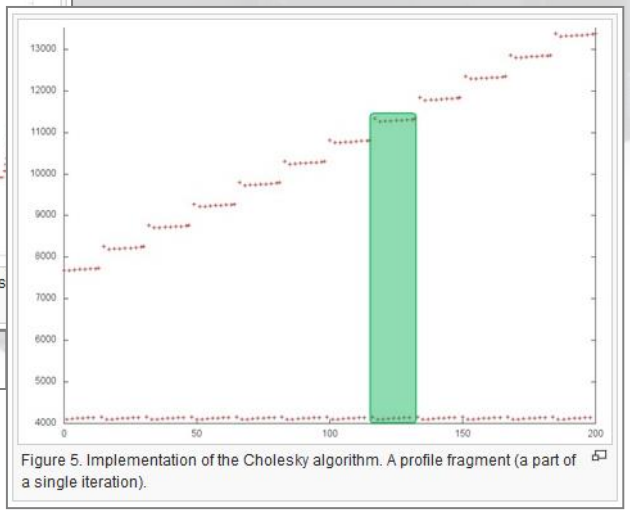
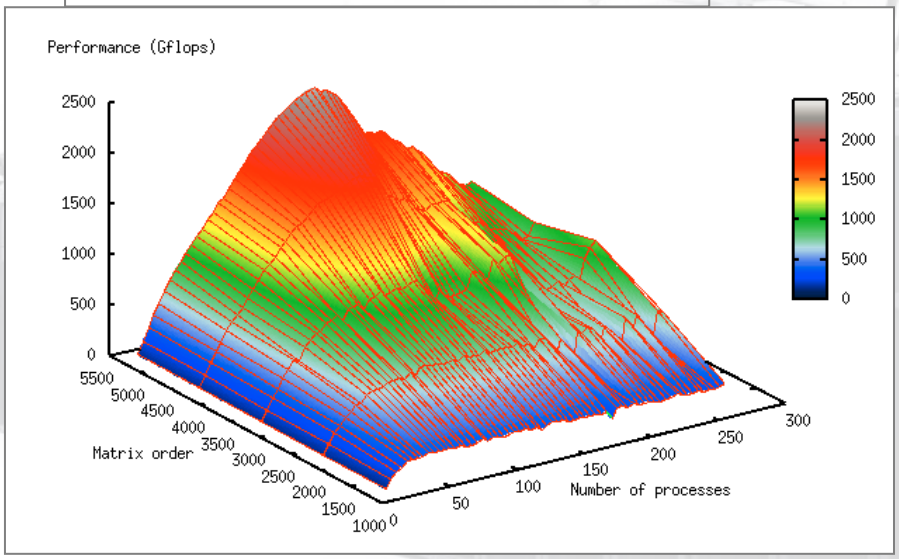


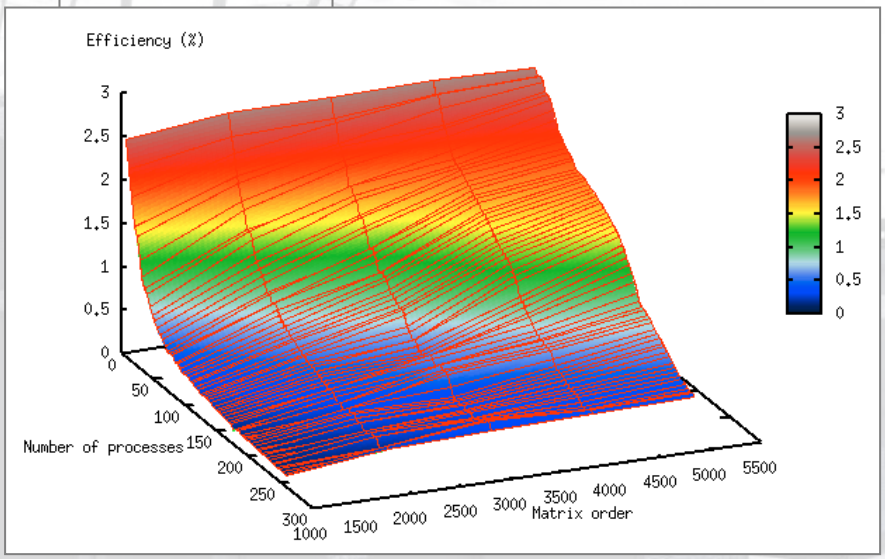
Figure 5. Implementation of the Cholesky algorithm. A profile fragment (a part of a single iteration).

Описание алгоритмов
(на примере разложения Холецкого)

Масштабируемость (производительность) *



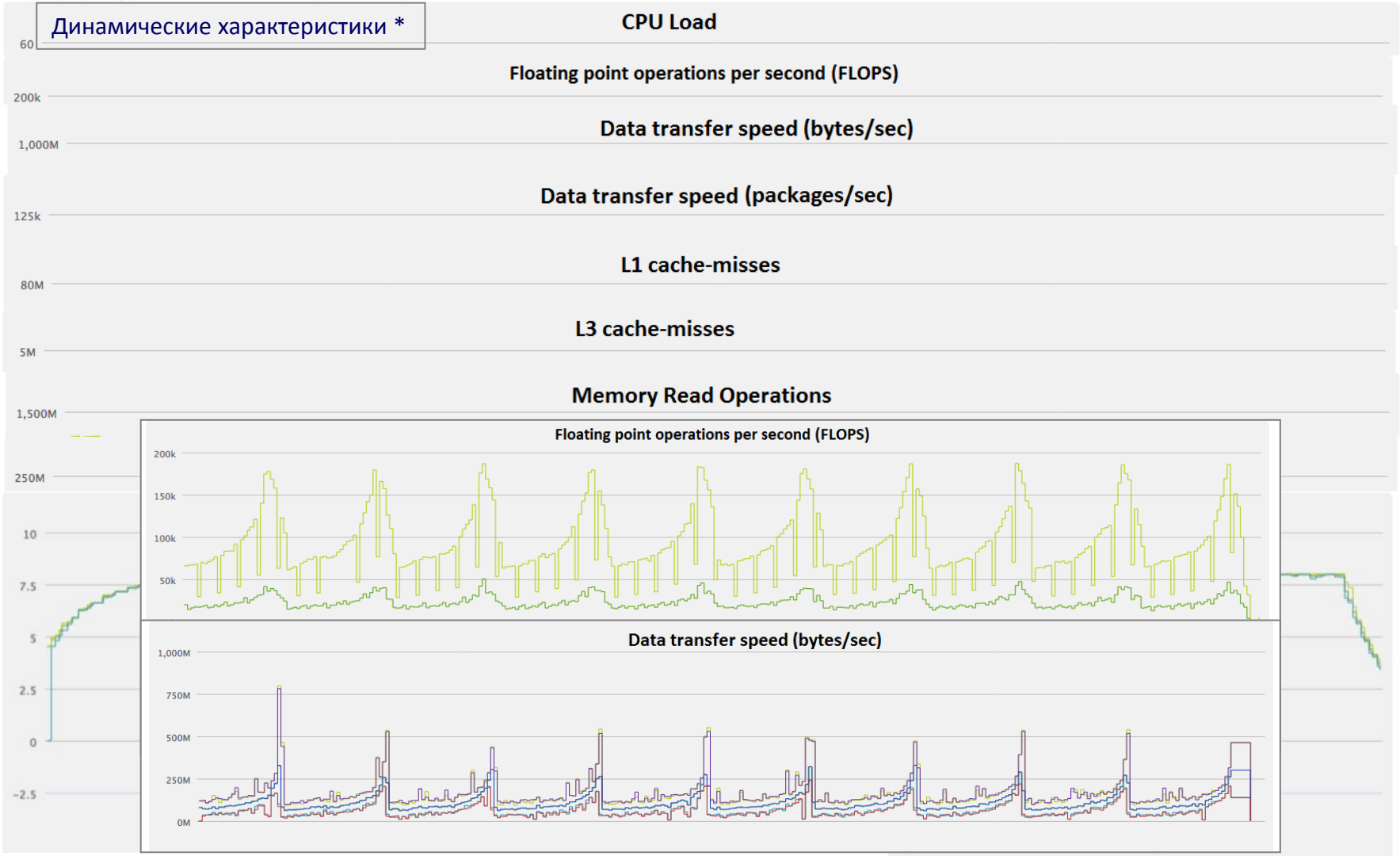
Эффективность *



* Масштабируемость, производительность, эффективность измерялись на суперкомпьютере МГУ “Ломоносов”.

Описание алгоритмов
(на примере разложения Холецкого)

Динамические характеристики *



* Динамические характеристики получены на суперкомпьютере МГУ “Ломоносов”.

Это очень полезная информация об алгоритме, она действительно нужна.

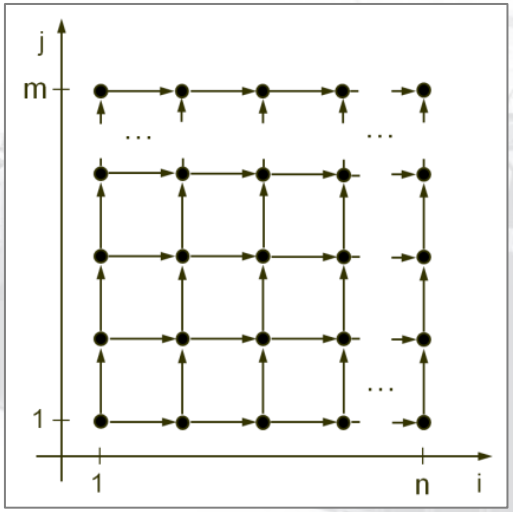
Но…

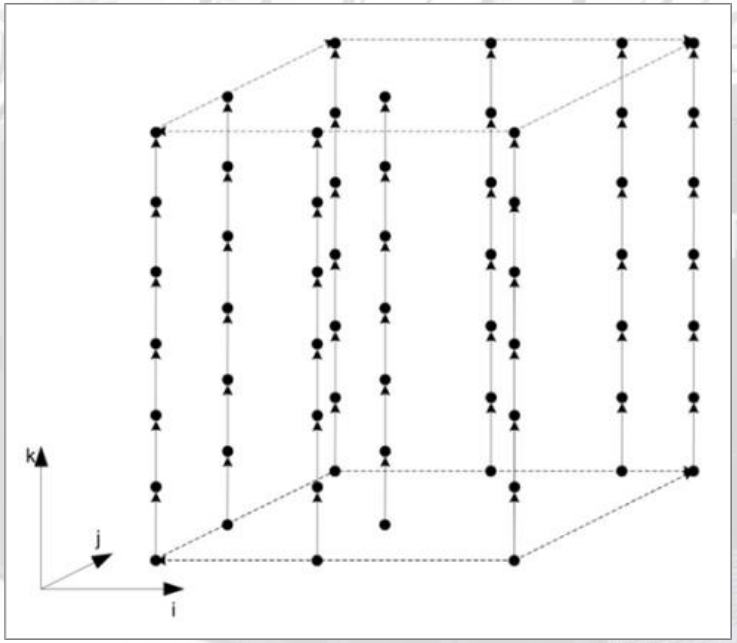
В процессе создания полного описания алгоритма возникает не просто много проблем.

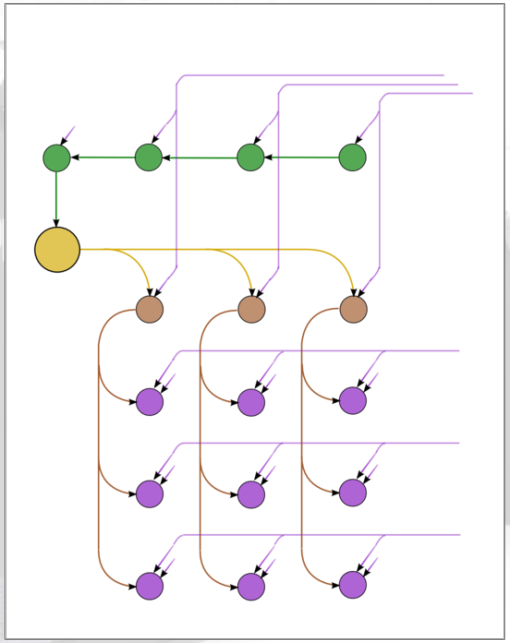
Возникает очень много сложных проблем !

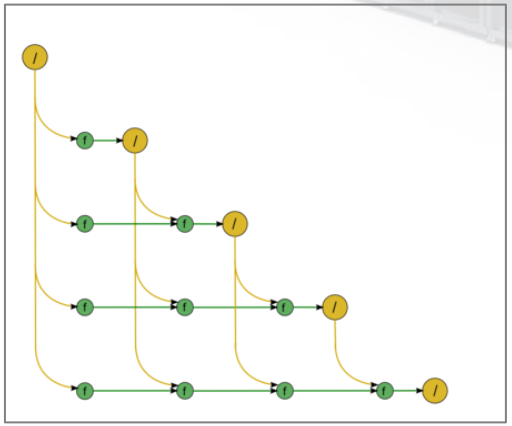
Информационная структура: как получать, описывать, показывать… ?

(сложности описания алгоритмов)





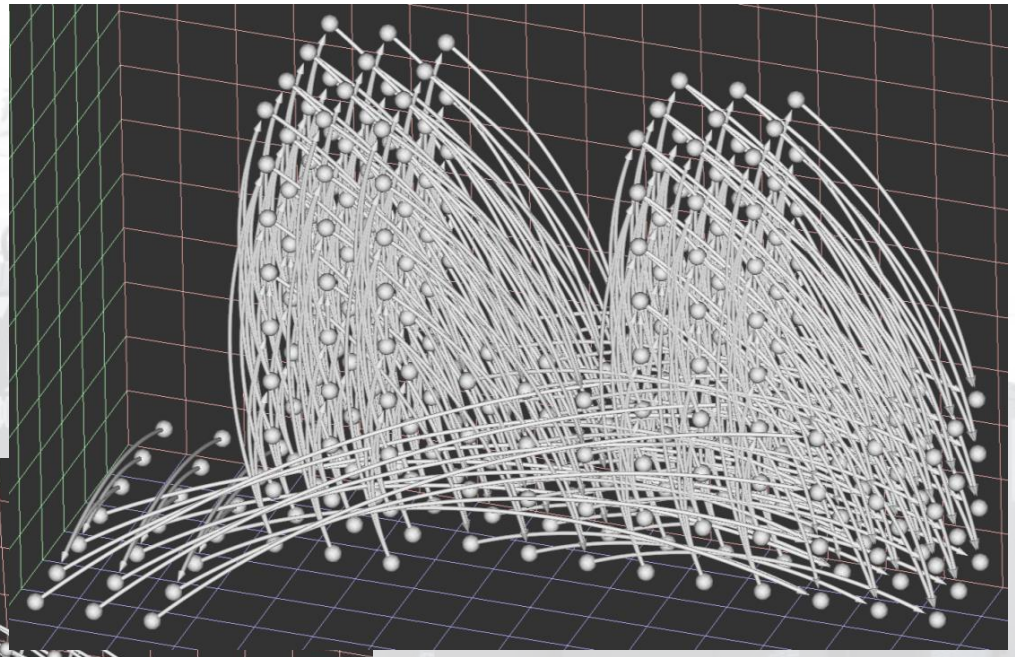


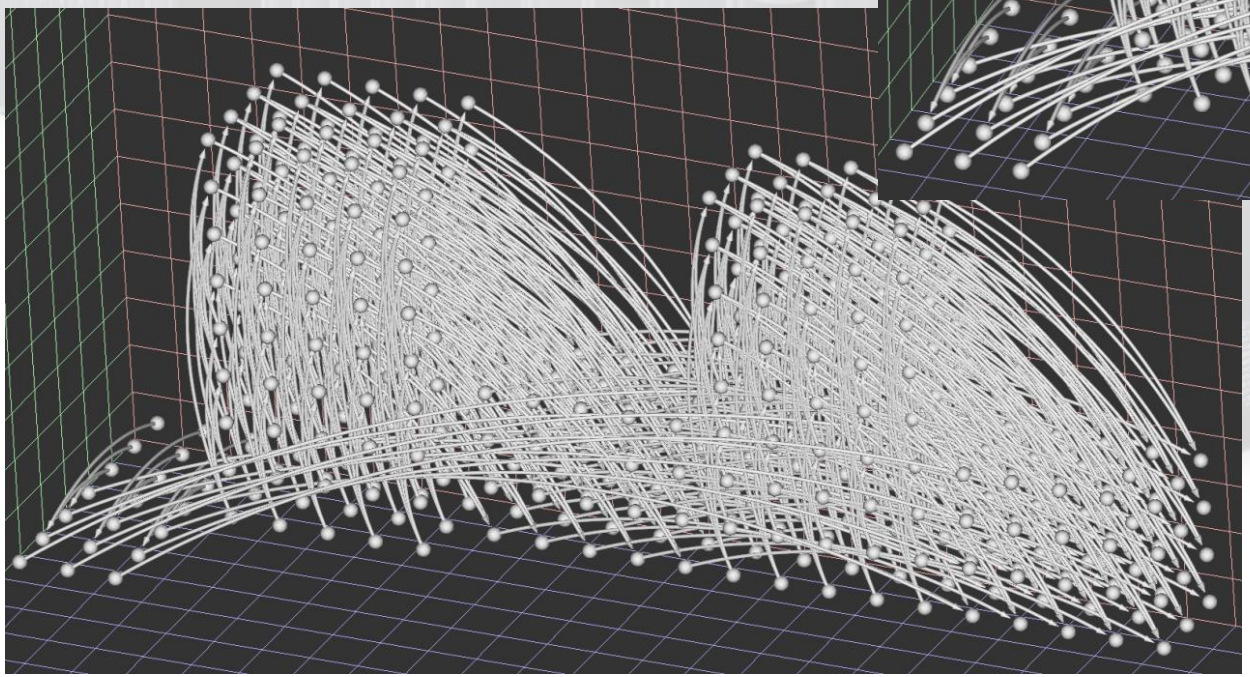


- Как изобразить потенциально бесконечный граф ?
- Как изобразить потенциально многомерный граф ?
- Как показать зависимость структуры графа от размера задачи ?

Информационная структура: как получать, описывать, показывать… ?

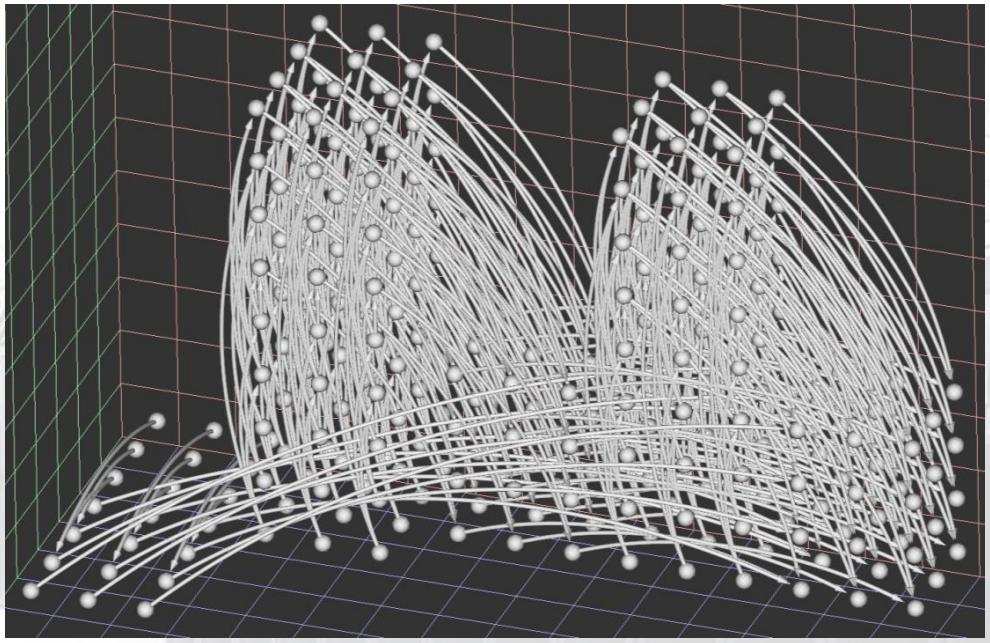
(сложности описания алгоритмов)

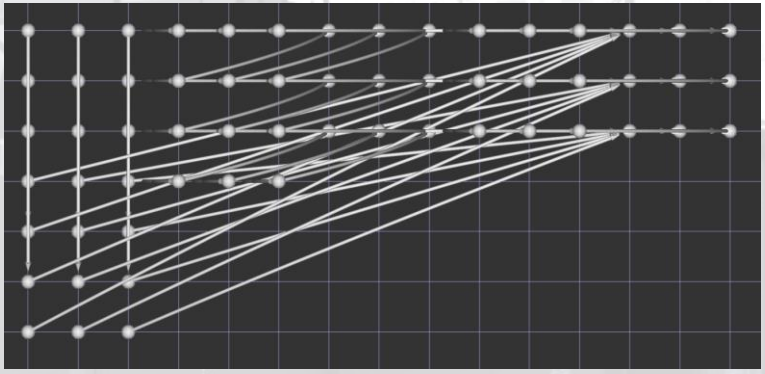


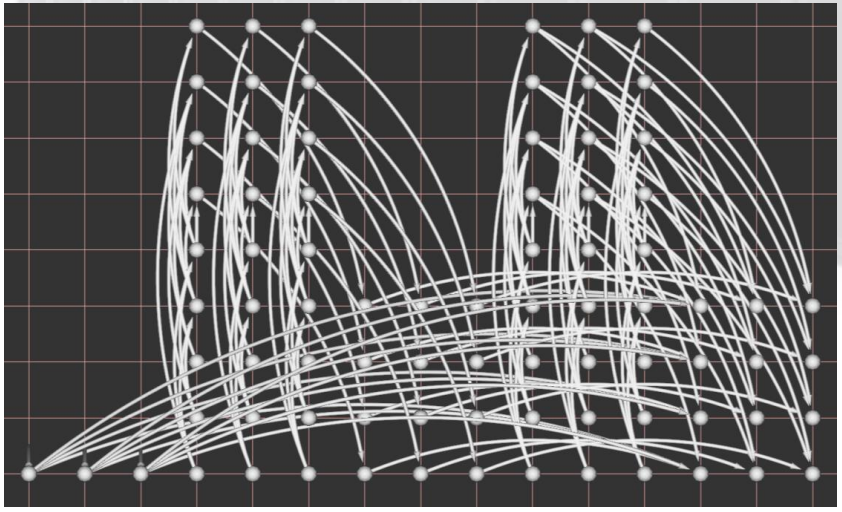


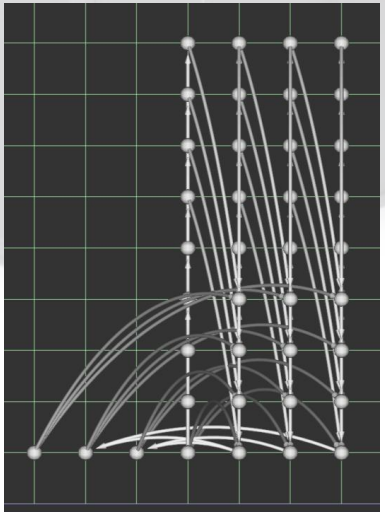
Информационная структура: как получать, описывать, показывать… ?

(сложности описания алгоритмов)

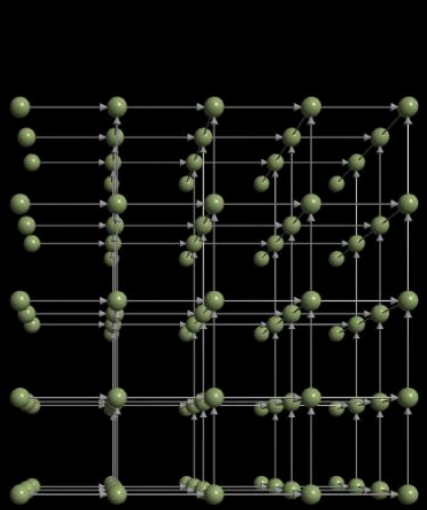


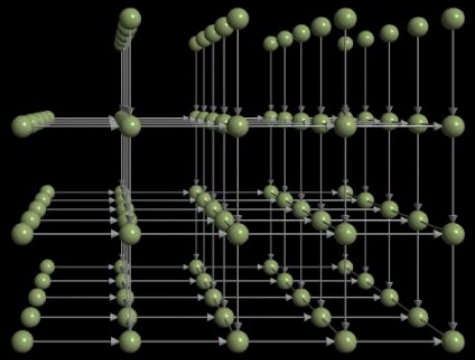


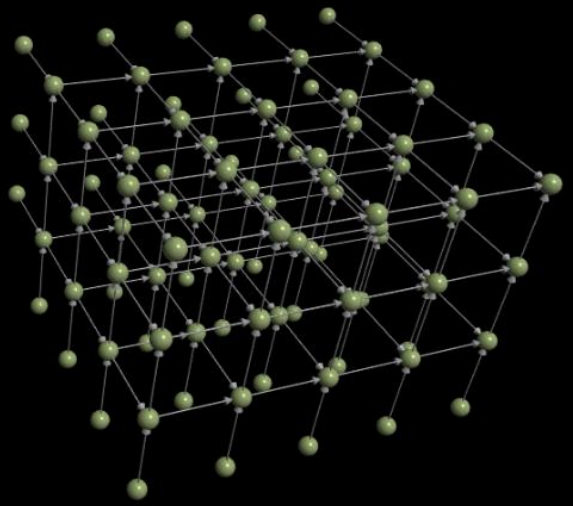


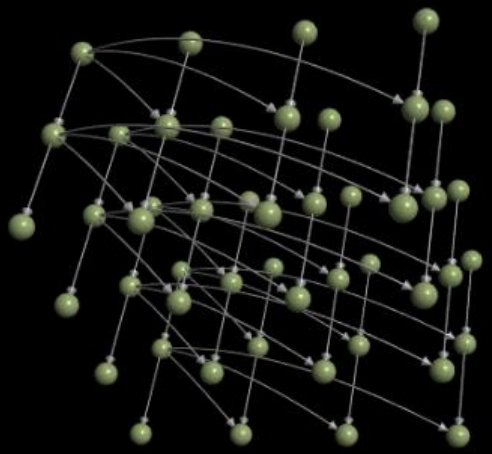


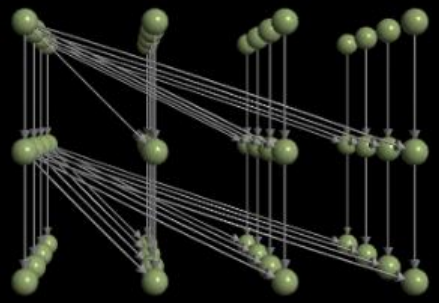
Информационная структура алгоритмов и программ

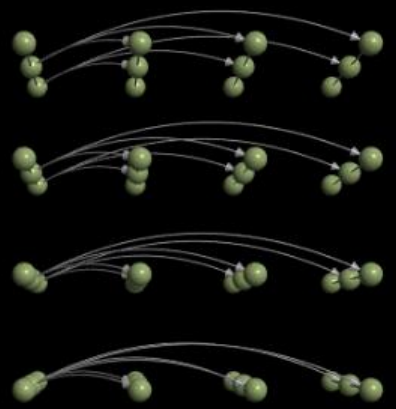




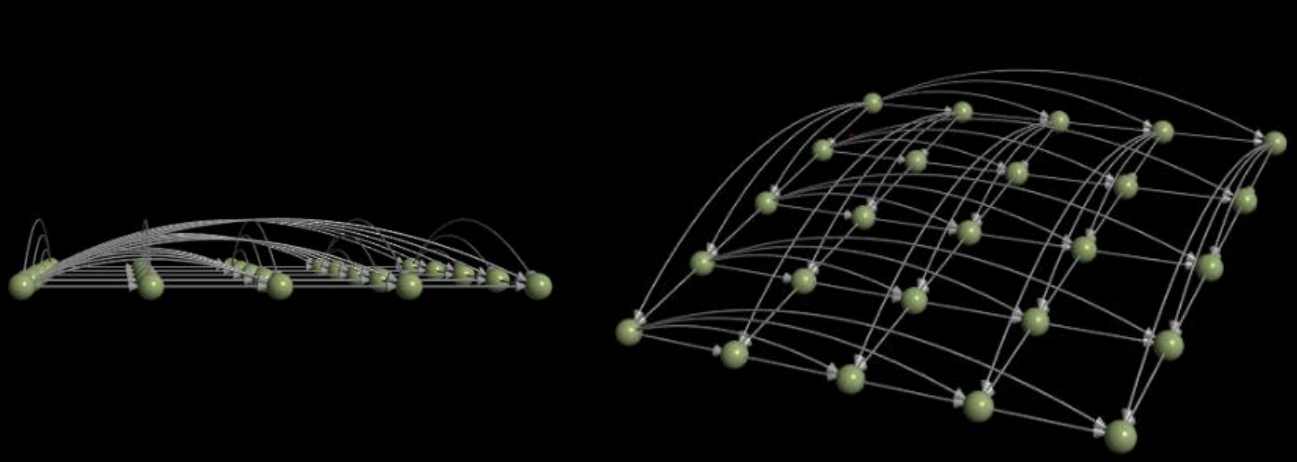




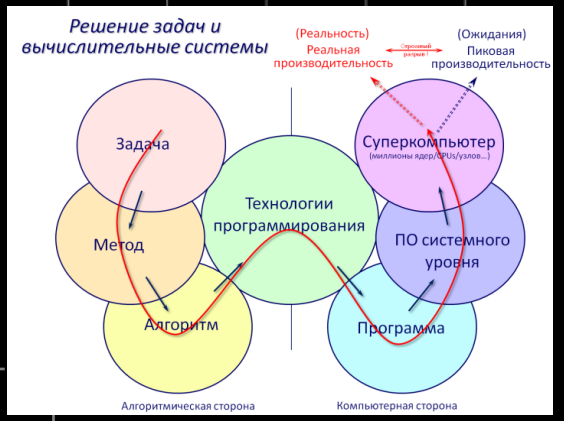


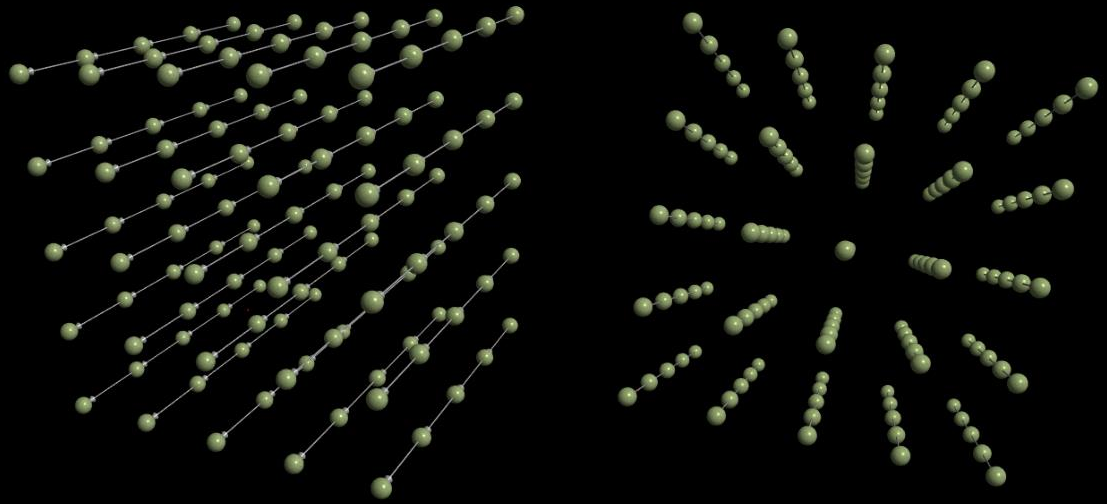


Информационная структура алгоритмов и программ

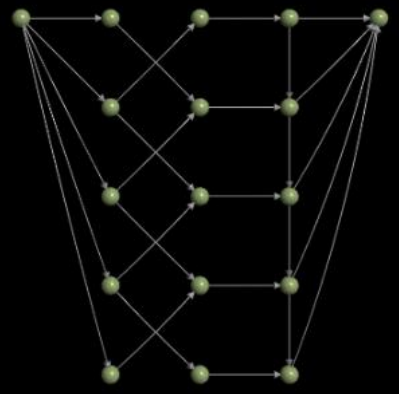


Типовые алгоритмические структуры





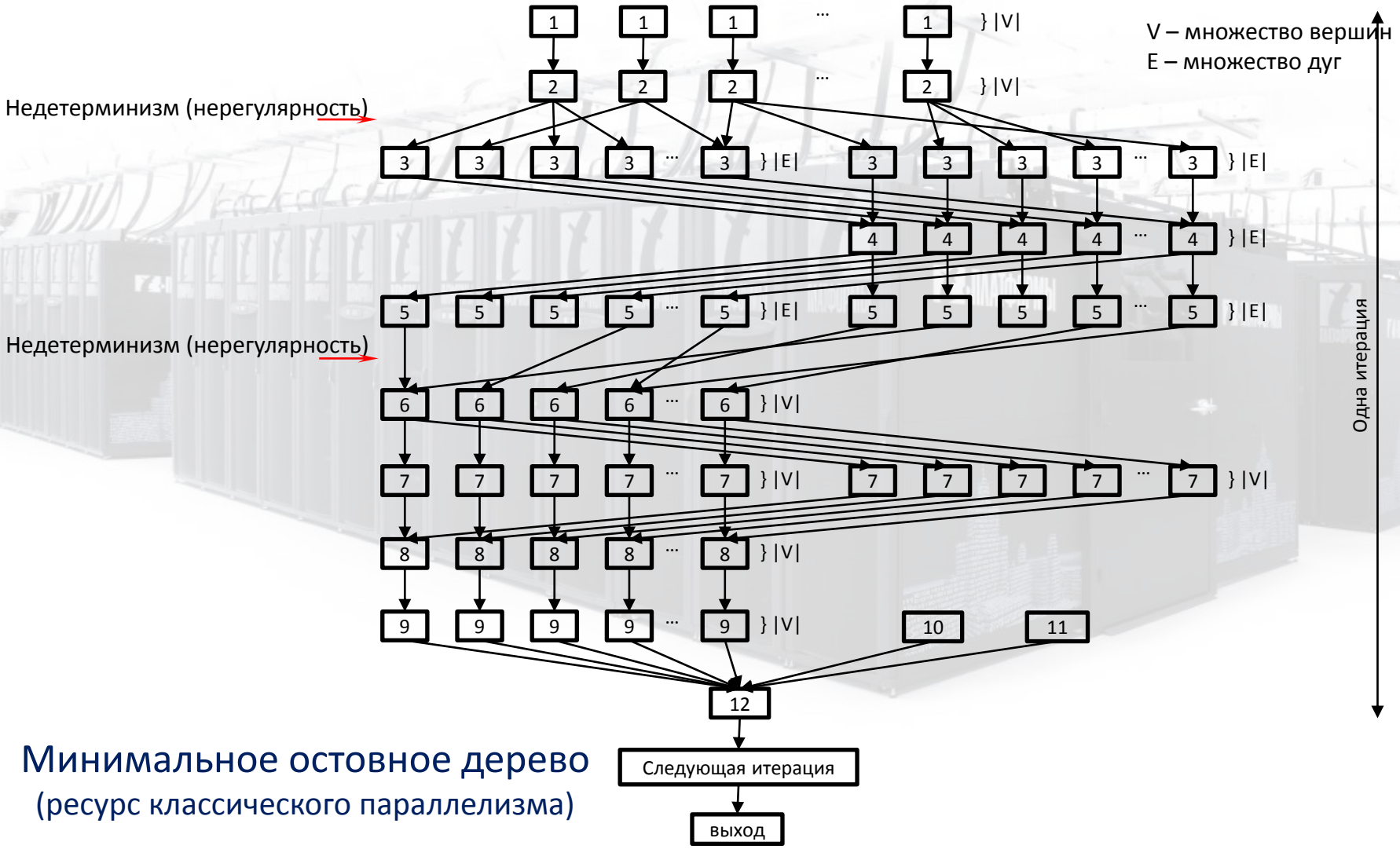
Огромный ресурс параллелизма



А такой структуры быть не может …

Потенциальный параллелизм: как находить, описывать, показывать… ?

(сложности описания алгоритмов)

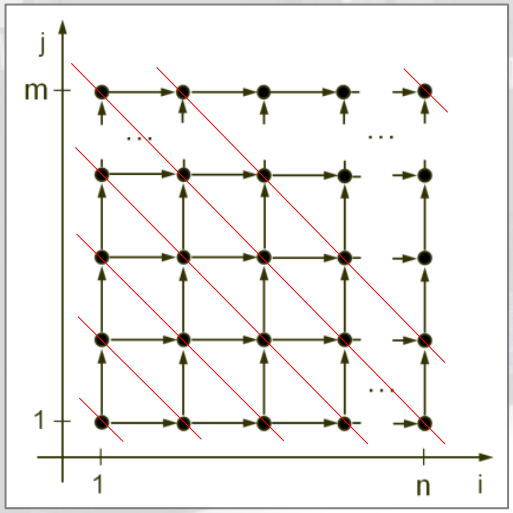


Минимальное остовное дерево

(ресурс классического параллелизма)

Возможные источники несбалансированности: это важно, это нужно отметить
(сложности описания алгоритмов)

- Баланс между арифметическими операциями +/- and * ;
- Баланс между арифметическими операциями и “чтениями/записями”;
- Баланс в числе операций, которые могут выполняться параллельно;



- Баланс между вычислительной и коммуникационной частями…

Возможные источники недетерминизма: это важно, это нужно отметить
(сложности описания алгоритмов)

• Структура входных данных (разреженные матрицы, графы произвольной структуры…);

• Итерационная природа алгоритмов;

• Датчики случайных чисел;

• Отсутствие повторяемости результатов из-за разного порядка выполнения ассоциативных операций…

Структура и свойства алгоритмов

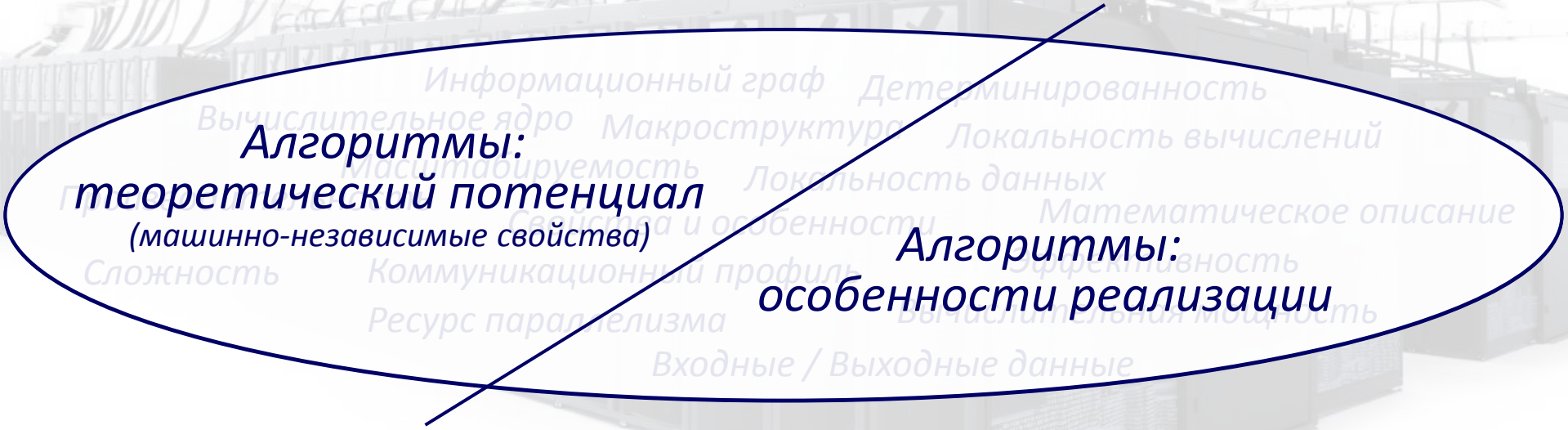
Да, можно: проект AlgoWiki

http://AlgoWiki-Project.org/

Можно ли выполнить такой анализ “раз и навсегда” ?

- Анализировать алгоритмы, чтобы понять, как их приспособить под новую компьютерную платформу ;

Структура и свойства алгоритмов
(от мобильных платформ до экзафлопсных суперкомпьютеров)



AlgoWiki

http://AlgoWiki-Project.org

Файл Правка Вид Журнал Закладки Инструменты Справка

Алговики

algowiki-project.org/ru/Открытая_энциклопедия_свойств_алгоритмов

Поиск

Войти Запрос учётной записи

Algowiki

Main page Обсуждение Читать Просмотр История Поиск

Открытая энциклопедия свойств алгоритмов

Заглавная страница
Общий форум
Технический форум
Справка
Свежие правки

Хранилище файлов
Новые файлы
Загрузить файл

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

На других языках
English

Добро пожаловать! Присоединяйтесь!

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

Структура проекта

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

Образцовая статья

Разложение Холецкого (метод квадратного корня)

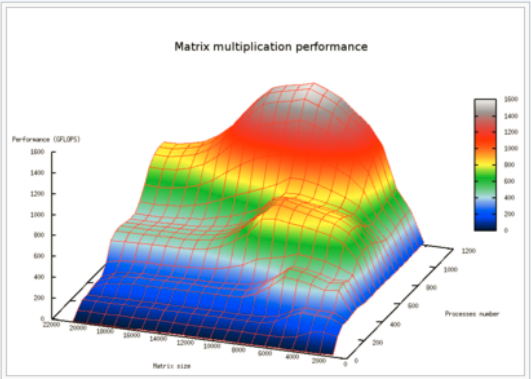
1 Свойства и структура алгоритма

1.1 Общее описание алгоритма

Свойства алгоритма:

- Последовательная сложность алгоритма: $O(n^3)$

Изображение дня



Производительность умножения плотных матриц

Организация работы

- Структура описания свойств алгоритмов
- Руководства по заполнению разделов описания
- Готовность статей
- Результаты прогона алгоритмов
- Глоссарий
- Помощь

Участники проекта

Описание алгоритмов в AlgoWiki
(предопределенная структура)

Содержание [убрать]

AlgoWiki: классификация алгоритмов

1 Linear algebra problems

1.1 Matrix and vector operations

1.1.1 Vector operations
1. Pairwise summation
1. Pairwise summation of numbers
2. Parallel prefix scan algorithm using pairwise summation
2. Uniform norm of a vector: Real version, serial-parallel variant
3. Dot product
4. The serial-parallel summation method

1.1.2 Matrix-vector operations

1.1.2.1 Multiplying a nonsingular matrix by a vector
1. Dense matrix-vector multiplication

1.1.2.2 Multiplying a matrix of special form by a vector
1. Fourier transform
1. Fast Fourier transform for composite dimension
1. Fast Fourier transform for powers-of-two
1. Cooley–Tukey Fast Fourier Transform, radix-2 case
2. Fast Fourier transform for even powers-of-two
2. Fast Fourier transform for composite dimension with small prime divisors (2,3,5,7)
2. Fast Fourier transform for prime dimension

1.1.3 Matrix operations
1. Dense matrix multiplication
1. Dense matrix multiplication (serial version for real matrices)
2. Strassen's algorithm

1.2 Matrix decompositions
Matrix decomposition problem
1. Triangular decompositions
1. Gaussian elimination (finding the LU decomposition)
1. LU decomposition using Gaussian elimination without pivoting
1. LU decomposition via Gaussian elimination
2. Gaussian elimination, compact scheme for tridiagonal matrices and its modifications
1. Compact scheme for Gaussian elimination: Dense matrix
2. Compact scheme for Gaussian elimination and its modifications: Tridiagonal matrix
1. Gaussian elimination, compact scheme for tridiagonal matrices, serial version
2. Spike doubling algorithm for the LU decomposition of a tridiagonal matrix
3. Serial-parallel algorithm for the LU decomposition of a tridiagonal matrix
2. LU decomposition using Gaussian elimination with pivoting
1. Gaussian elimination with column pivoting
2. Gaussian elimination with row pivoting
3. Gaussian elimination with diagonal pivoting
4. Gaussian elimination with complete pivoting
2. Cholesky method
1. Cholesky decomposition
2. Invertable triangular decompositions for matrices of special form
3. Unitary-triangular factorizations
1. QR decomposition of dense nonsingular matrices
1. Givens (rotations) method for the QR decomposition of a matrix
1. Givens method
2. Householder (reflections) method for the QR decomposition of a matrix
1. Householder (reflections) method for the QR decomposition of a square matrix, real pairwise version
3. Orthogonalization method
1. Classical orthogonalization method
2. Orthogonalization method with reorthogonalization
4. Triangular decomposition of a Gram matrix
2. QR decomposition methods for dense Hessenberg matrices
1. Givens (rotations) method for the QR decomposition of a (real) Hessenberg matrix
2. Householder (reflections) method for the QR decomposition of a (real) Hessenberg matrix
4. Reducing matrices to compact forms
1. Unitary reductions to Hessenberg form
1. Householder (reflections) method for reducing of a matrix to Hessenberg form
1. Classical pairwise Householder (reflections) method for reducing a matrix to Hessenberg form
2. Givens (rotations) method for reducing a matrix to Hessenberg form
1. Classical pairwise Givens (rotations) method for reducing a matrix to Hessenberg form
2. Unitary reductions to tridiagonal form
1. Householder (reflections) method for reducing to tridiagonal form
1. Householder (reflections) method for reducing a symmetric matrix to tridiagonal form
2. Householder (reflections) method for reducing a complex Hermitian matrix to symmetric tridiagonal form
2. Givens (rotations) reduction to tridiagonal form
3. Eigenvalue decomposition (finding eigenvalues and eigenvectors)
5. Unitary non-similarity reductions to compact forms
1. Unitary reduction to bidiagonal form
1. Householder (reflections) reduction of a matrix to bidiagonal form
2. Givens (rotations) reduction of a matrix to bidiagonal form
2. Singular value decomposition
1. Singular value decomposition (finding singular values and singular vectors)

1.3 Solving systems of linear algebraic equations
1. Direct methods
1. Linpack benchmark
2. Matrices of a special form
1. Triangular matrices
1. Forward substitution
2. Backward substitution
2. Bidiagonal matrices
1. Forward and backward substitution for bidiagonal matrices
2. Spike doubling algorithm for solving bidiagonal SLAEs
3. Serial-parallel variant of the backward substitution
3. Methods for solving tridiagonal SLAEs
1. Methods based on the conventional LU decomposition
1. Thomas algorithm
1. Thomas algorithm, pairwise version
2. Regessed Thomas algorithm, pairwise version
2. Spike doubling algorithm
1. Spike doubling algorithm for the LU decomposition of tridiagonal matrices
2. Spike doubling algorithm for solving bidiagonal SLAEs
3. Serial-parallel method for solving tridiagonal matrices based on the LU decomposition and backward substitutions
2. Other methods
1. Reduction method
1. Complete reduction method
2. Reduction method regessed for a new right-hand side
2. Two-sided Thomas algorithm
1. Two-sided Thomas algorithm, pairwise version
2. Regessed two-sided Thomas algorithm, pairwise version
3. Cyclic reduction
1. Complete cyclic reduction
2. Cyclic reduction regessed for a new right-hand side
4. Bordering method
3. Methods for solving block triangular matrices
1. Block forward substitution (real version)
2. Block backward substitution (real version)
3. Methods for solving block bidiagonal matrices
1. Forward and backward substitution for block bidiagonal matrices
2. Spike doubling algorithm for solving block bidiagonal matrices
3. Serial-parallel variant of the block backward substitution for solving block bidiagonal matrices
4. Methods for solving block tridiagonal matrices
1. Methods based on the conventional LU decomposition
1. Block Thomas algorithm
2. Serial-parallel method for solving block systems of linear algebraic equations based on the LU decomposition and backward substitutions
2. Other methods
1. Two-sided Thomas algorithm, block variant
2. Block cyclic reduction
3. Block bordering method
3. Solving systems of linear algebraic equations with coefficient matrices of special form whose inverses are known
2. Iterative methods for solving systems of linear algebraic equations
1. High Performance Conjugate Gradient (HPCG) benchmark
2. Biconjugate gradients stabilized method (BiCGStab)
3. Kaczmarz's algorithm

1.4 Solving eigenvalue problems
1. Eigenvalue decomposition (finding eigenvalues and eigenvectors)
1. QR algorithm
1. QR algorithm as implemented in SCALAPACK
1. Classical pairwise Householder (reflections) method for reducing a matrix to Hessenberg form
2. Hessenberg QR algorithm as implemented in SCALAPACK
2. Symmetric QR algorithm as implemented in SCALAPACK
1. Householder (reflections) method for reducing a symmetric matrix to tridiagonal form
2. Symmetric tridiagonal QR algorithm as implemented in SCALAPACK
3. QR algorithm for complex Hermitian matrices as implemented in SCALAPACK
1. Householder (reflections) method for reducing a complex Hermitian matrix to symmetric tridiagonal form
2. Symmetric tridiagonal QR algorithm as implemented in SCALAPACK
2. The Jacobi (rotations) method for solving the symmetric eigenvalue problem
1. The classical Jacobi (rotations) method with pivoting for symmetric matrices
2. Serial Jacobi (rotations) method for symmetric matrices
3. Serial Jacobi (rotations) method with thresholds for symmetric matrices
3. Lanczos algorithm
1. Lanczos algorithm in exact algorithm (without reorthogonalization)
2. Partial eigenvalue problem
1. Method of bisection
3. Singular value decomposition (finding singular values and singular vectors)
1. Jacobi (rotations) method for finding singular values
1. Serial Jacobi (rotations) method for finding singular values
2. Jacobi method with a special choice of rotations for finding singular values
2. QR algorithm as applied to singular value decomposition arrangums

1.5 Algebra of polynomials
1. Horners method

2 Algorithms on lists and arrays

2.1 Search algorithms
1. Linear search: Finding an item in an arbitrary list; $O(n)$
2. Binary search: Finding the position of a target value within a sorted array; $O(\log(n))$

2.2 Sorting algorithms
1. Binary tree sort
2. Bubble sort
3. Merge sort (serial and parallel variants)

2.3 Graph algorithms
1. Graph traversal
1. Breadth-first search (BFS)
2. Depth-first search (DFS)
2. Single Source Shortest Path (SSSP)
1. Breadth-first search (BFS) (for unweighted graphs)
2. Dijkstra's algorithm
3. Bellman-Ford algorithm
4. Δ-stepping algorithm
3. All Pairs Shortest Path (APSP)
1. Johnson's algorithm
2. Floyd–Warshall algorithm
4. Transitive closure of a directed graph
1. Purdom's algorithm
5. Longest shortest path
6. Construction of the minimum spanning tree (MST)
1. Boruvka's algorithm
2. Kruskal's algorithm
3. Prim's algorithm
4. GHS algorithm
7. Search for isomorphic subgraphs
1. Ullman's algorithm
2. VF2 algorithm
8. Graph connectivity
1. Shiloach–Vishkin algorithm for finding the connected components
2. Dinic's algorithm
3. Tarjan's strongly connected components algorithm
4. OCSC algorithm for finding the strongly connected components
5. Tarjan's biconnected components algorithm
6. Tarjan–Vishkin biconnected components algorithm
7. Tarjan's algorithm for finding the bridges of a graph
8. Vertex connectivity of a graph
9. Gabow's edge connectivity algorithm
9. Finding maximal flow in a transportation network
1. Ford–Fulkerson algorithm
2. Prefow–Push algorithm
10. Finding minimal-cost flow in a transportation network
11. Assignment problem
1. Hungarian algorithm
2. Jiuzhen algorithm
3. Hopcroft–Karp algorithm
12. Betweenness centrality algorithm

3 Computational geometry
1. Finding the diameter of a point set
2. Finding the convex hull of a point set
3. Delaunay triangulation
4. Voronoi diagram
5. Point-in-polygon problem
6. Convex polygon intersection
7. Star-shaped polygon intersection

3.1 Computer graphics
1. Line drawing algorithms: Approximating a line segment on discrete graphical media
2. Determining visible parts of a three-dimensional scene
3. Ray tracing: Rendering realistic images
4. Global illumination: Regarding direct illumination and reflection from other objects

4 Computer analysis and modeling

4.1 Computer benchmarks
1. High Performance Conjugate Gradient (HPCG) benchmark
2. Linpack benchmark

4.2 Algorithms of quantum system simulation
1. Algorithms of quantum computation simulation
1. Single-qubit transform of a state vector
2. Two-qubit transform of a state vector
3. Quantum Fourier transform simulation

Благодарю за внимание!

voevodin@parallel.ru