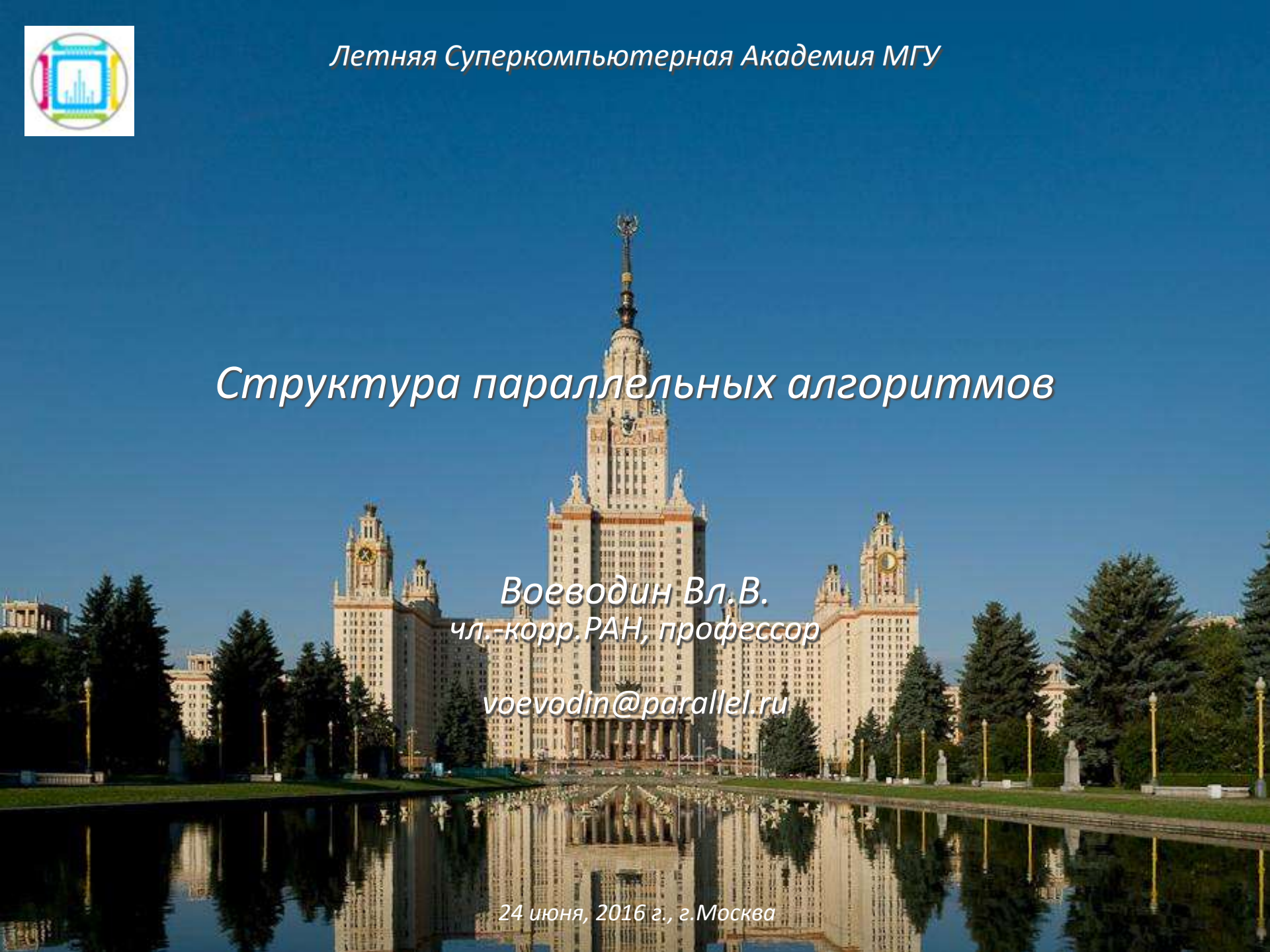


Летняя Суперкомпьютерная Академия МГУ

Структура параллельных алгоритмов

Воеводин Вл.В.
чл.-корр.РАН, профессор

voevodin@parallel.ru

24 июня, 2016 г., г.Москва

Суперкомпьютер МГУ “Ломоносов”

Суперкомпьютер МГУ “Ломоносов-2”

Суперкомпьютерный центр МГУ сегодня:
Пользователи: 2955
Проекты: 880

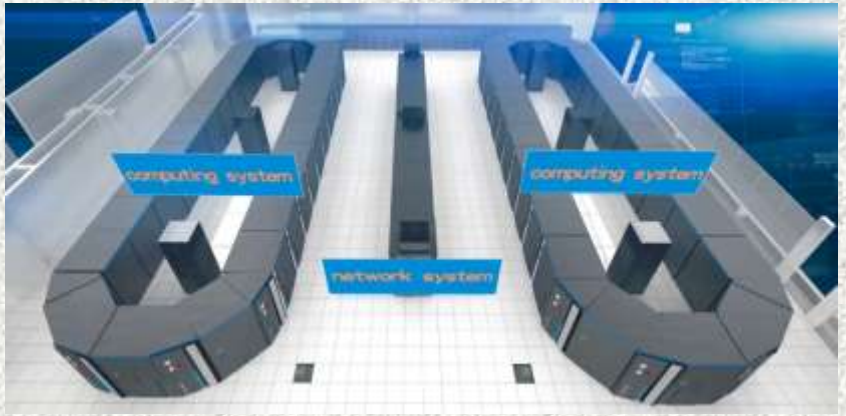
Факультеты / Институты МГУ: 21
Институты РАН: 95
Университеты России: 102

1 стойка = 256 узлов: Intel Xeon (14c) + NVIDIA K40= 515 Tflop/s

Суперкомпьютер “Ломоносов-2” (6 стоек) = 3 Pflop/s

Суперкомпьютер Sunway TaihuLight System, Китай

(#1 Top500 в 2016 г.)



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

Всего: 10 649 600 ядер

1,31 PBytes
15,4 MW (6 Gflops/W)

Производительность:
Peak: 125,4 Pflop/s
Linpack: 93 Pflop/s (74%)

Хорошо ли мы знаем свойства, особенности, статические и динамические характеристики параллельных алгоритмов?

Хорошо ли мы знаем свойства, особенности, статические и динамические характеристики параллельных алгоритмов?

Насколько хорошо мы знаем архитектуру параллельных компьютеров ?

А должны ли мы думать об архитектуре?

Да… К сожалению, Да…

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

Суперкомпьютер Cray-1

Векторно-конвейерные компьютеры

Середина 70-х годов.

Особенности архитектуры: векторные функциональные устройства, зацепление функциональных устройств, векторные команды в системе команд, векторные регистры.

Программирование: векторизация самых внутренних циклов.

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

Суперкомпьютер Cray X-MP

Суперкомпьютер Cray Y-MP

Векторно-параллельные компьютеры

Начало 80-х годов.

Особенности архитектуры: векторные функциональные устройства, зацепление функциональных устройств, векторные команды в системе команд, векторные регистры.
Небольшое число процессоров объединяются над общей памятью.

Программирование: векторизация самых внутренних циклов и распараллеливание на внешнем уровне, единое адресное пространство, локальные и глобальные переменные.

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

Суперкомпьютер Cray T3D

Суперкомпьютер Intel Paragon XPS140

Массивно-параллельные компьютеры

Начало 90-х годов.

Особенности архитектуры: тысячи процессоров объединяются с помощью коммуникационной сети по некоторой топологии, распределенная память.

Программирование: обмен сообщениями, отсутствие единого адресного пространства, PVM, Message Passing Interface. Необходимость выделения массового параллелизма, явного распределения данных и согласования параллелизма с распределением.

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

DEC AlphaServer

Суперкомпьютер Sun StarFire

Параллельные компьютеры с общей памятью

Середина 90-х годов.

Особенности архитектуры: сотни процессоров объединяются над общей памятью.

Программирование: единое адресное пространство, локальные и глобальные переменные, Linda, OpenMP.

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

Суперкомпьютер МГУ “Чебышев”

“K” суперкомпьютер

Кластеры из узлов с общей памятью

Начало 2000-х.

Особенности архитектуры: большое число многопроцессорных узлов объединяются вместе с помощью коммуникационной сети по некоторой топологии, распределенная память; в рамках каждого узла несколько (многоядерных) процессоров объединяются над общей памятью.

Программирование: неоднородная схема MPI+OpenMP;
необходимость выделения массового параллелизма, явное распределение данных, обмен сообщениями на внешнем уровне; распараллеливание в едином адресном пространстве, локальные и глобальные переменные на уровне узла с общей памятью.

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

Суперкомпьютер МГУ “Ломоносов”

Суперкомпьютер Tianhe-2

Кластеры из узлов с общей памятью с ускорителями

Середина 2000-х.

Особенности архитектуры: большое число многопроцессорных узлов объединяются вместе с помощью коммуникационной сети по некоторой топологии, распределенная память; в рамках каждого узла несколько (многоядерных) процессоров объединяются над общей памятью; на каждом узле несколько ускорителей (GPU, Phi).

Программирование:
MPI+OpenMP+OpenCL/CUDA;

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)
…

Виден ли конец процессу переписывания программ?..

Можно ли выполнить такой анализ “раз и навсегда” ?

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…

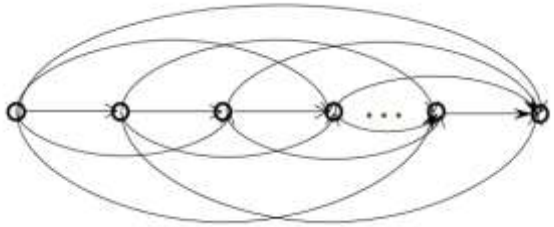
Хорошо ли мы знаем свойства, особенности, статические и динамические характеристики параллельных алгоритмов?

Какие свойства алгоритмов важны?
На какие свойства алгоритмов
нужно обращать внимание? (анализ, обучение)

И простые свойства могут быть важны…

(объем входных/выходных данных)

Нахождение транзитивного замыкания графа:



На входе: n вершин, n-1 дуга.

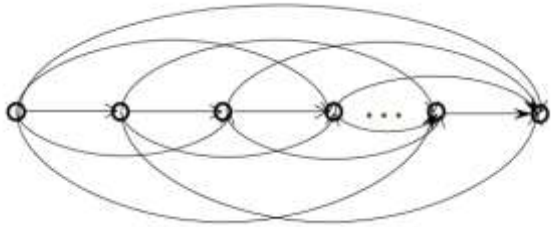
На выходе: n вершин, n(n-1)/2 дуга.

Социальные сети: 10^8 вершин, 10^{11} дуг.

И простые свойства могут быть важны…

(объём входных/выходных данных)

Нахождение транзитивного замыкания графа:



На входе: n вершин, n-1 дуга. На выходе: n вершин, n(n-1)/2 дуга.

Социальные сети: 10^8 вершин, 10^{11} дуг.

$$\text{Вычислительная мощность алгоритма} = \frac{\text{Число операций}}{\text{Объём данных}}$$

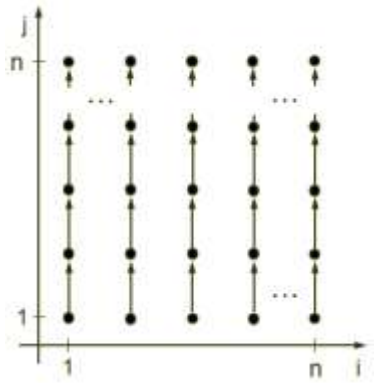
Тест Linpack (решение линейной системы): $\approx n$

Поэлементное сложение двух векторов: 1/3

Параллелизм бывает неудобным

(Что нужно знать про алгоритмы)

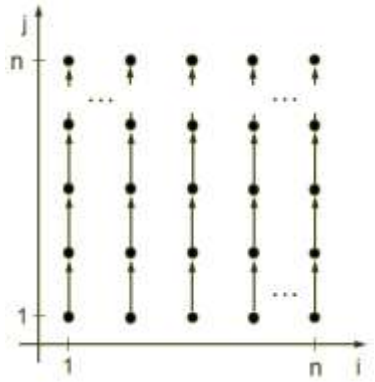
```
#pragma OpenMP parallel for
for( i = 1 ; i <= n ; ++i)
    for( j = 1 ; j <= m ; ++j)
        A[i][j] = (A[i][j] * A[i][j-1]) / 2 ;
```



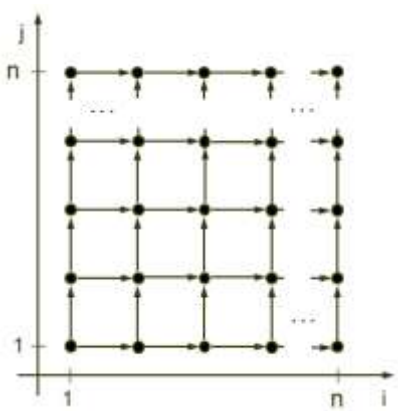
Параллелизм бывает неудобным

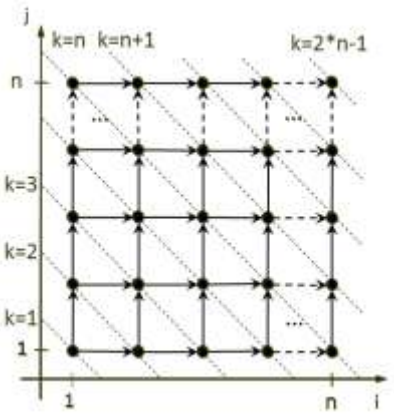
(Что нужно знать про алгоритмы)

```
#pragma OpenMP parallel for
for( i = 1 ; i <= n ; ++i)
    for( j = 1 ; j <= m ; ++j)
        A[i][j] = (A[i][j] * A[i][j-1]) / 2 ;
```



```
for( i = 1 ; i <= n ; ++i)
    for( j = 1 ; j <= m ; ++j)
        A[i][j] = (A[i-1][j] * A[i][j-1]) / 2 ;
```

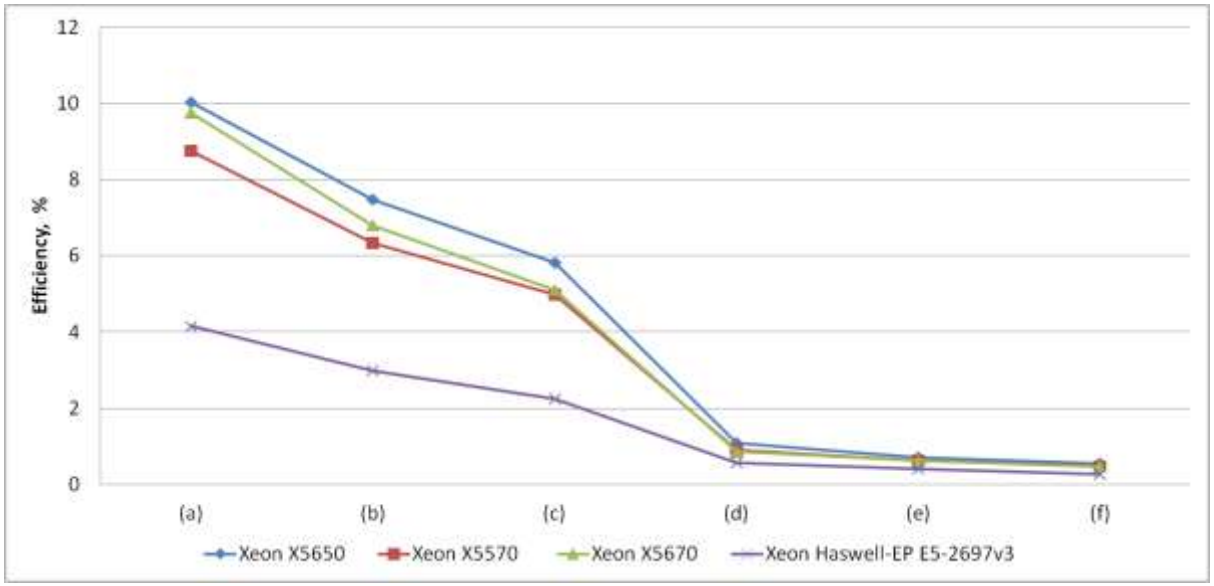




Локальность определяет многое

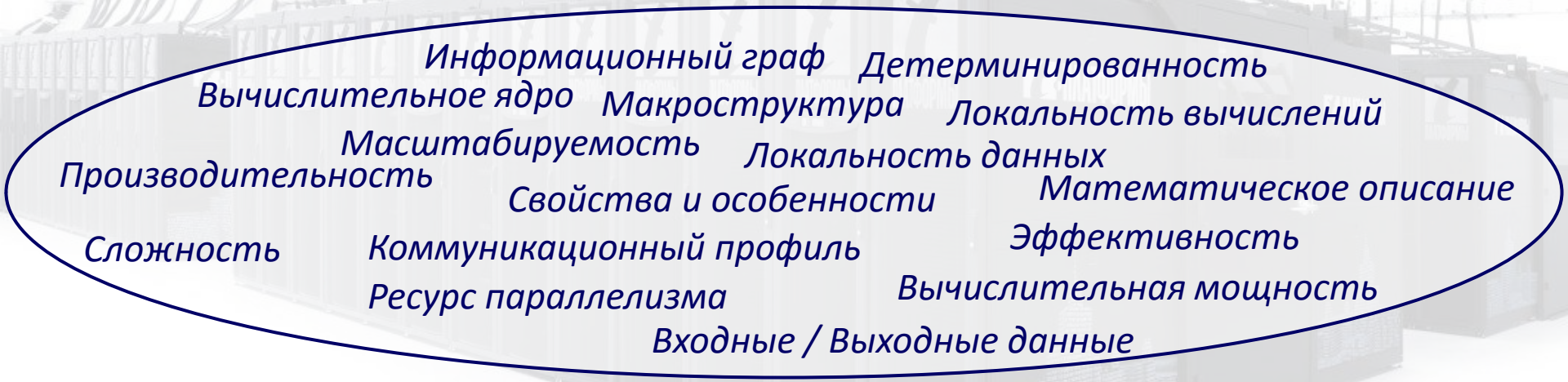
(Что нужно знать про алгоритмы)

(a) A[i] = B[i]*x + c
(b) A[i] = B[i]*x + C[i]
(c) A[i] = B[i]*X[i] + C[i]
(d) A[ind[i]] = B[ind[i]]*x+c
(e) A[ind[i]] = B[ind[i]]*x+C[ind[i]]
(f) A[ind[i]] = B[ind[i]]*X[ind[i]]+C[ind[i]]

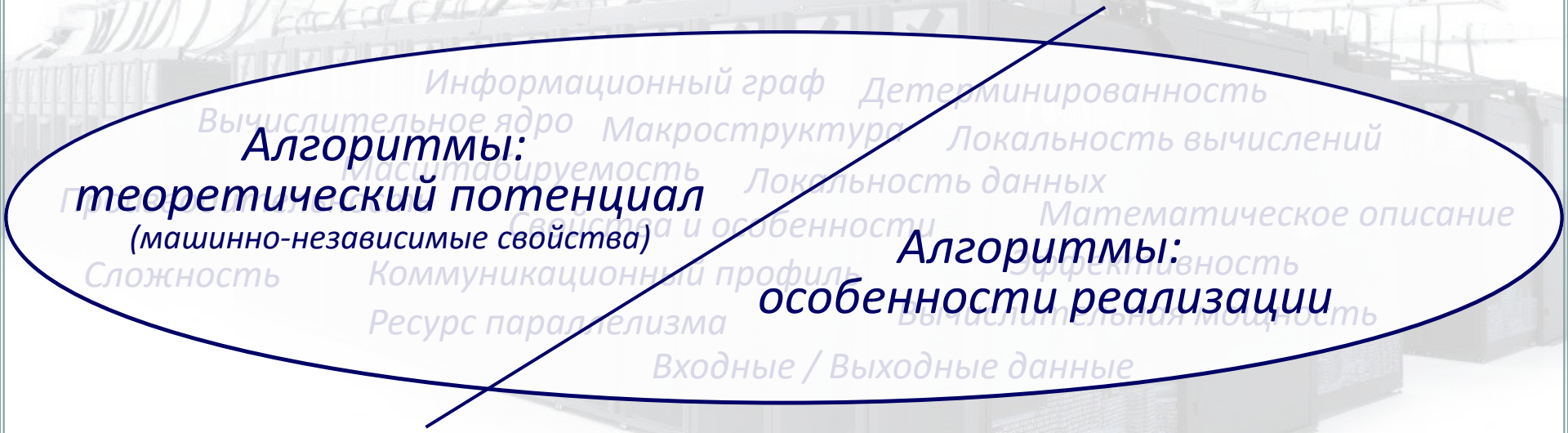


Какие свойства должны войти в “универсальное” (“полное”) описание алгоритмов?

Описание алгоритмов
(Что должно быть учтено в подобном описании?)



Описание алгоритмов
(Что должно быть учтено в подобном описании?)



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:

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

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.

Базовая схема реализации

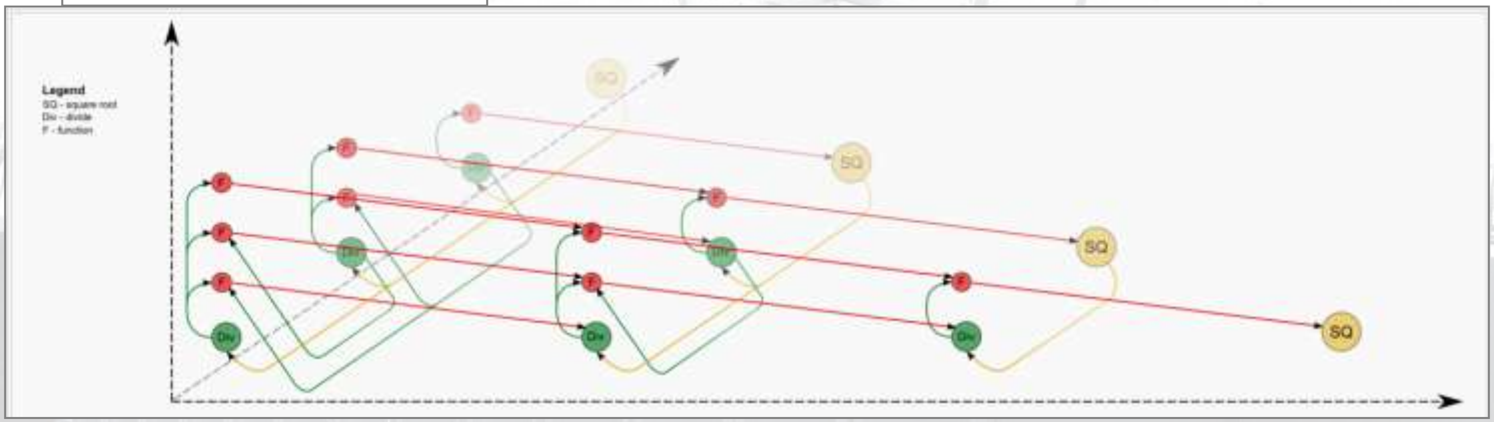
```
DO  I = 1, N
        S = A(I,I)
        DO  IP=1, I-1
                S = S - DPROD(A(I,IP), A(I,IP))
        END DO
        A(I,I) = SQRT (S)
        DO  J = I+1, N
                S = A(J,I)
                DO  IP=1, I-1
                        S = S - DPROD(A(I,IP), A(J,IP))
                END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

Дополнительная информация

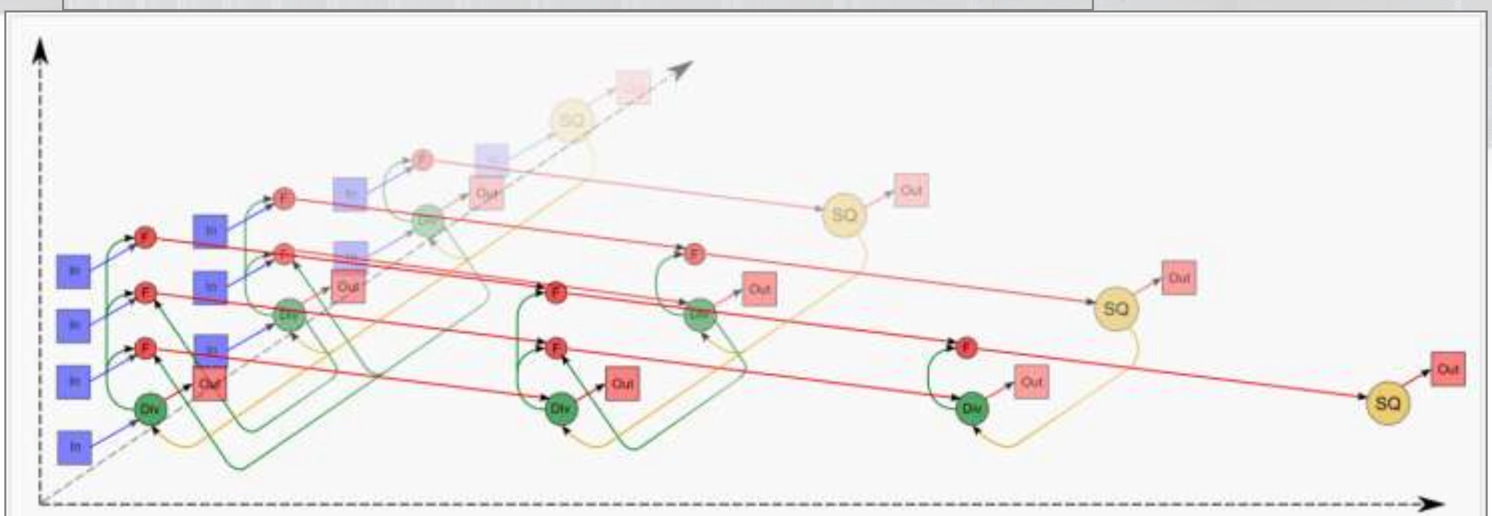
There exist block versions of this algorithm;

Описание алгоритмов
(на примере разложения Холецкого)

Информационная структура



Информационная структура с указанием входных и выходных данных



Описание алгоритмов
(на примере разложения Холецкого)

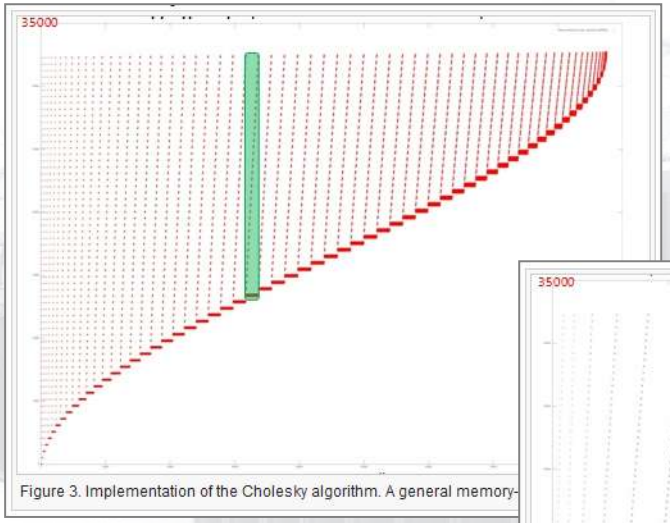


Figure 3. Implementation of the Cholesky algorithm. A general memory-

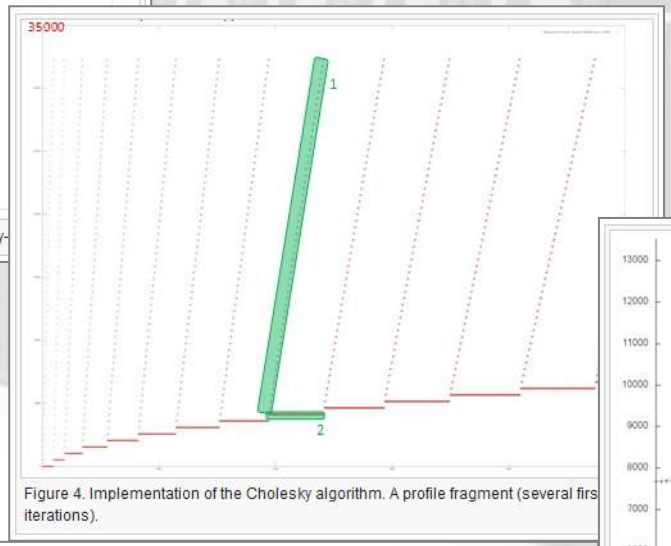


Figure 4. Implementation of the Cholesky algorithm. A profile fragment (several firs
iterations).

Локальность данных (профиль работы с памятью)

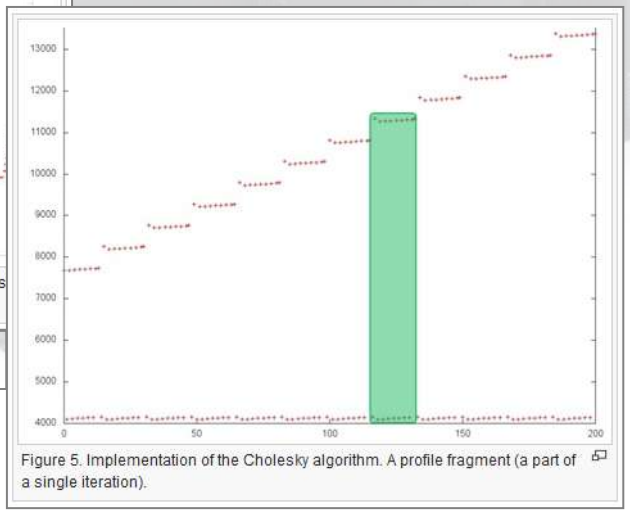
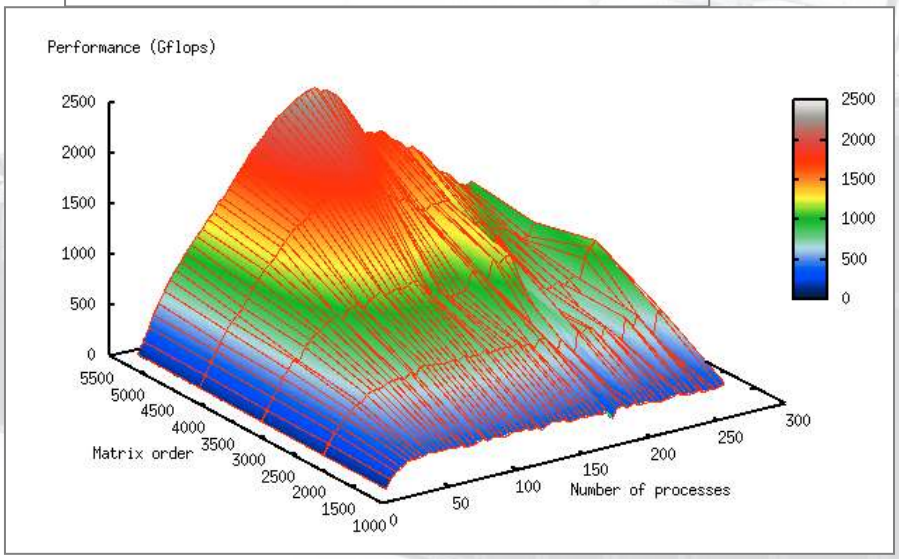


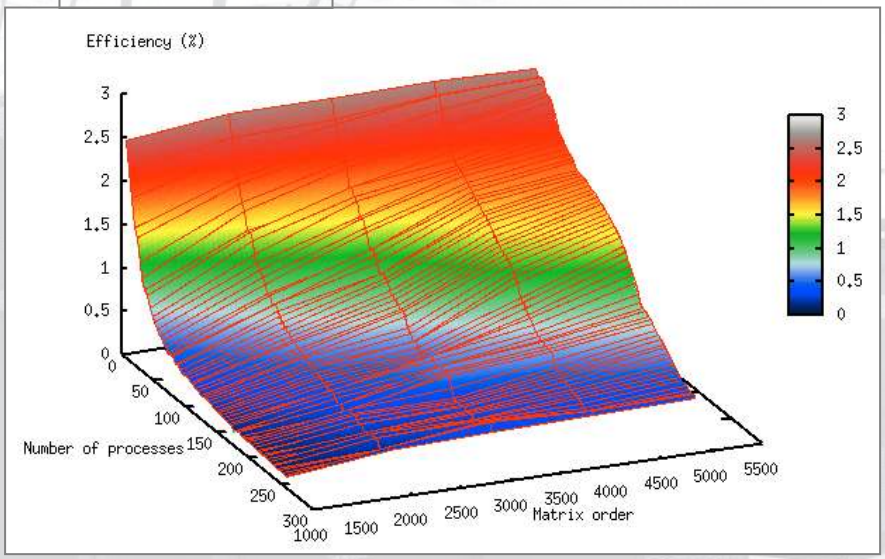
Figure 5. Implementation of the Cholesky algorithm. A profile fragment (a part of a single iteration).

Описание алгоритмов
(на примере разложения Холецкого)

Масштабируемость (производительность) *



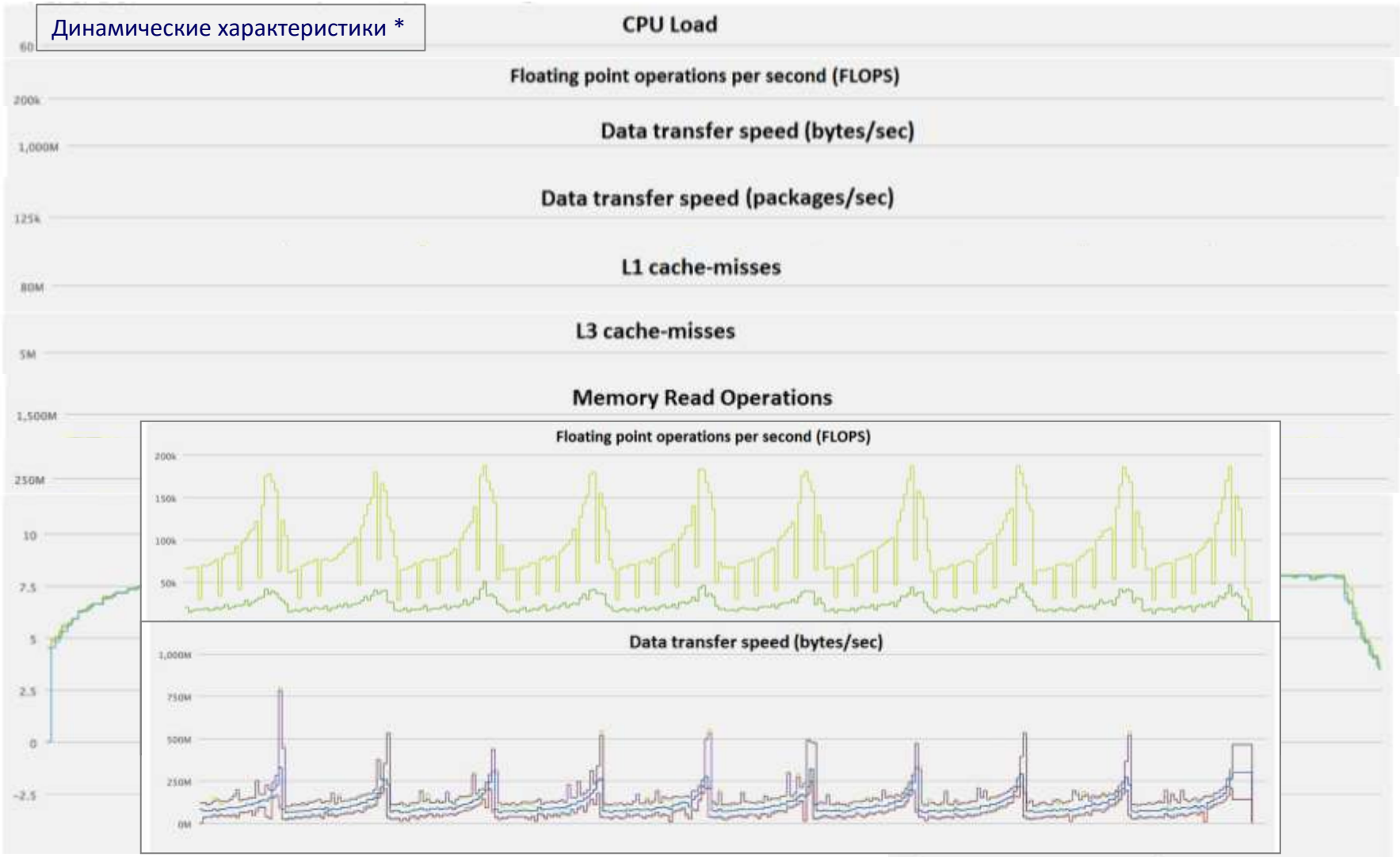
Эффективность *



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

Описание алгоритмов
(на примере разложения Холецкого)

Динамические характеристики *

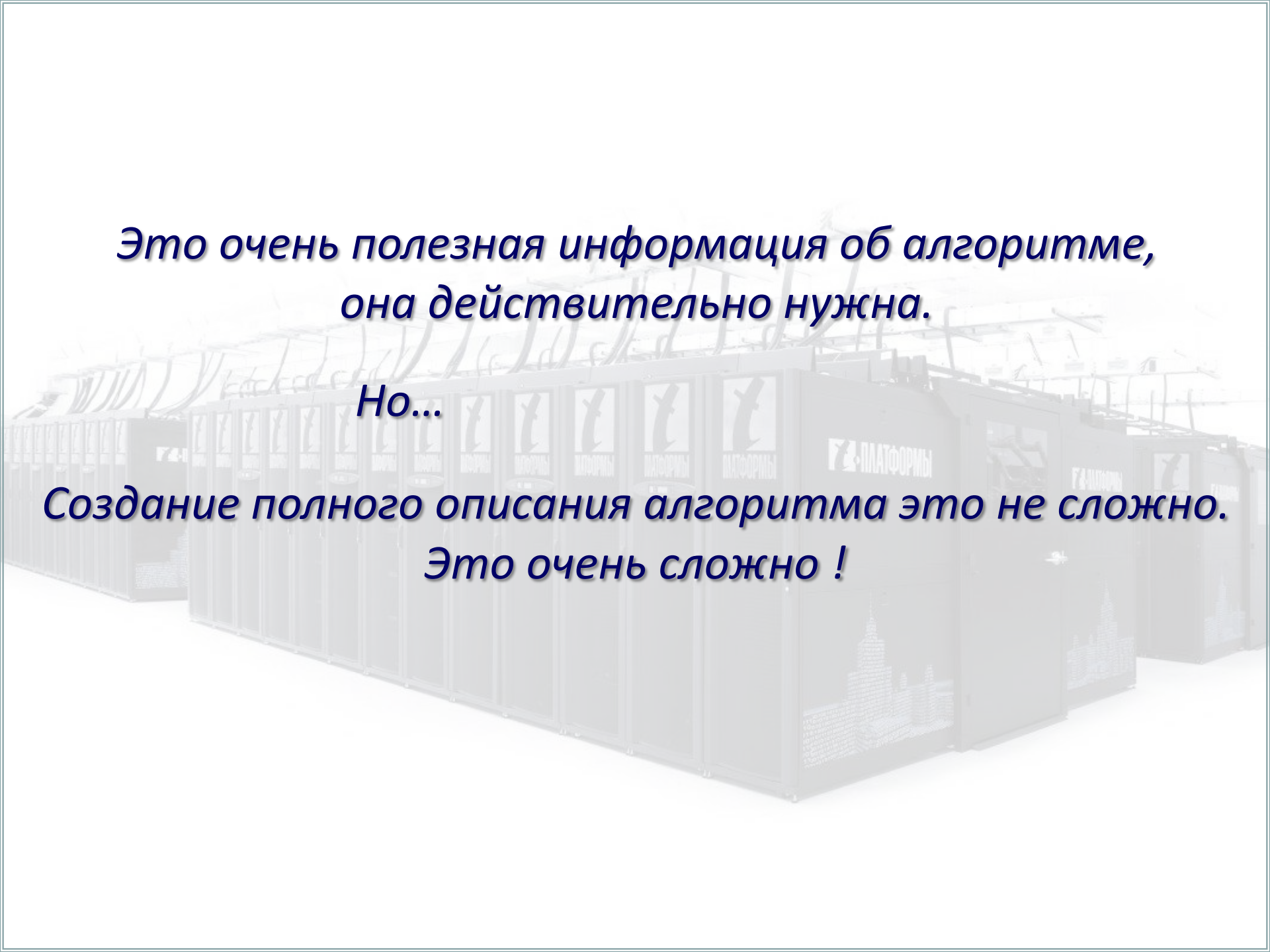


* Динамические характеристики получены на суперкомпьютере МГУ “Ломоносов”.

Это очень полезная информация об алгоритме, она действительно нужна.

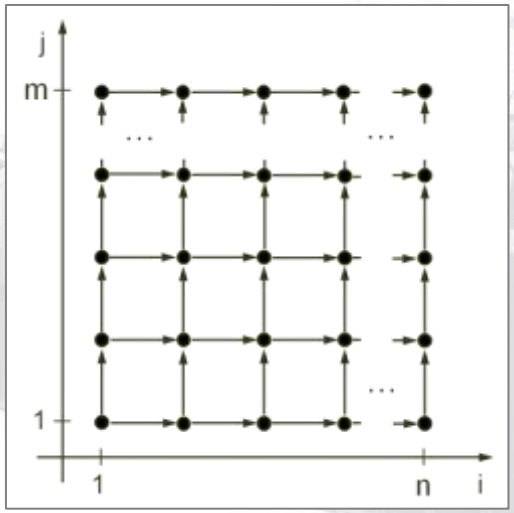
Но…

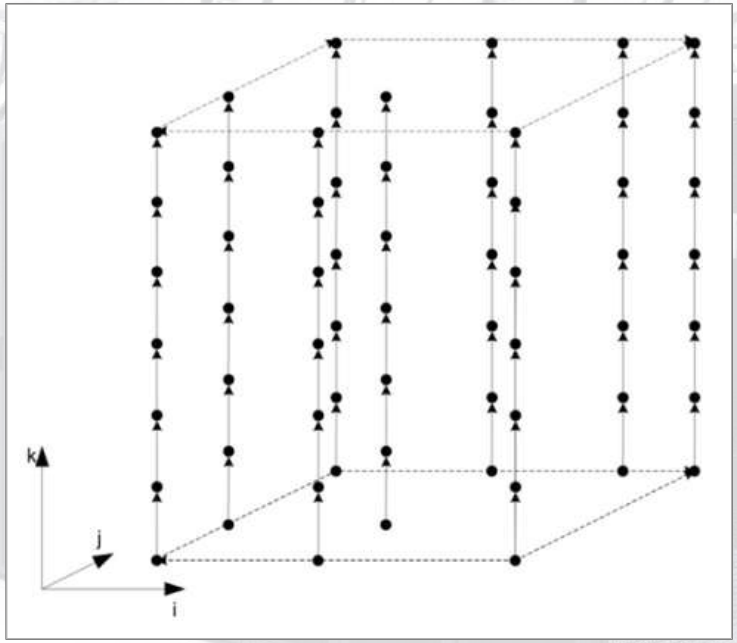
Создание полного описания алгоритма это не сложно. Это очень сложно !

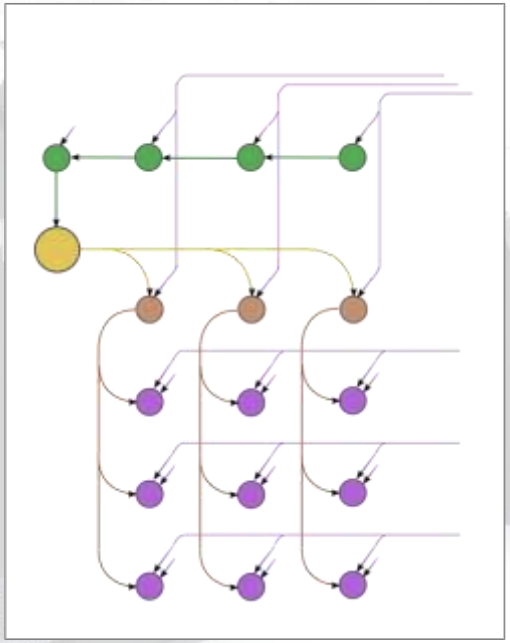


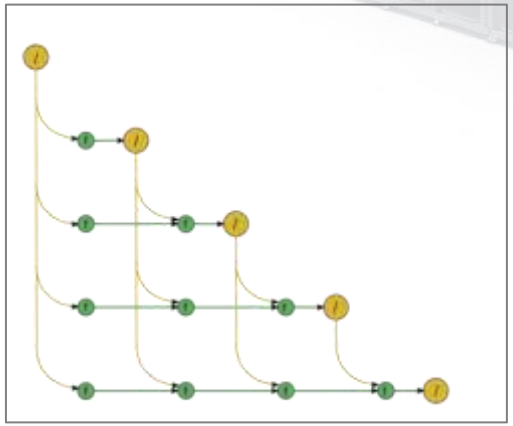
Информационная структура: как получать, описывать, показывать… ?

(сложности описания алгоритмов)





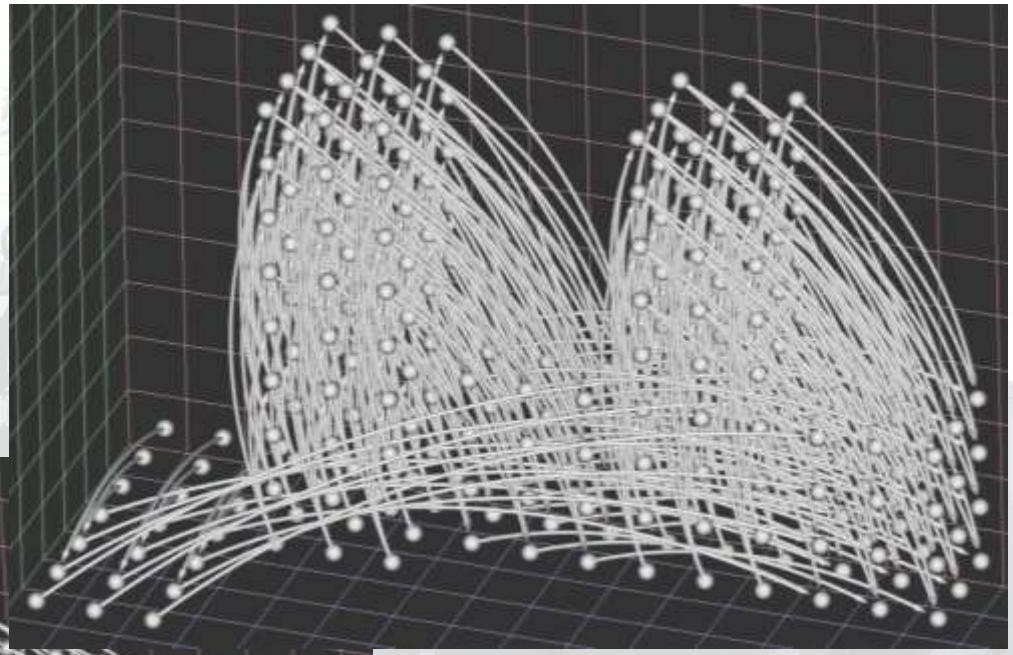


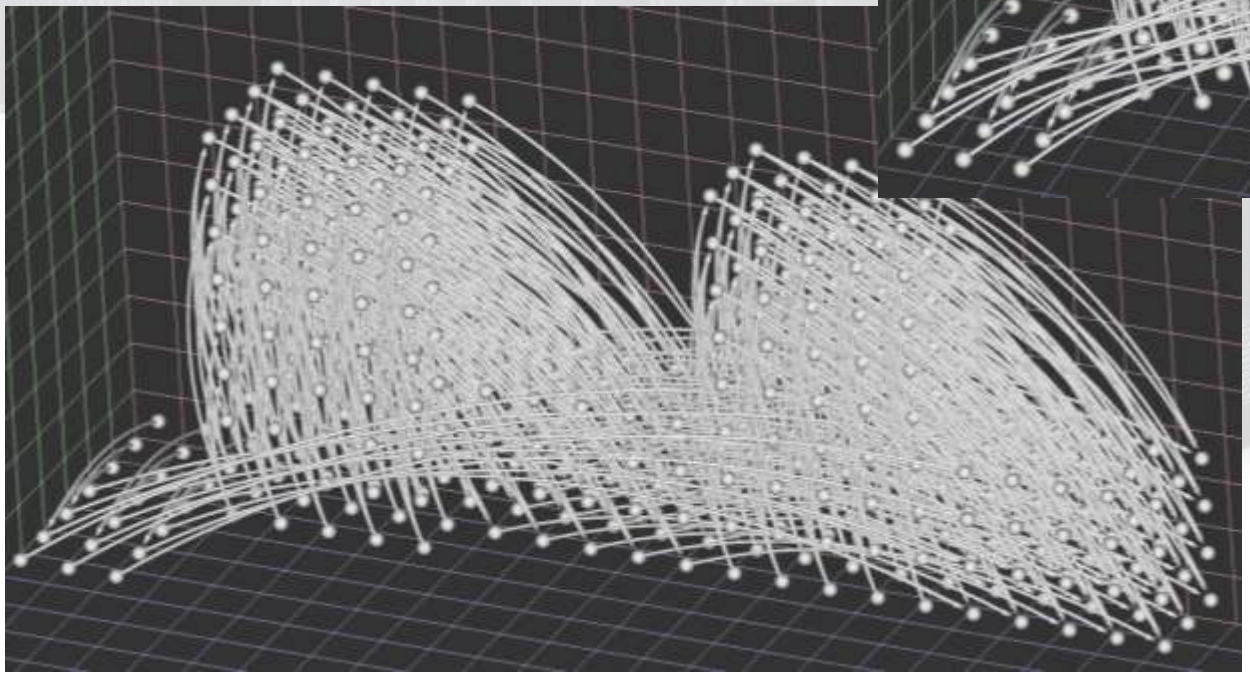


- Как изобразить потенциально бесконечный граф ?
- Как изобразить потенциально многомерный граф ?
- Как показать зависимость структуры графа от размера задачи ?

Информационная структура: как получать, описывать, показывать… ?

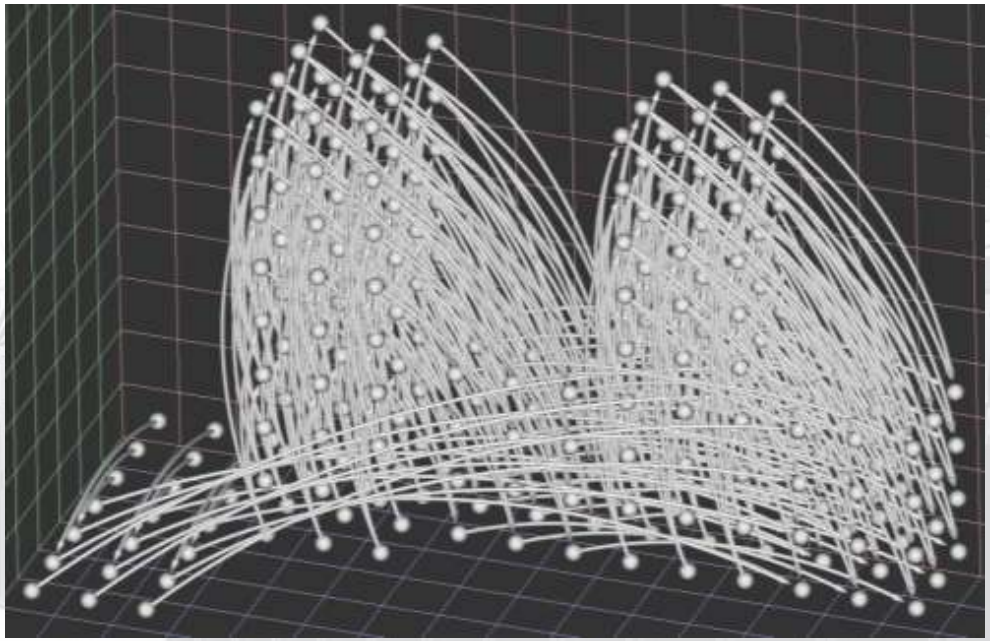
(сложности описания алгоритмов)

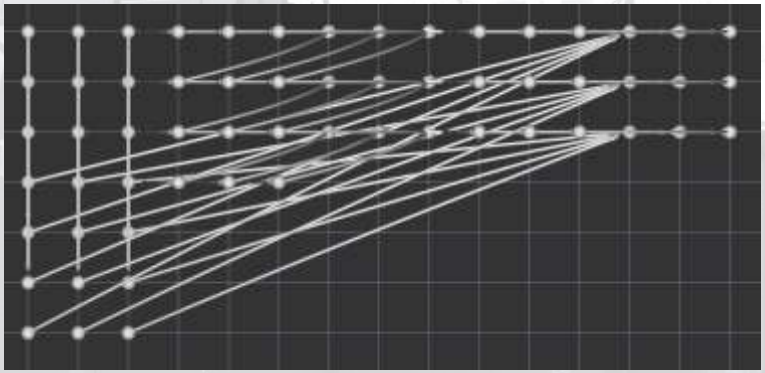


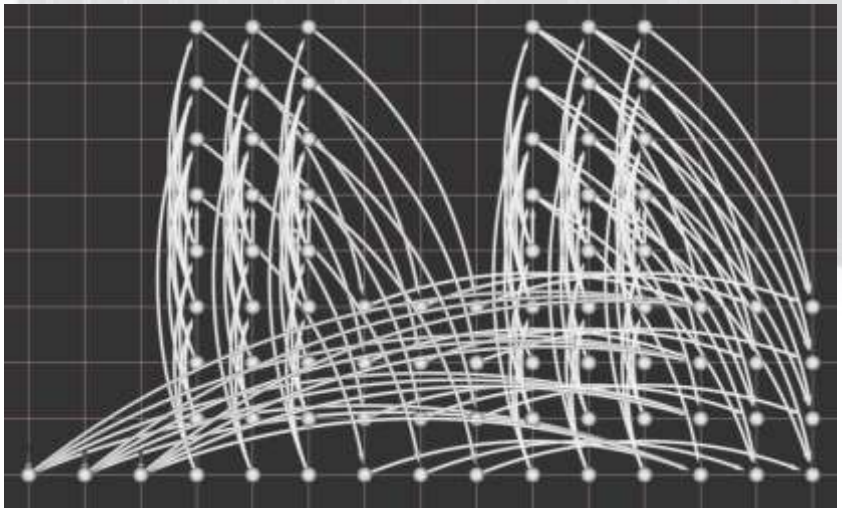


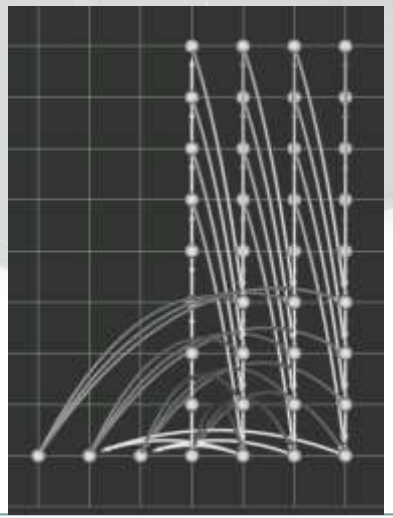
Информационная структура: как получать, описывать, показывать… ?

(сложности описания алгоритмов)



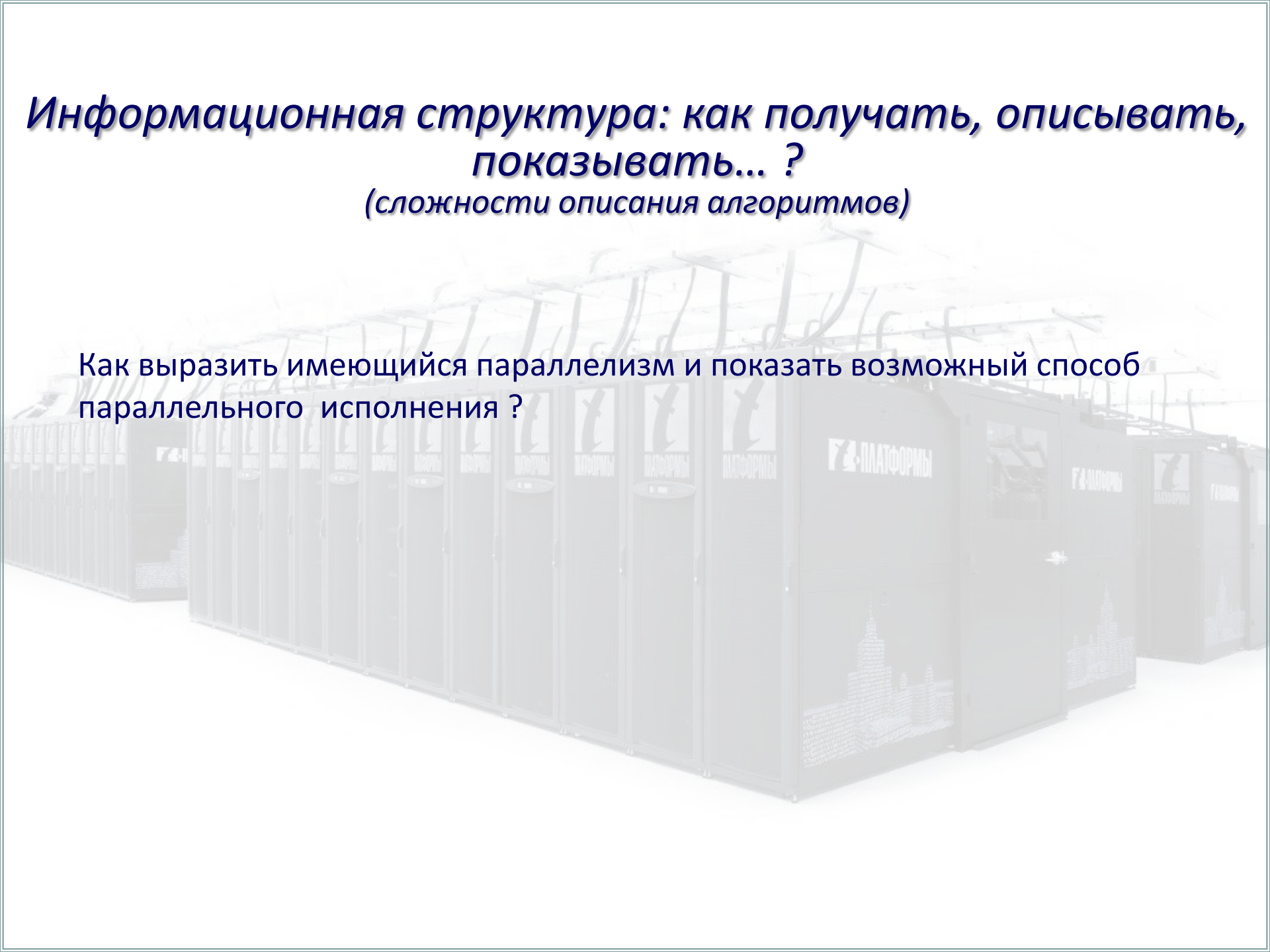






Информационная структура: как получать, описывать, показывать… ?

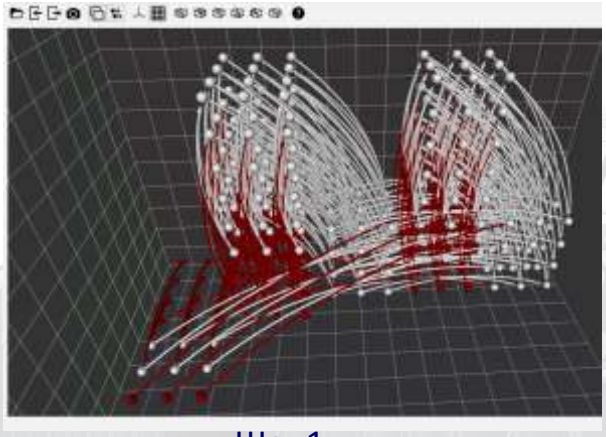
(сложности описания алгоритмов)



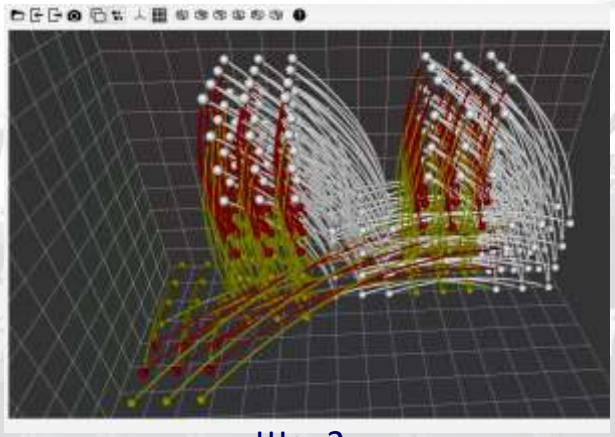
Как выразить имеющийся параллелизм и показать возможный способ параллельного исполнения ?

Информационная структура: как получать, описывать, показывать… ?

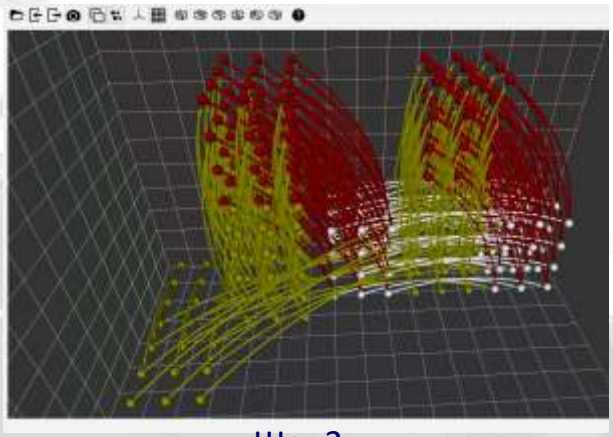
(сложности описания алгоритмов)



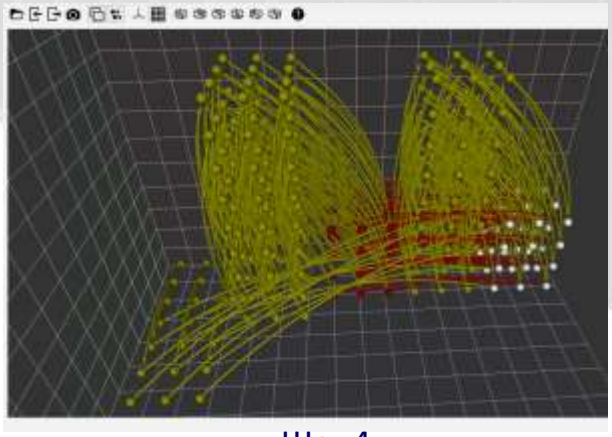
Шаг 1



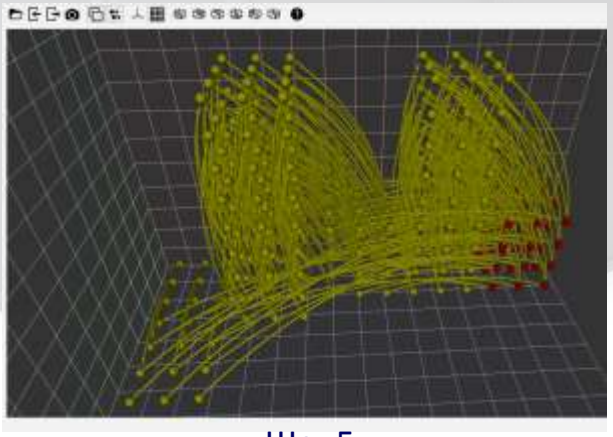
Шаг 2



Шаг 3



Шаг 4



Шаг 5

Цвета в ярусно-параллельной форме:

Красный – слой выполняющихся операций; **Зеленый** – уже выполненные операции;

Белый – операции должны выполниться позже.

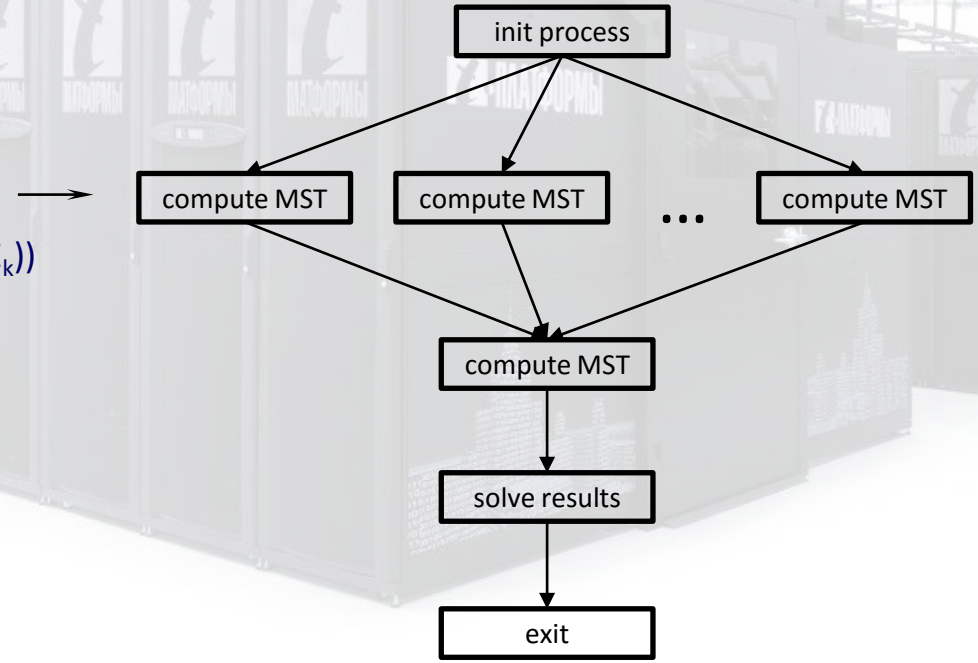
Потенциальный параллелизм: как находить, описывать, показывать… ?

(сложности описания алгоритмов)

$E = E_1 \cup E_2 \cup \ldots \cup E_k$

$MST(E) = MST(E_1 \cup E_2 \cup \ldots \cup E_k) =$

$= MST(\ MST(E_1) \cup MST(E_2) \cup \ldots \cup MST(E_k))$

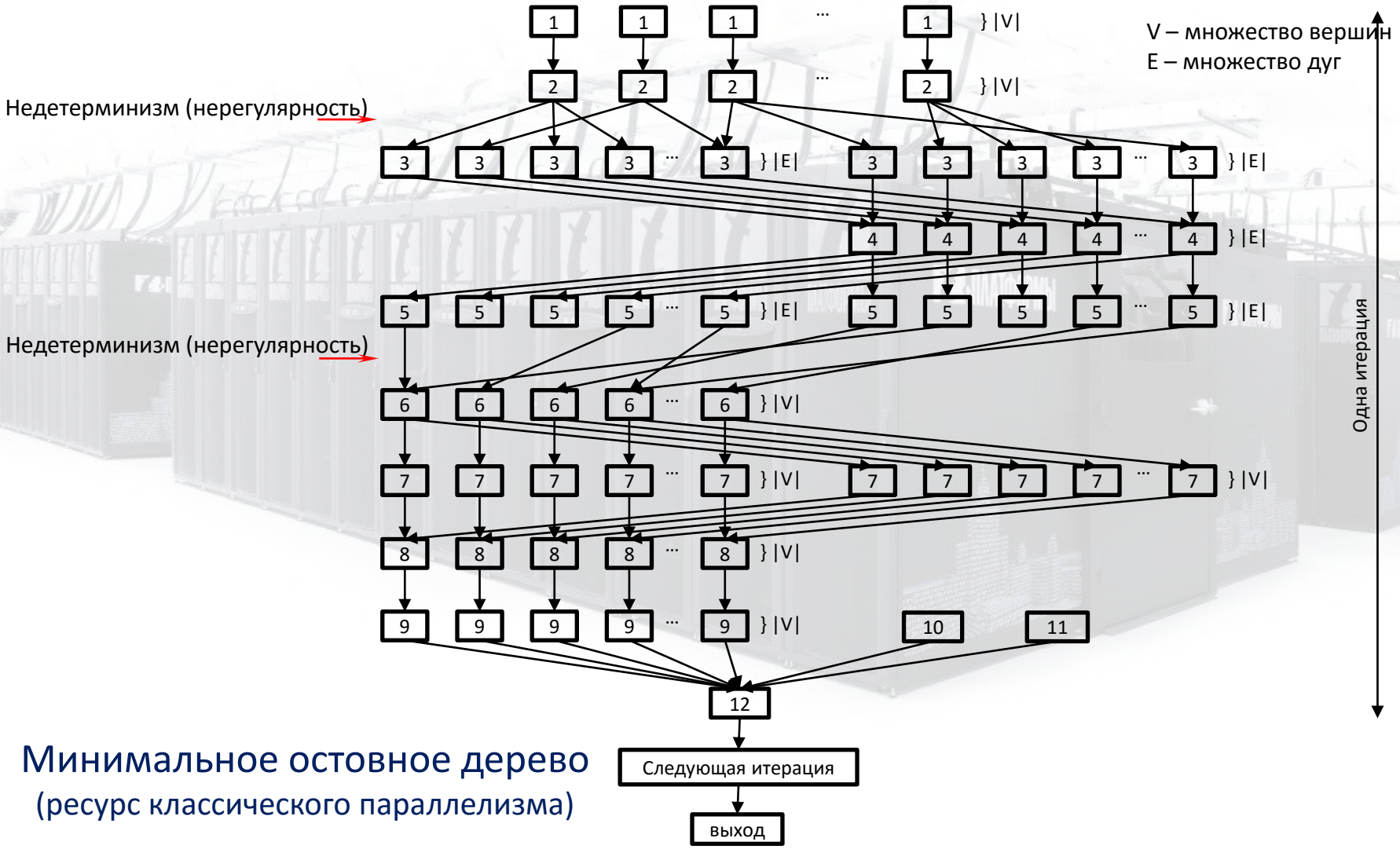


Минимальное остовное дерево (MST)

(ресурс “математического” параллелизма)

Потенциальный параллелизм: как находить, описывать, показывать… ?

(сложности описания алгоритмов)



Минимальное остовное дерево

(ресурс классического параллелизма)

Структура алгоритмов: возможные варианты для параллельного кода

(сложности описания алгоритмов)

Предположим, что мы знаем потенциальный параллелизм алгоритма...
Как выразить “потенциальную свободу” в выборе походящей формы для параллельной программы ?

Структура алгоритмов: возможные варианты для параллельного кода

(сложности описания алгоритмов)

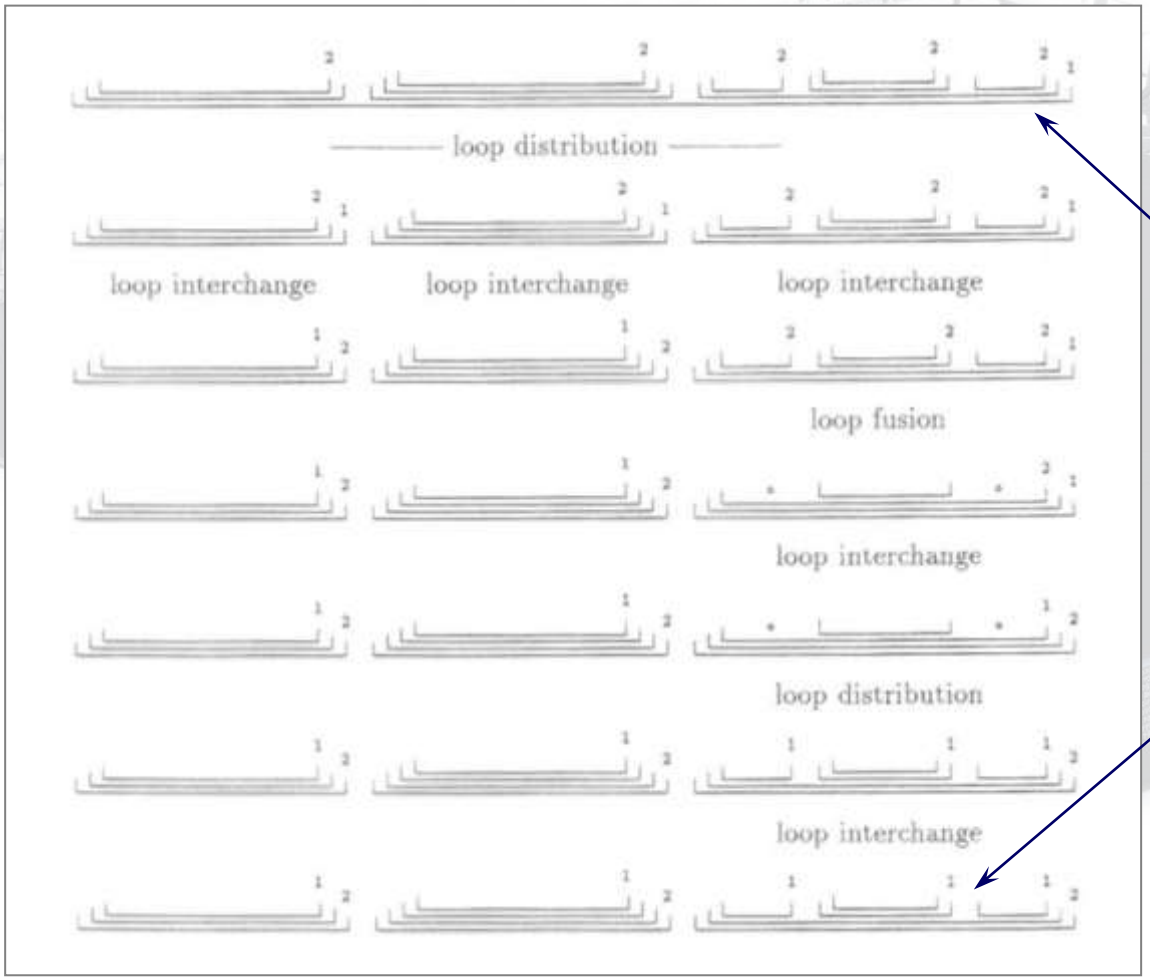
Циклический профиль алгоритма / программы

Самый внешний цикл (отмечен “1”) - параллельный.

Можно ли сделать данный цикл самым внутренним, чтобы векторизовать?

Структура алгоритмов: возможные варианты для параллельного кода

(сложности описания алгоритмов)



Эта последовательность преобразований делает исходный внешний цикл (1) внутренним и позволяет векторизовать программу (алгоритм).

Как показать подобный потенциал преобразований в общем случае?

Возможные источники несбалансированности: это важно, это нужно отметить

(сложности описания алгоритмов)

Баланс между арифметическими операциями +/- and * ;

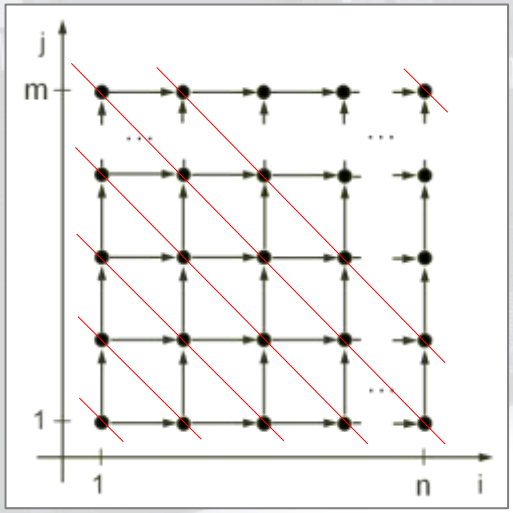
Баланс между арифметическими операциями и “чтениями/записями”;

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;

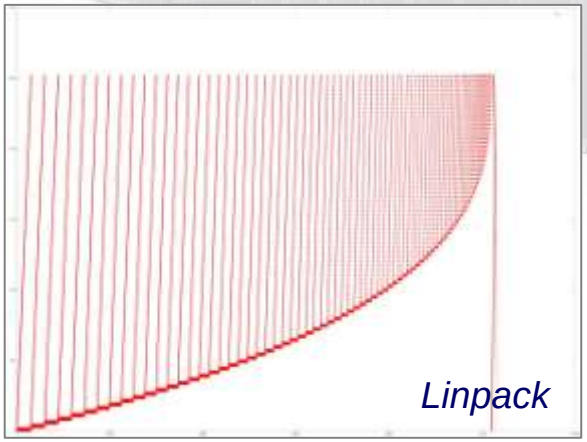
Датчики случайных чисел;

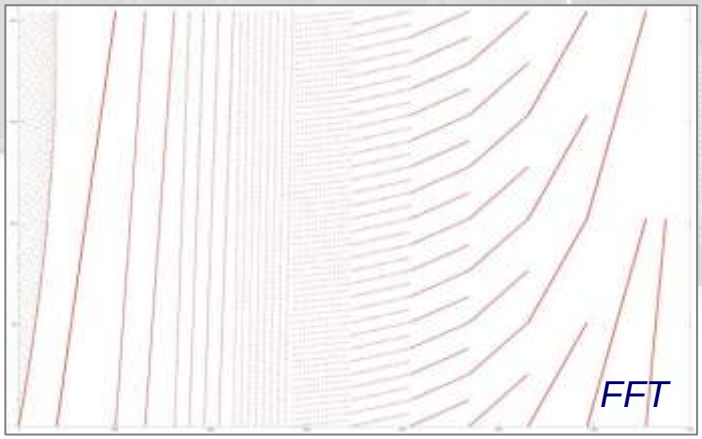
Отсутствие повторяемости результатов из-за разного порядка выполнения ассоциативных операций…

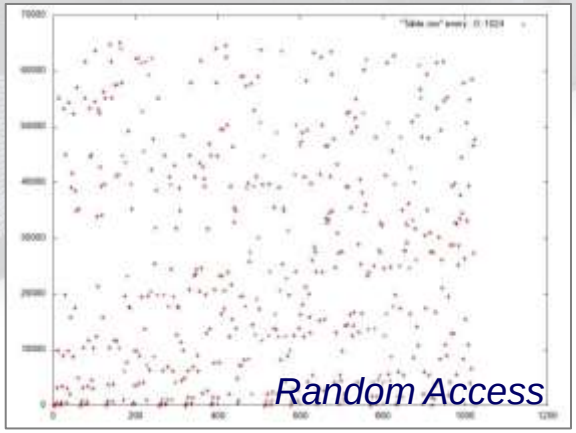
Локальность данных: множество открытых вопросов
(сложности описания алгоритмов)

Как оценивать пространственную и временную локальность данных в программе ?

Как сравнивать пространственную и временную локальность данных разных программ ?







Локальность данных: множество открытых вопросов
(сложности описания алгоритмов)

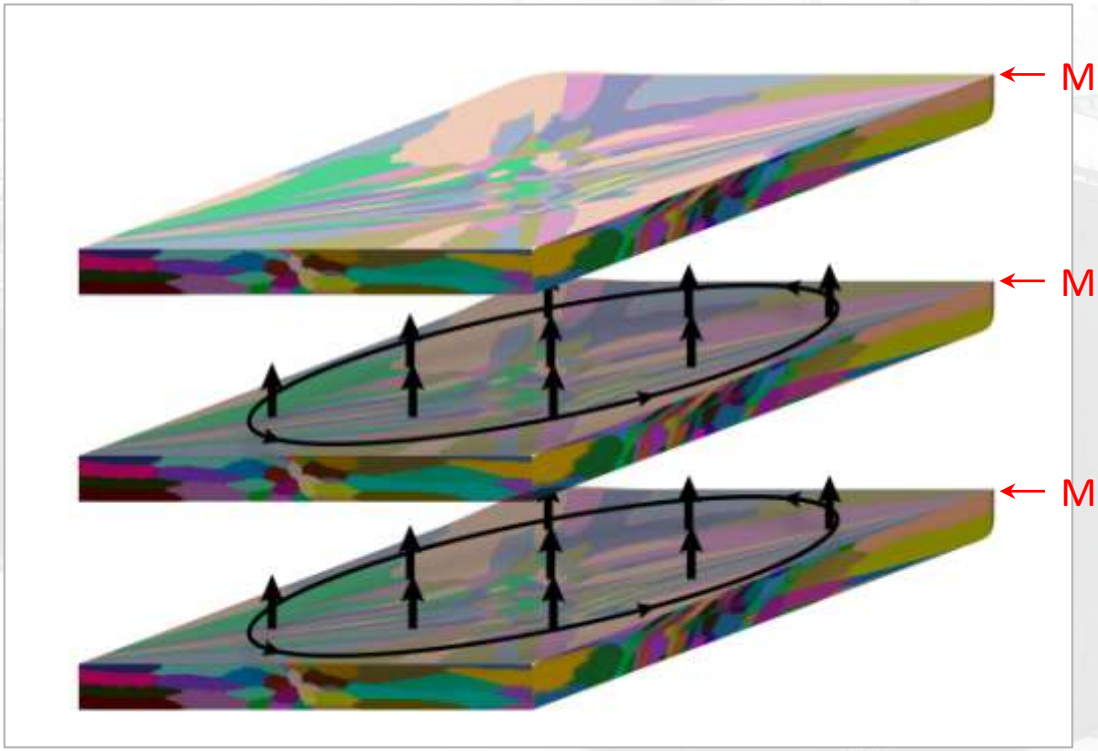
Как оценивать пространственную и временную локальность данных в программе ?

Как сравнивать пространственную и временную локальность данных разных программ ?

Можем ли мы предсказывать локальность данных в будущих реализациях, опираясь только на информацию об алгоритмах?

Что означает “локальность алгоритма” ?
Что означает “алгоритм обладает хорошей/плохой локальностью” ?
В алгоритмах нет структур данных, а значит и понятие “локальность” к ним напрямую не применимо! Вместе с этим, алгоритмы определяют суть программ, где локальность уместна и важна.

Коммуникационный профиль приложений
(сложности описания алгоритмов)



А что такое коммуникационный профиль приложений?
Как его получить и описывать? (Scalasca, Vampir…)

Все ли мы знаем о разложении Холецкого ?

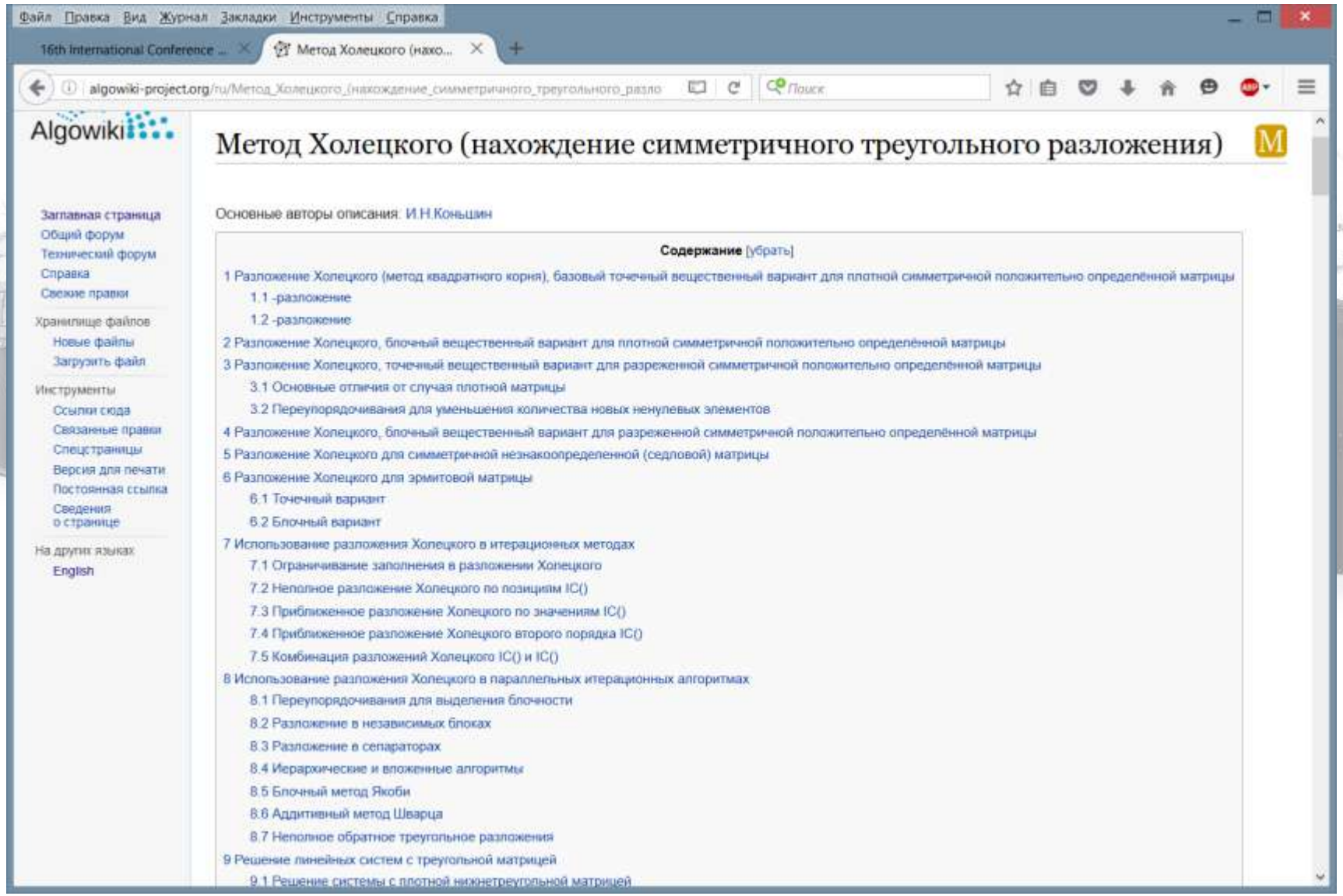
Да… Кажется, что Да…

Фундаментальный вопрос:

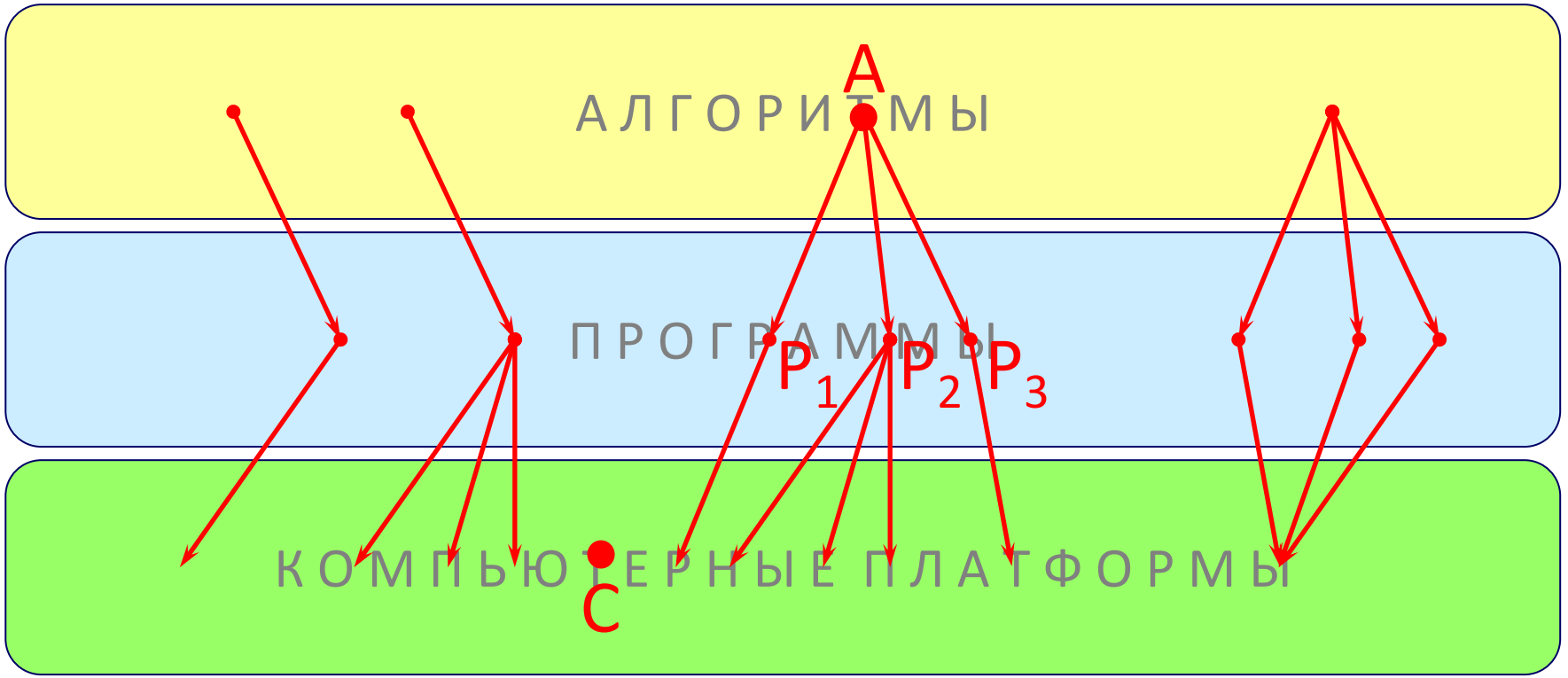
Что означает выполнить полное описание алгоритма?

Пока ответа не знает никто...

Разновидности одного алгоритма…
(сложности описания алгоритмов)



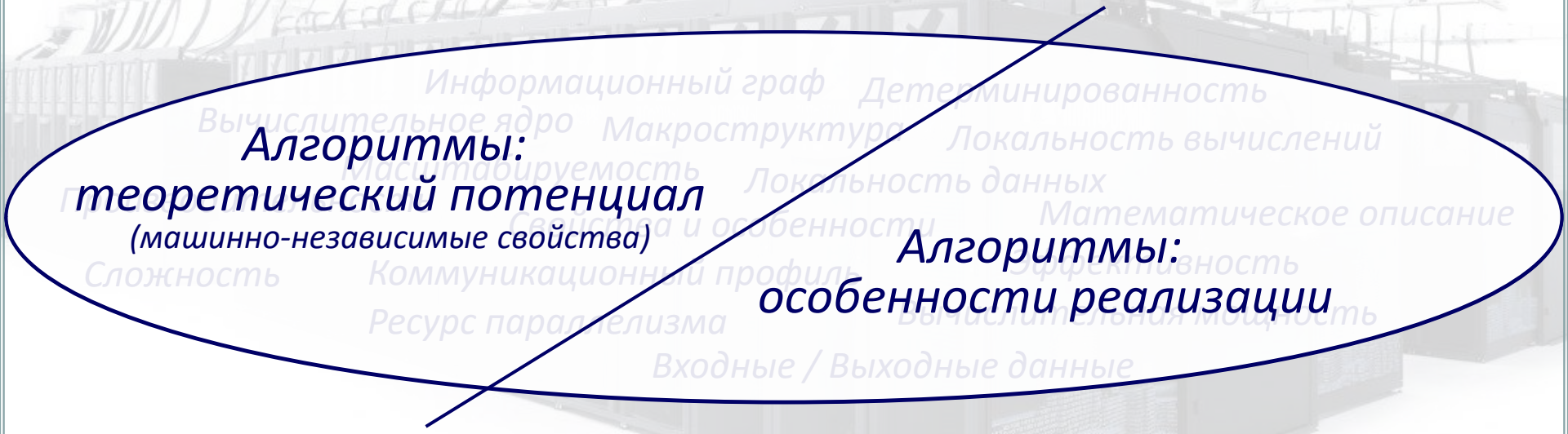
Алгоритмы и их реализации



Достаточно ли у нас информации об алгоритме A, чтобы создать эффективную реализацию для платформы C?

Можем ли мы использовать реализации P_1, P_2, P_3 или мы должны создавать еще одну реализацию P для платформы C?

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)

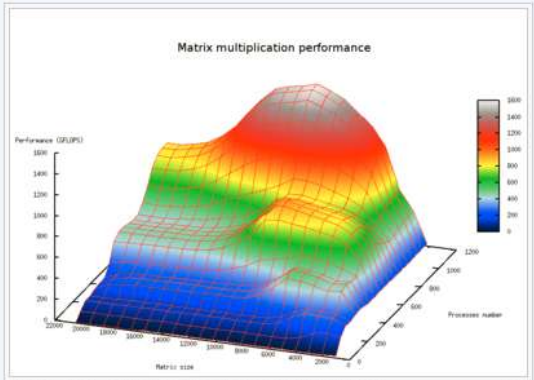
1 Свойства и структура алгоритма

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

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** для различных программно-аппаратных платформах от мобильных платформ до суперкомпьютерных систем с возможностью коллективной работы всего мирового сообщества.

Цель **AlgoWiki** - дать исчерпывающее описание алгоритма, которое позволяет 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, локальности данных, эффективность и масштабируемость, коммуникационный профиль параллельных реализаций и многое другое.

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)

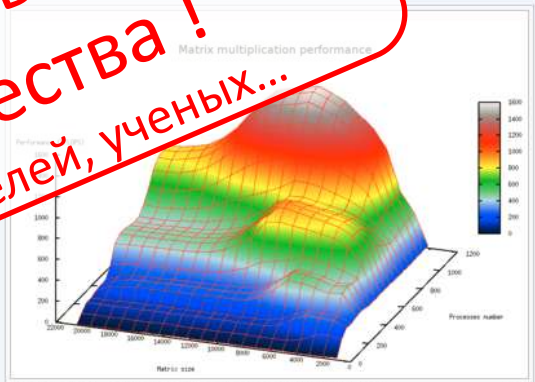
1 Свойства и структура алгоритма

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

AlgoWiki – это проект для всего вычислительного сообщества !
Для студентов, аспирантов, преподавателей, ученых...

CALL FOR PAPERS

"Supercomputing Co-Design Technology"

(http://agora.guru.ru/SCDT)

As part of **ICA3PP 2016**
16th International Conference on Algorithms and Architectures for Parallel Processing
(http://www.arcos.inf.uc3m.es/wp/ica3pp2016)

14–15th DECEMBER 2016, GRANADA, SPAIN

SCOPE AN

The computing world is changing rapidly. All de
high-performance supercomputers – are b

IMPORTANT DATES

Paper Submission Deadline: July 25th, 2016
Author Notification: September 15th, 2016
Camera-Ready Paper Due: October 15th, 2016
Workshop Dates: December 14th–15th, 2016

PROGRAM COMMITTEE

- **Victor Gergel, co-chair**, Lobachevsky State University of Nizhni Novgorod, Russia
- **Vladimir Voevodin, co-chair**, Moscow State University, Russia
- Arndt Bode, Leibniz Supercomputing Centre, Germany
- Alexander Boukhanovsky, ITMO University, Russia
- Yuefan Deng, Stony Brook University, USA
- Florent de Dinechin, INSA Lyon, France
- Torsten Hoefler, Swiss Federal Institute of Technology, Switzerland
- Thomas Ludwig, German Climate Computing Center, Germany
- Iosif Meyerov, Lobachevsky State University of Nizhni Novgorod, Russia
- Marek Michalewicz, A*STAR Computational Resource Centre, Singapore
- Bernd Mohr, Jülich Supercomputing Centre, Germany
- Mikhail Moshkov, King Abdullah University of Science and Technology, Saudi Arabia
- Sergey Orlov, Transport and Telecommunication Institute, Latvia
- Nina Popova, Moscow State University, Russia
- Arnold Rosenberg, Northeastern University, USA
- Ahmed Seffah, Lappeenranta University of Technology, Finland
- Yaroslav Sergeyev, University of Calabria, Italy
- Andrey Sozykin, Ural Federal University, Russia
- Roman Wyrzykowski, Czestochowa University of Technology, Poland

PUBLICATIONS

All papers selected by the Program Committee and presented at the Workshop will be published in **the Workshop Proceedings by Springer LNCS series**. After the conference, the authors of the papers presented at the workshop will be invited to submit extended versions of the papers in the following scientific journal: Supercomputing Frontiers and Innovations (http://superfri.org/superfri).

International Conference

Russian Supercomputing Days

Москва
26-27 сентября 2016 г.

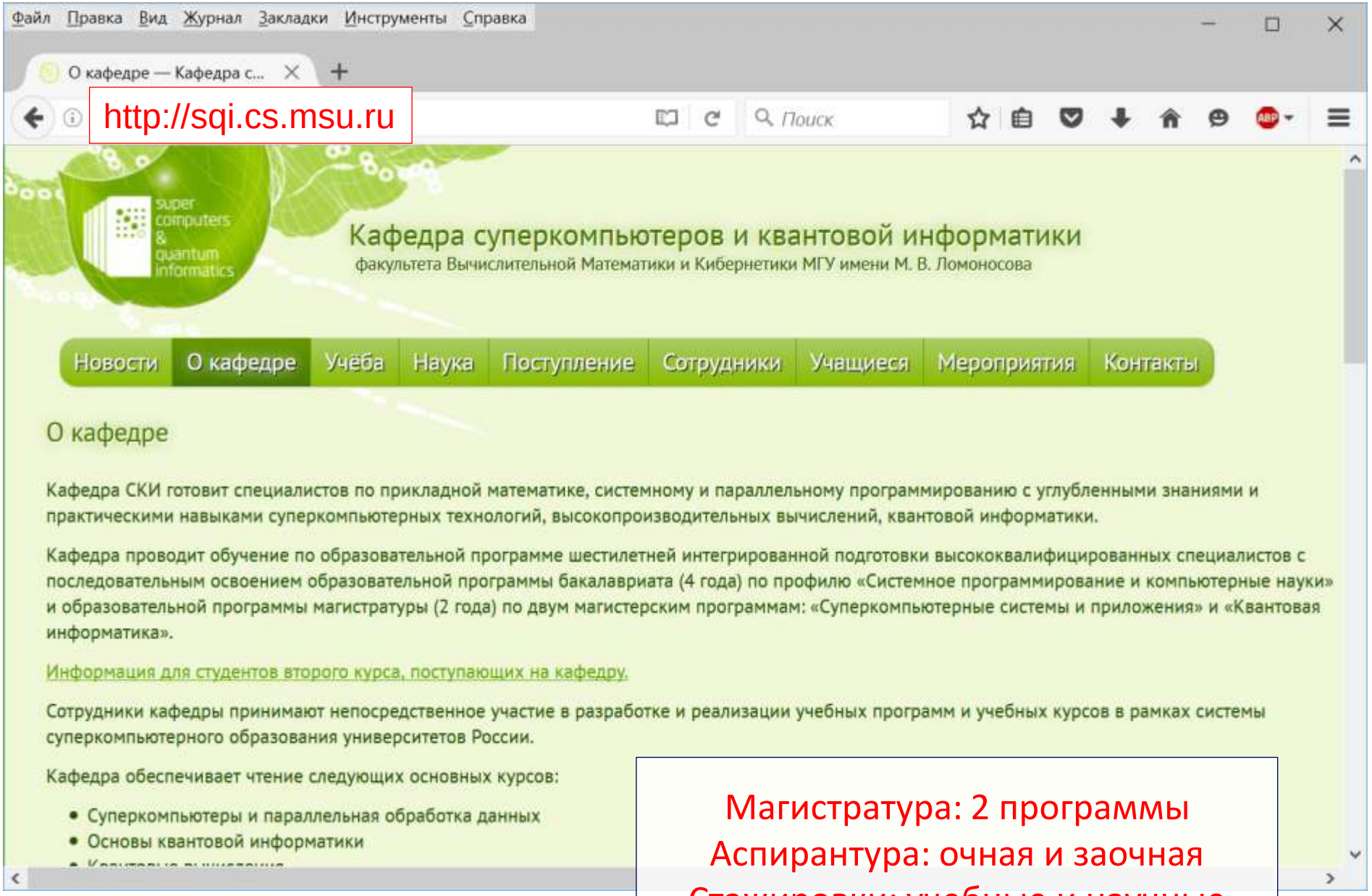
«Суперкомпьютерные дни в России» — новая серия ежегодных международных суперкомпьютерных конференций, в основу которой легли давние традиции российских и международных суперкомпьютерных мероприятий.

RussianSCDays.org

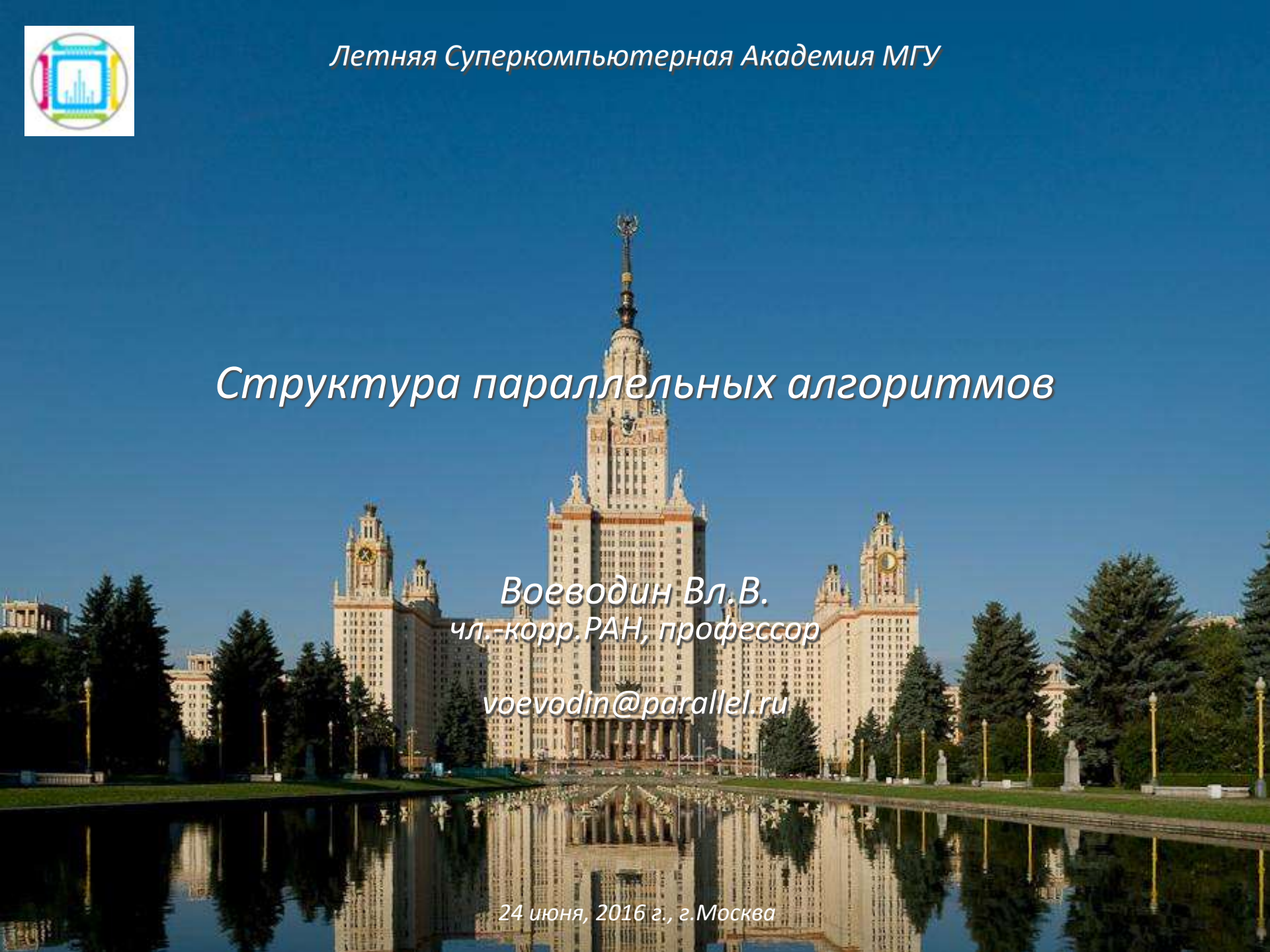
- 1 апреля 2016 г. - начало приема докладов
- 1 июня - представление аннотаций докладов
- 15 июня - представление полных версий докладов
- 15 июля - уведомление о включении доклада в программу конференции
- 30 июля - представление окончательного варианта статьи

СУПЕРКОМПЬЮТЕРНЫЕ ДНИ В РОССИИ 2015

- Два исключительно наполненных суперкомпьютерными событиями дня
- 450 участников из России, Европы, Китая, Японии, США и других стран
- Представители 35 университетов, 28 институтов РАН, 60 коммерческих компаний
- Пленарные доклады и 7 параллельных секций
- Стендовая секция
- Суперкомпьютерная выставка
- Конференция молодых ученых
- Тренинги от ведущих HPC-компаний: Intel, Mellanox, NVIDIA, HP, PCK ...
- Анонс рейтинга Top50, индустриальные секции, тренинги для школьных учителей информатики, дискуссионные группы, 3D-визуализация и т.п.



Магистратура: 2 программы
Аспирантура: очная и заочная
Стажировки: учебные и научные



Летняя Суперкомпьютерная Академия МГУ

Структура параллельных алгоритмов

Воеводин Вл.В.
чл.-корр.РАН, профессор

voevodin@parallel.ru

24 июня, 2016 г., г.Москва