IX Международная молодежная научно-практическая школа «Высокопроизводительные вычисления на Grid системах»
(Архангельск, 5 - 10 февраля 2018 г.)

Модели параллельных вычислений

д.ф.-м.н., профессор Л.Б. Соколинский

Южно-Уральский государственный университет
(национальный исследовательский университет)

Содержание

- Тенденции в области высокопроизводительных вычислений
- Изменение основной парадигмы дизайна алгоритма
- Масштабируемость, ускорение, параллельная эффективность
- Модель параллельных вычислений: определение, состав, требования
- Три популярные модели: PRAM, BSP, LogP
- Дерево моделей параллельных вычислений
- Новая модель параллельных вычислений BSF

www.top500.org – 500 самых мощных суперкомпьютеров мира

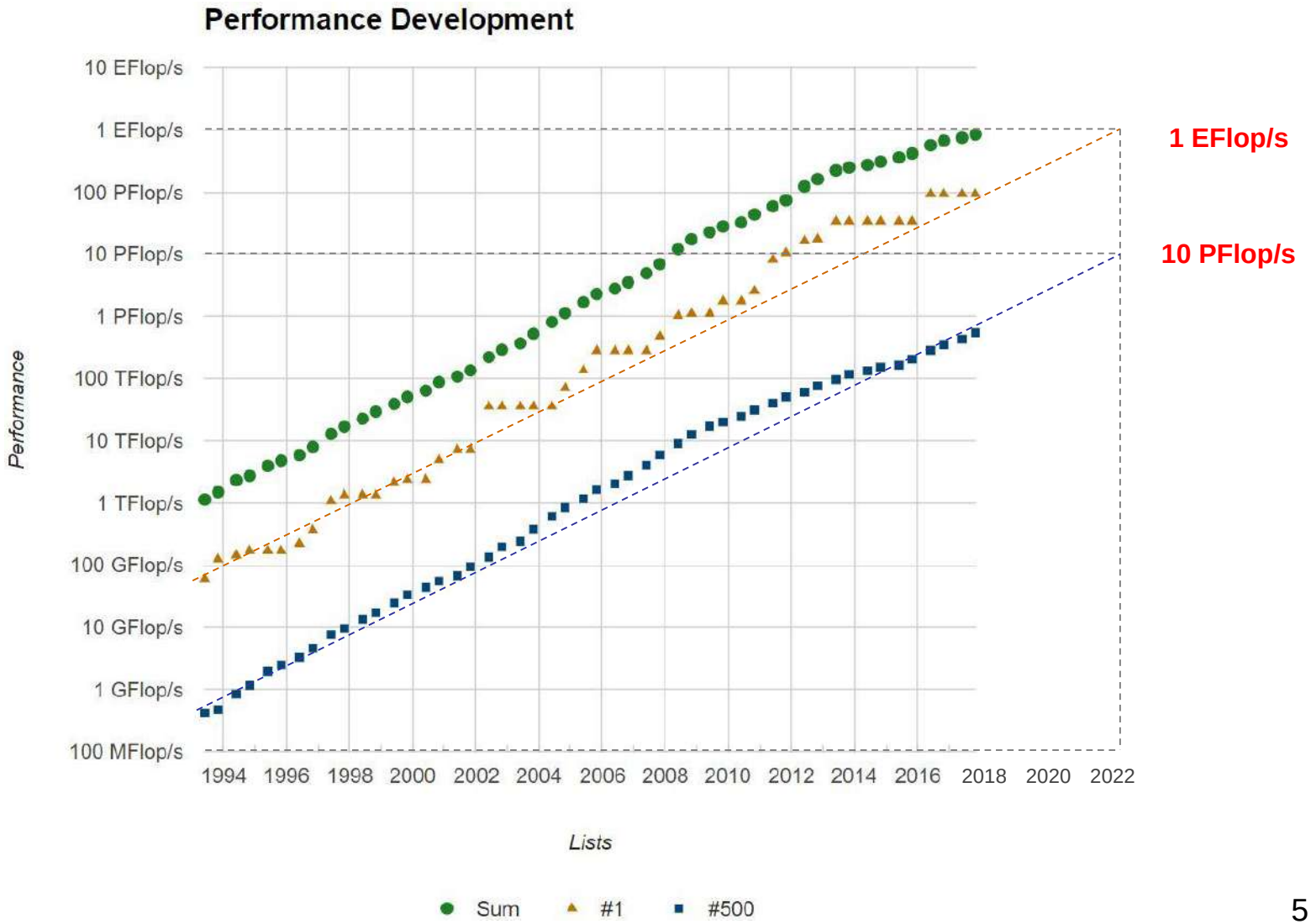
№1 в TOP500

- Ядер: **10 649 600**
- Память: **1 310 720 GB**
- Linpack: **93 PFlop/s**
- Узлов: **40 960**

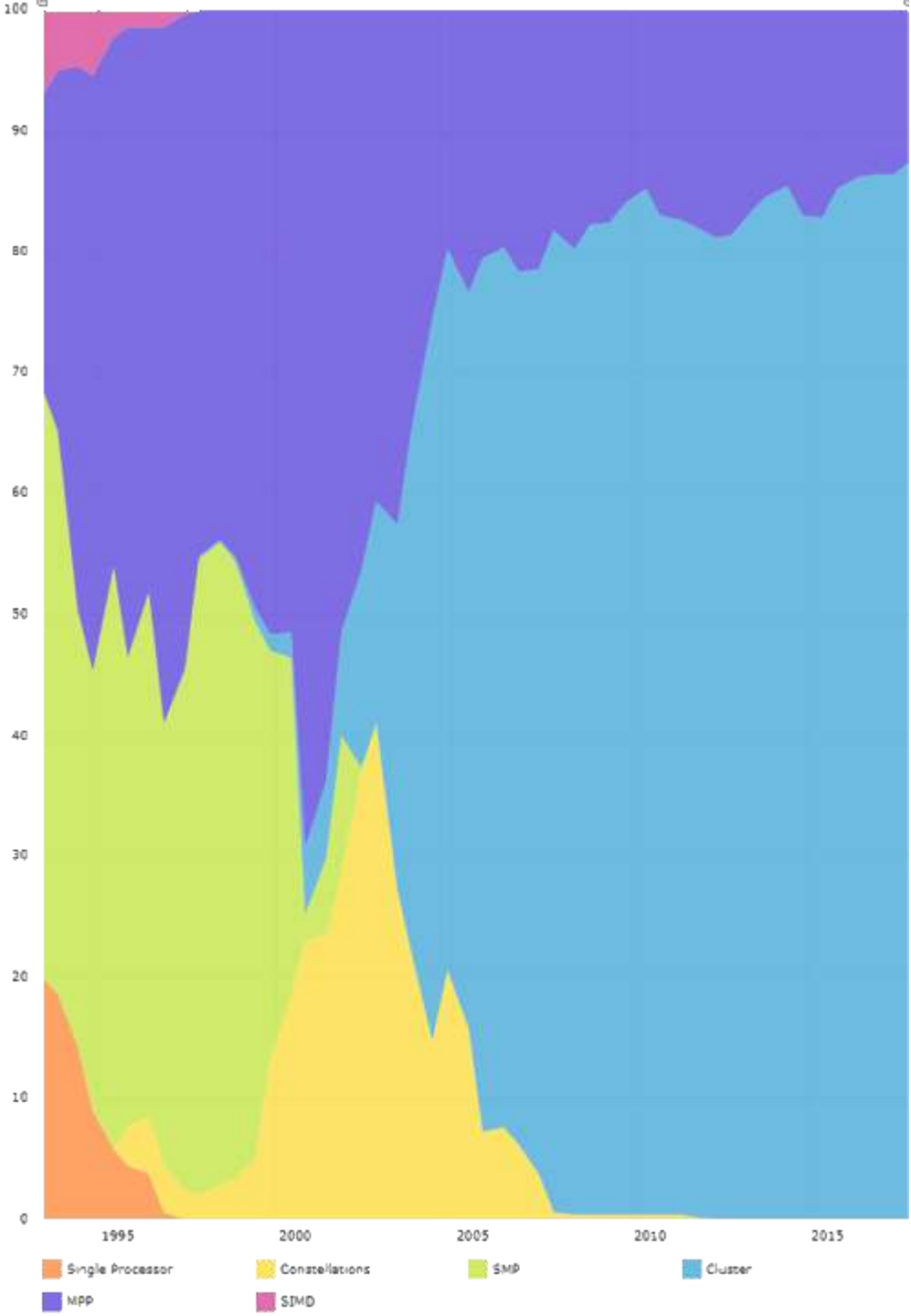
Sunway TaihuLight Supercomputer

National Supercomputing Center in Wuxi, China

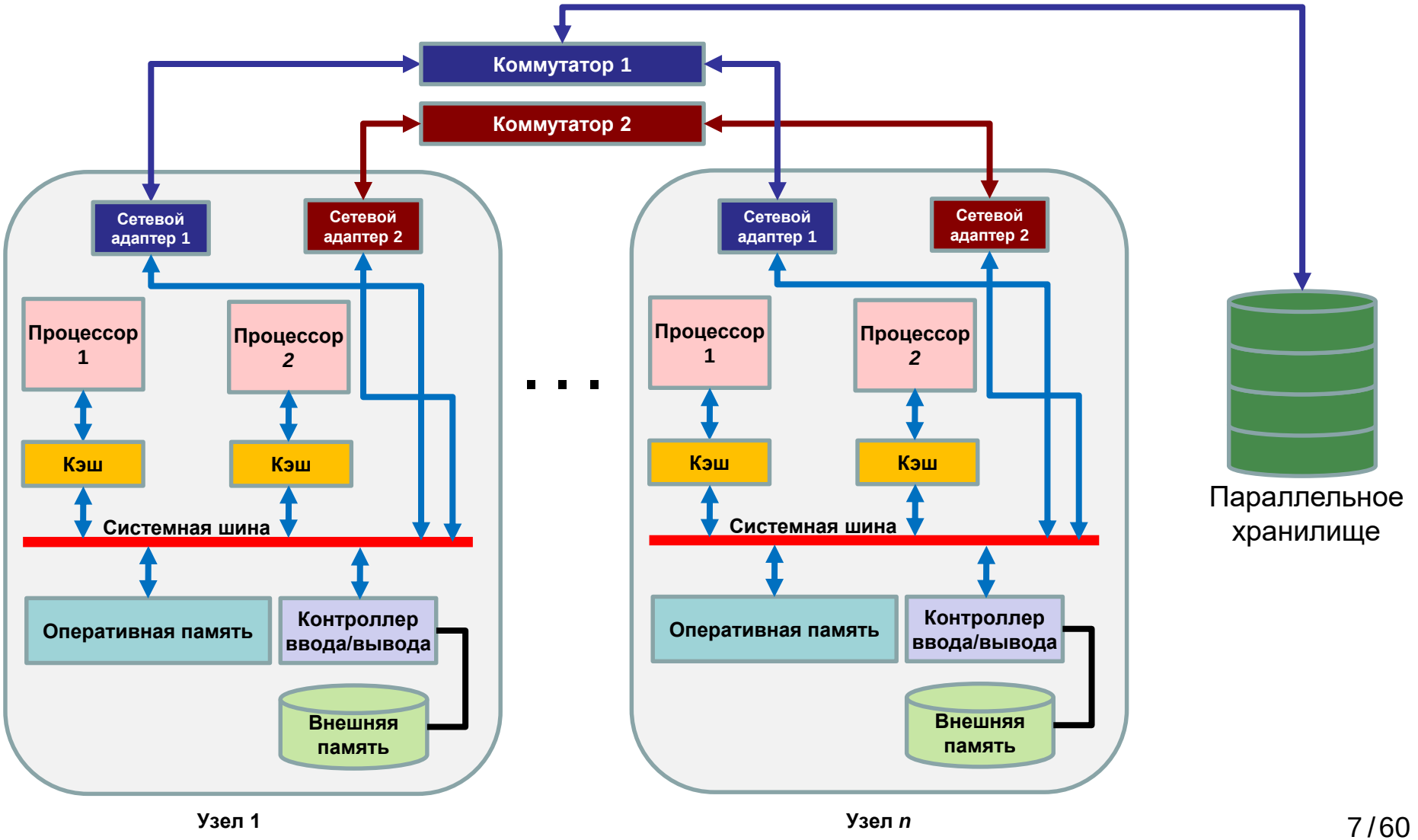
TOP500 (ноябрь 2017)



Архитектура суперкомпьютеров в TOP500

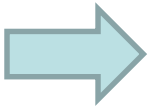


Кластерная (cluster) архитектура



Изменение основной парадигмы дизайна алгоритма

Разработать алгоритм, эффективно работающий на малых вычислительных ресурсах

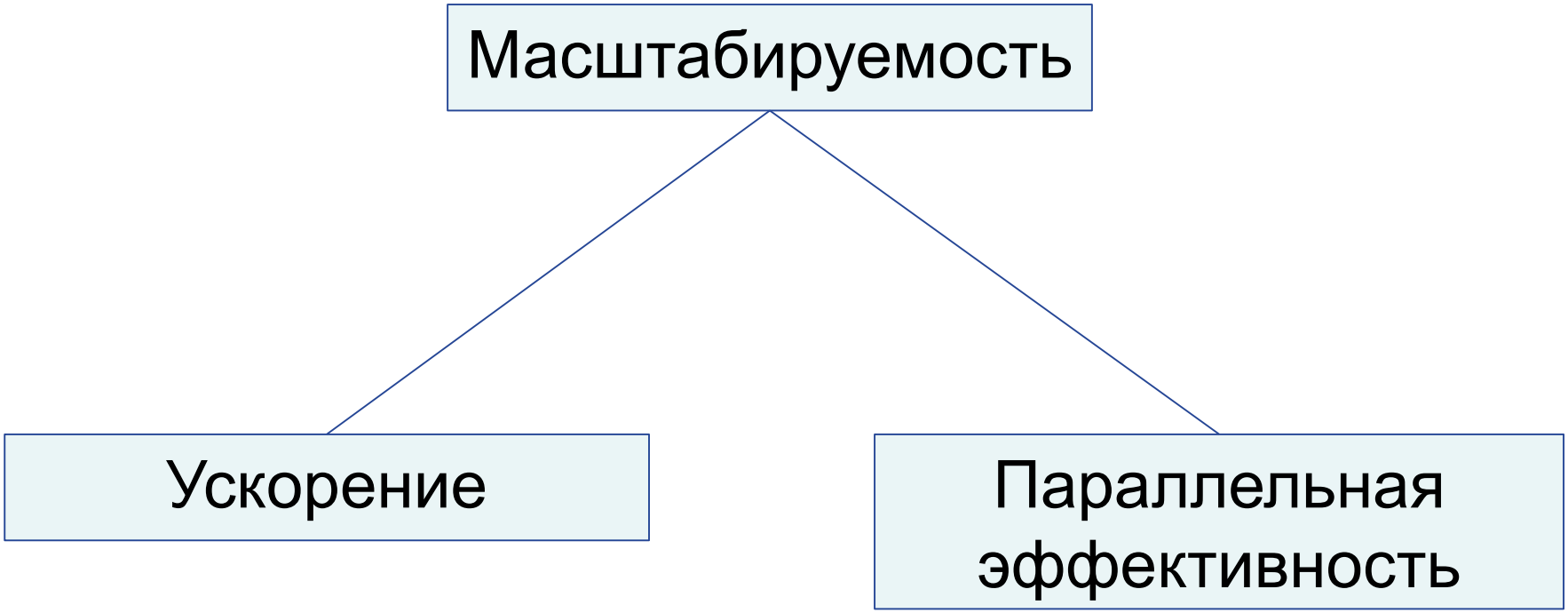


Разработать алгоритм, способный эффективно загрузить большие вычислительные ресурсы

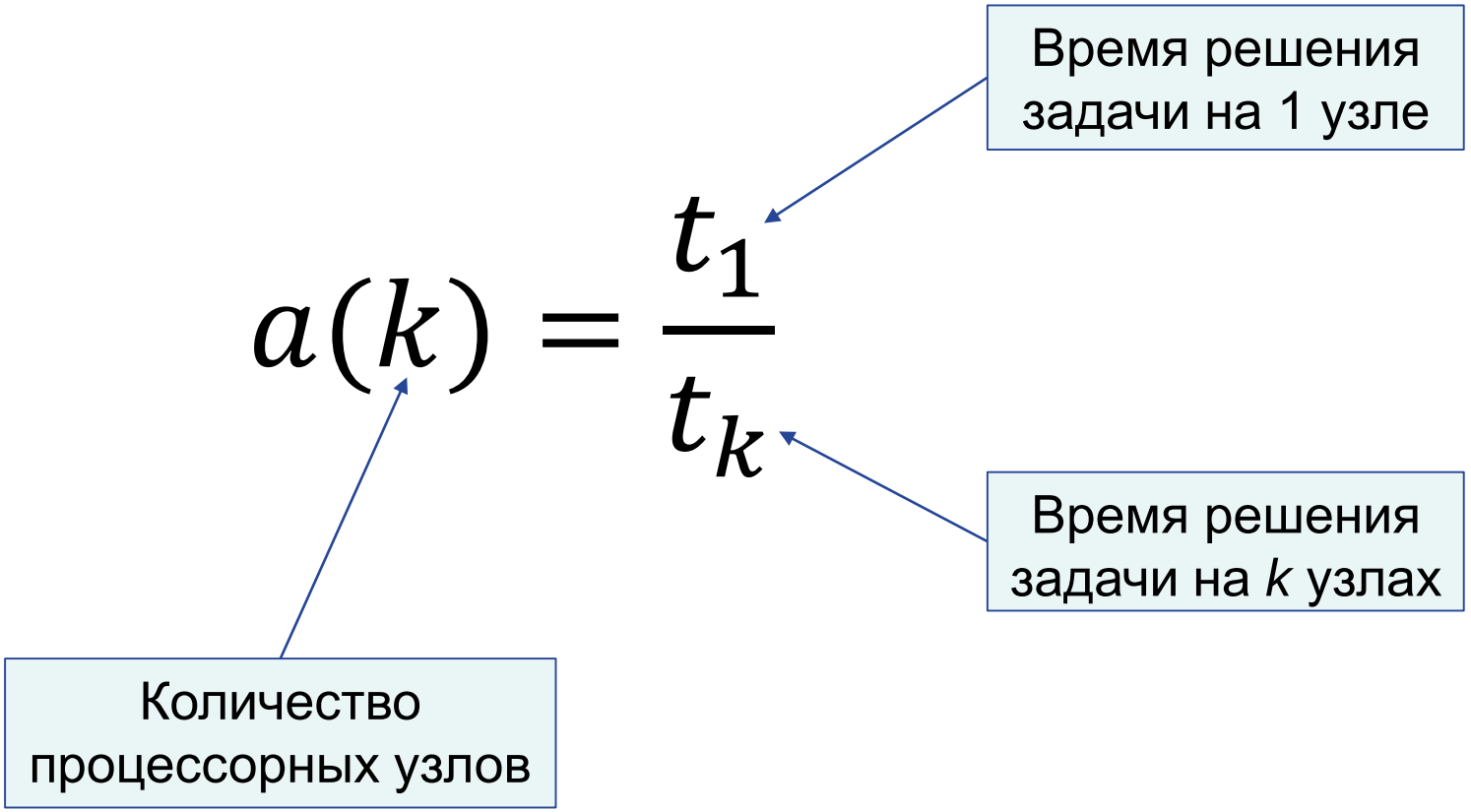
1970

2018

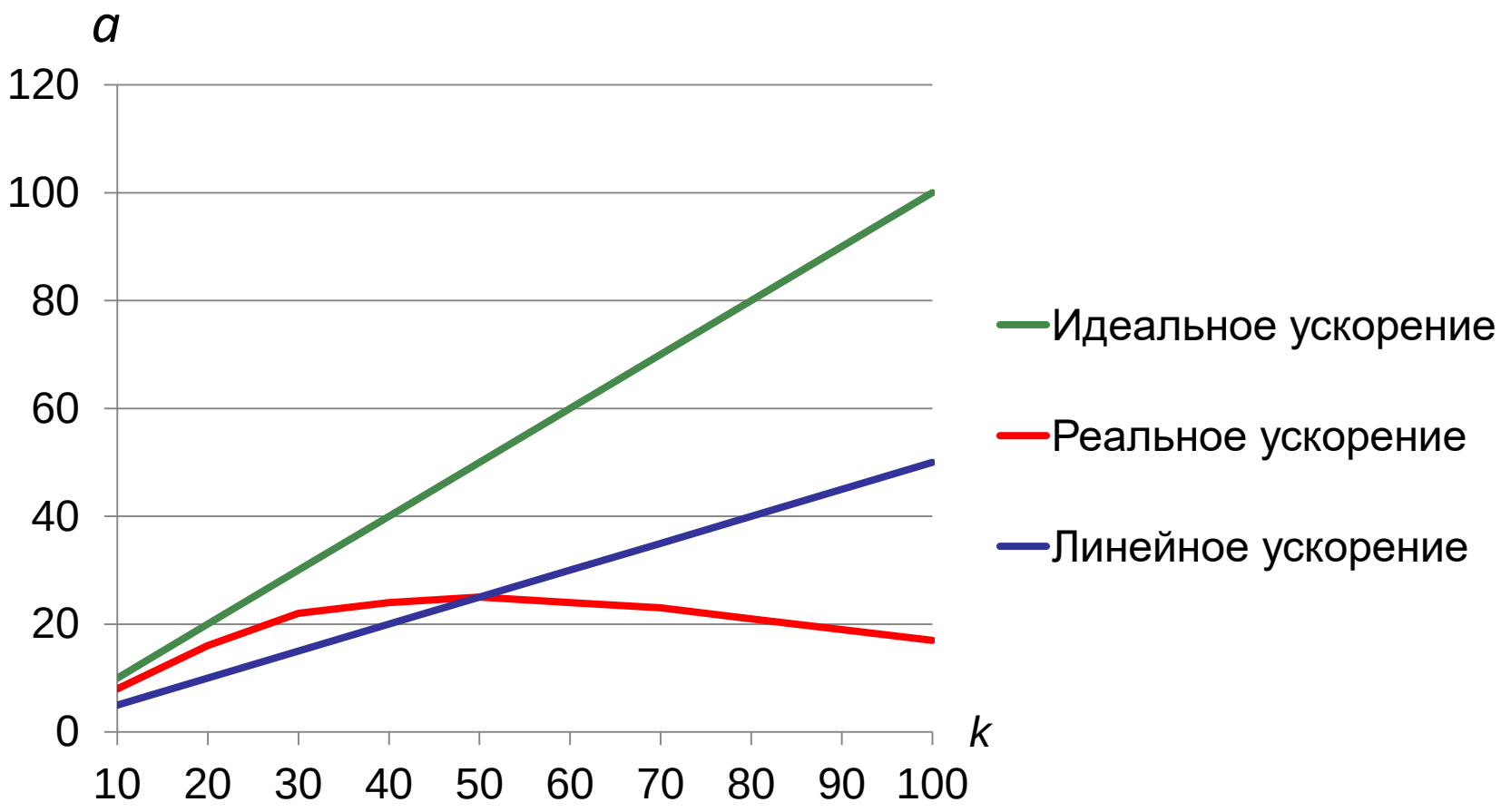
Главная характеристика современного параллельного численного алгоритма



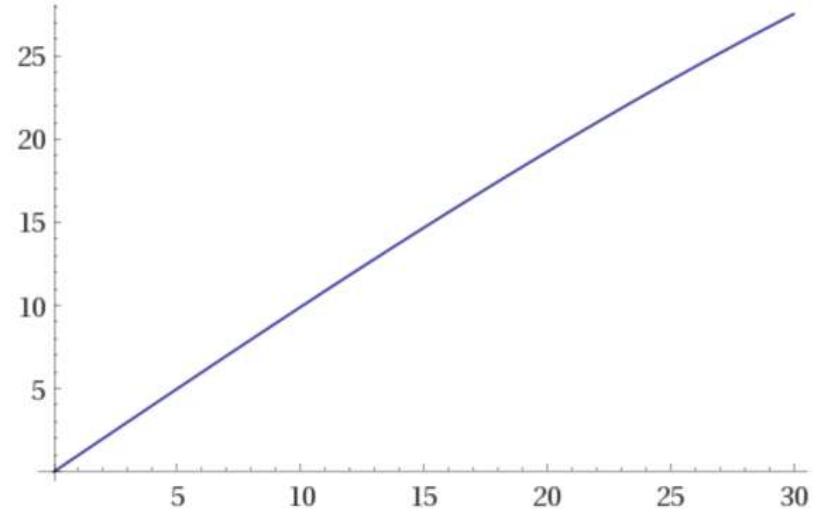
Ускорение

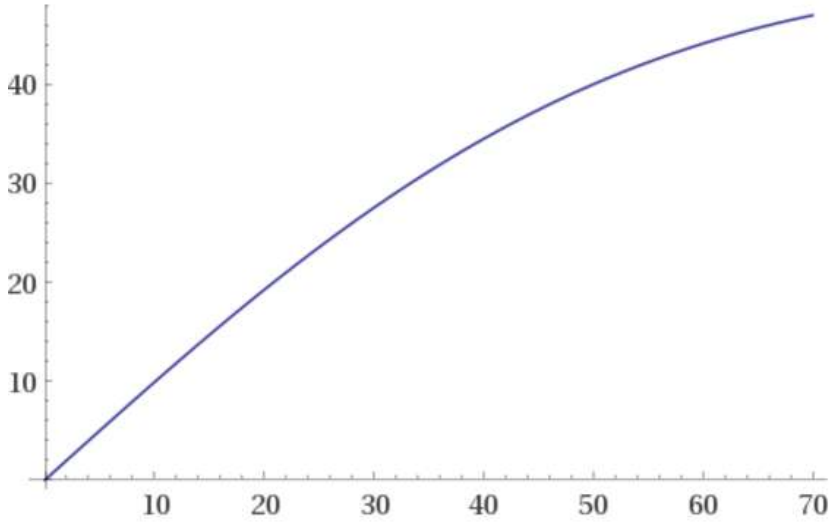

Время решения задачи на 1 узле
$a(k) = \frac{t_1}{t_k}$
Время решения задачи на *k* узлах
Количество процессорных узлов

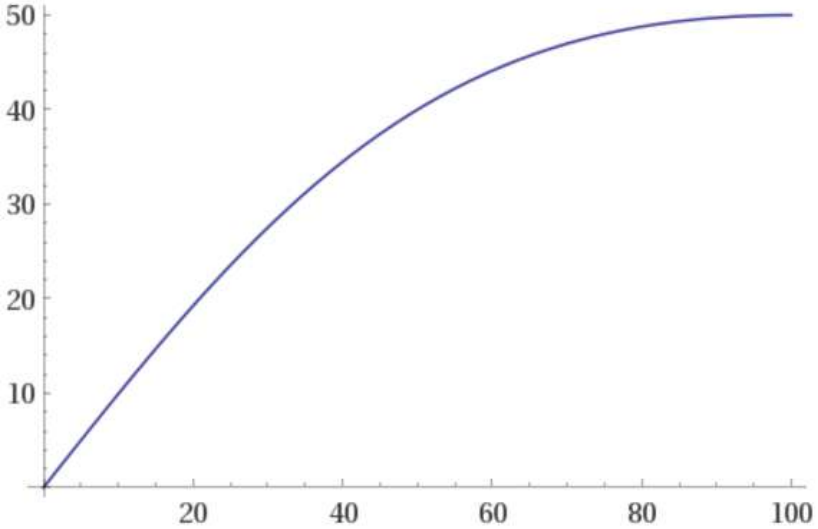
Ускорение: идеальное, линейное, реальное

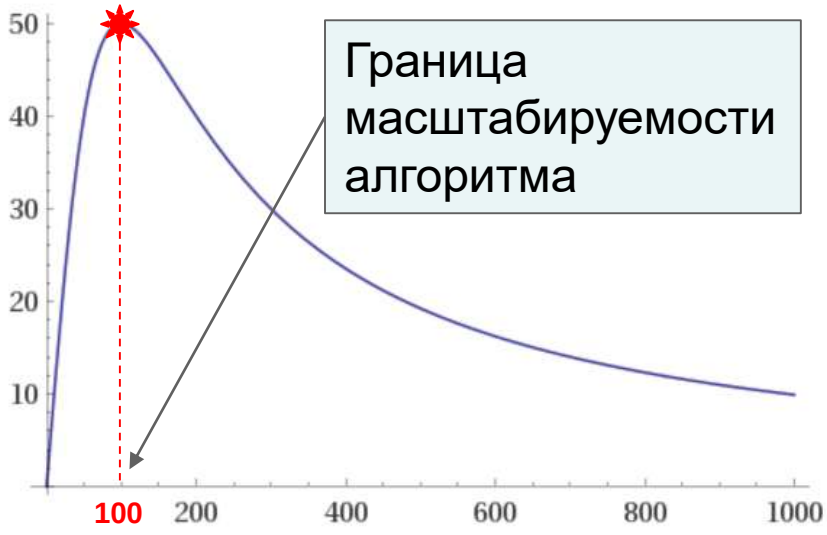


Ускорение реальной задачи на кластерной вычислительной системе

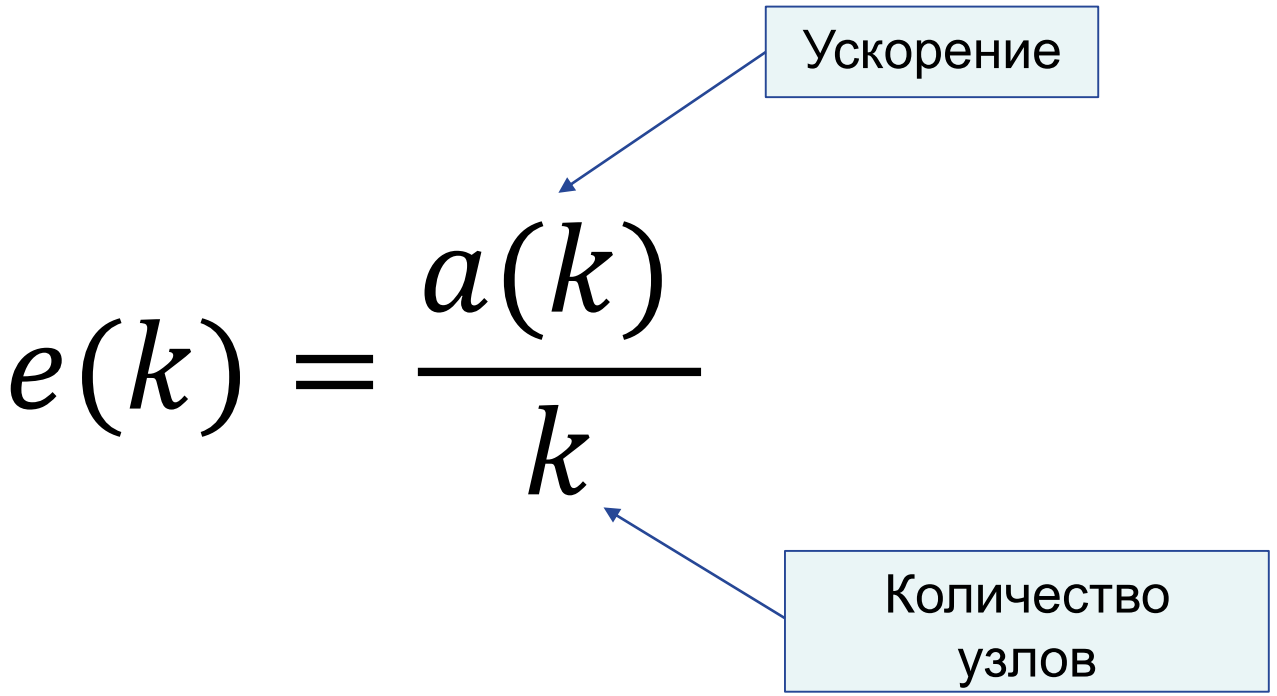




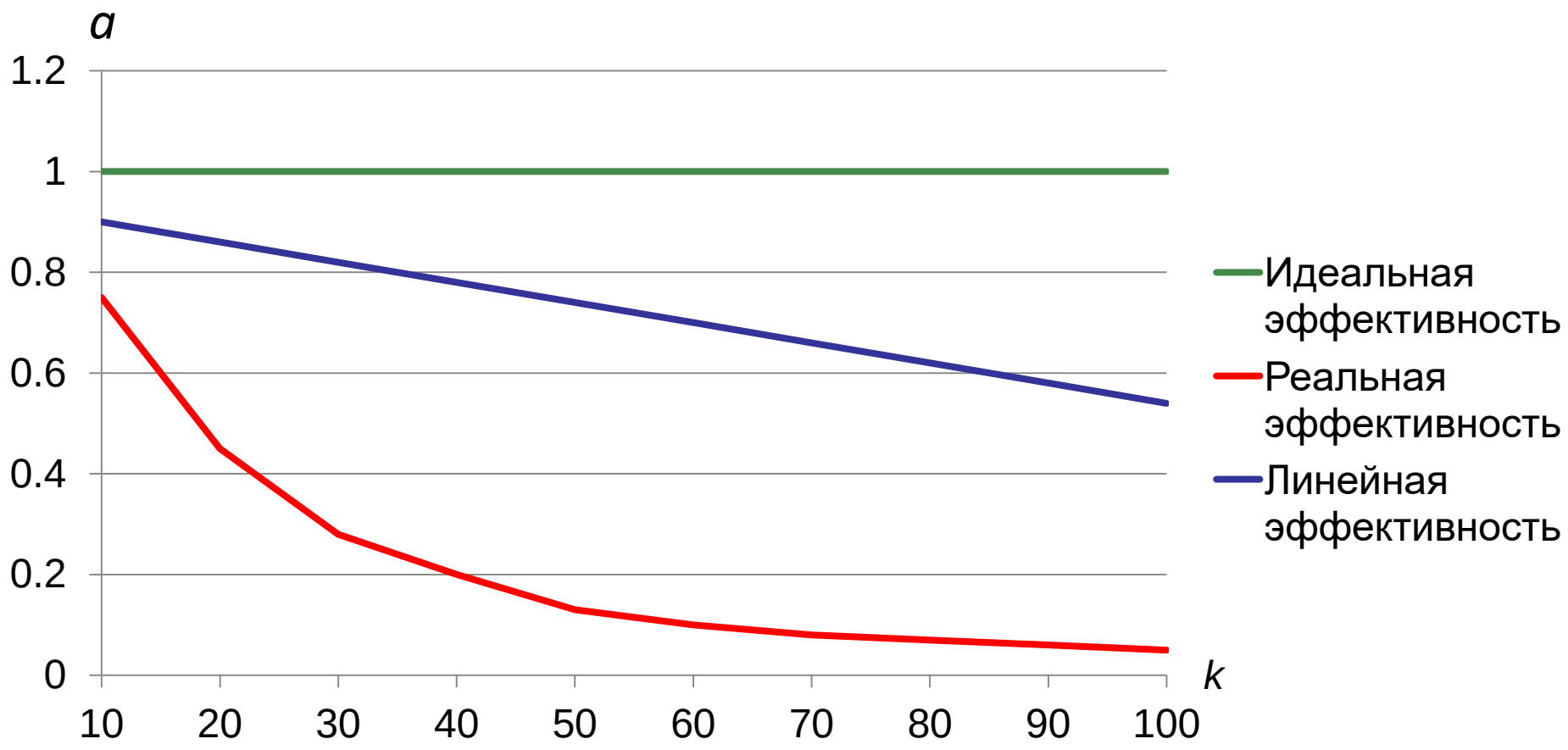




Параллельная эффективность

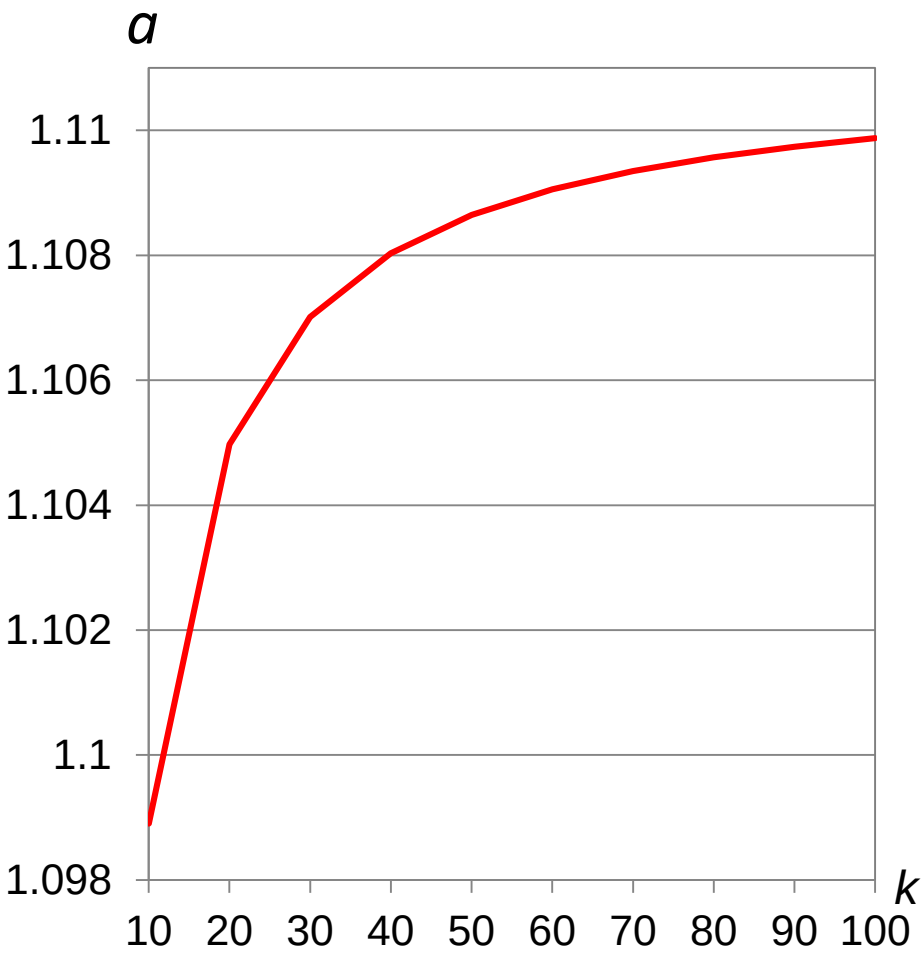


Параллельная эффективность: идеальная, линейная, реальная

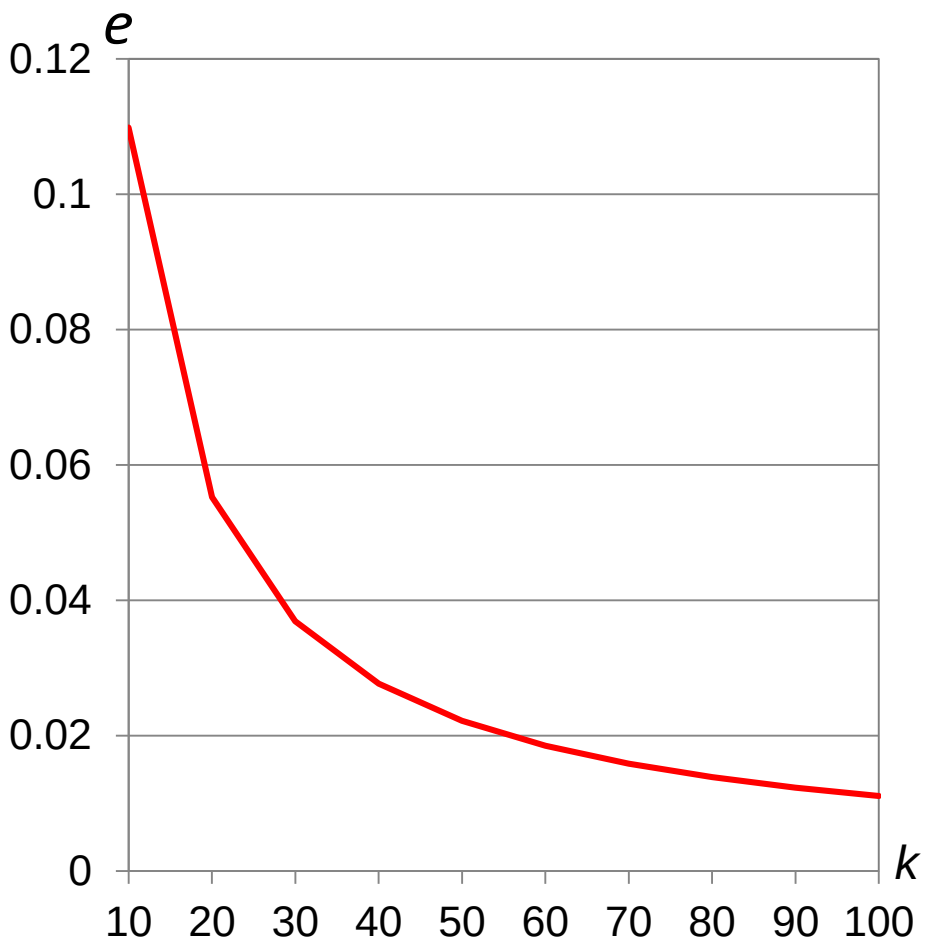


10% параллельного кода

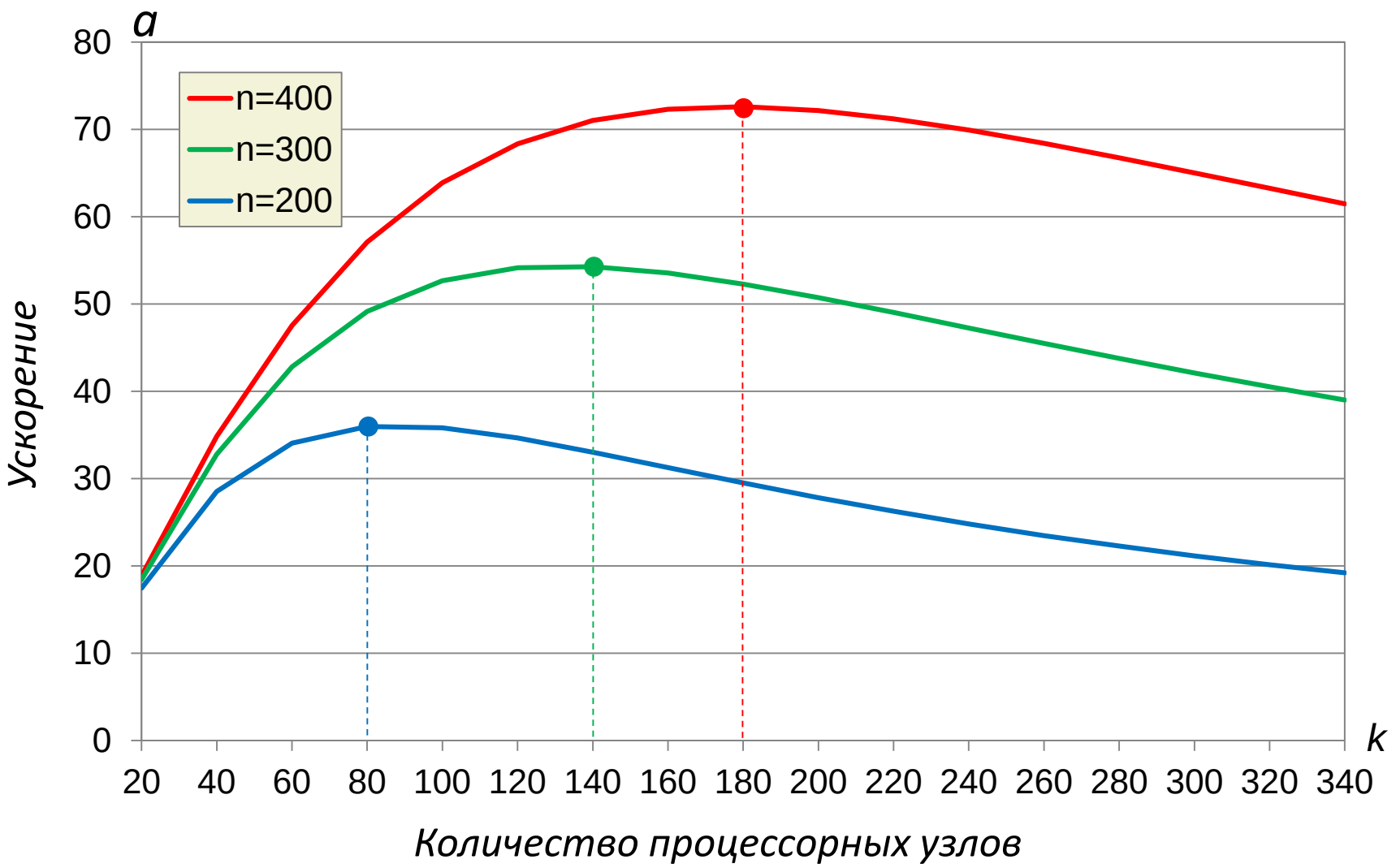
Ускорение



Параллельная эффективность



Размер задачи имеет значение!



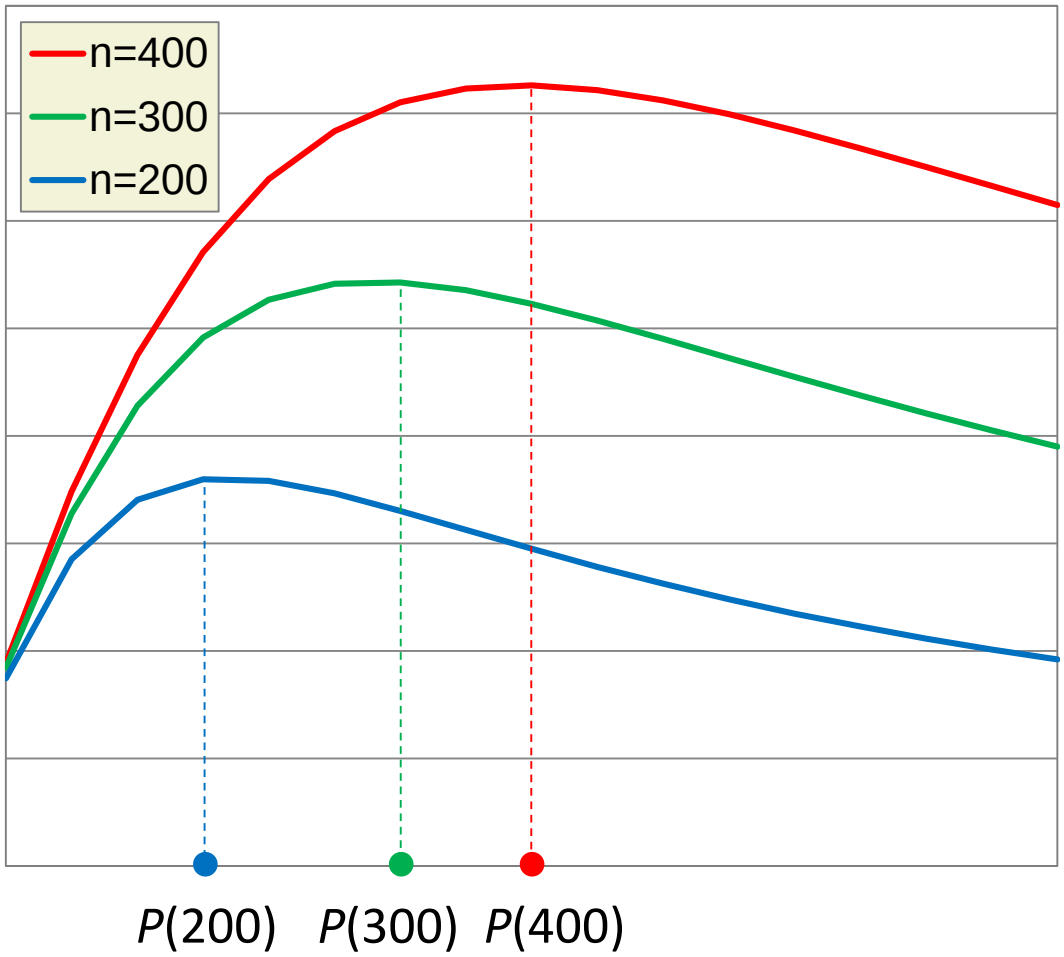
Хорошо масштабируемый алгоритм

Алгоритм является хорошо масштабируемым, если:

1) ускорение близко к линейному
2) параллельная эффективность > 0.5

Необходима формула для оценка масштабируемости алгоритма **до** написания программы

$$k = P(n)$$



Модель параллельных вычислений

- *Модель параллельных вычислений* – это фреймворк (система правил и ограничений) для описания и анализа параллельных алгоритмов и программ
- Модель параллельных вычислений позволяет оценить время выполнения параллельной программы (без ее реального выполнения на компьютере)

Модель параллельных вычислений включает в себя:

- ***Архитектурный компонент*** - помеченный граф, узлы которого соответствуют модулям с различной функциональностью, а дуги – межмодульным соединениям для передачи данных
- ***Спецификационный компонент*** - определяет что есть (синтаксически) корректный алгоритм/программа
- ***Компонент выполнения*** - задает последовательность состояний архитектурных модулей, обеспечивающих корректное выполнение алгоритма/программы для определенных входных данных
- ***Распараллеливающий компонент*** - определяет способы распределения вычислений, синхронизации и обмена данными между процессорными узлами, работающими независимо (параллельно)
- ***Стоимостный компонент*** - определяет одну или несколько метрик, позволяющих предсказать параметры выполнения алгоритма/программы (время выполнения, объем необходимой оперативной памяти и др.)

Требования к модели параллельных вычислений

- ***Юзабилити*** – легкость описания алгоритма и анализа его стоимости средствами модели (модель должна быть легкой в использовании)
- ***Переносимость*** – широта класса целевых платформ, для которых модель оказывается применимой
- ***Предиктивность*** – возможность с помощью стоимостных метрик модели предсказать *временные характеристики* выполнения алгоритмов/программ на целевой вычислительной системе

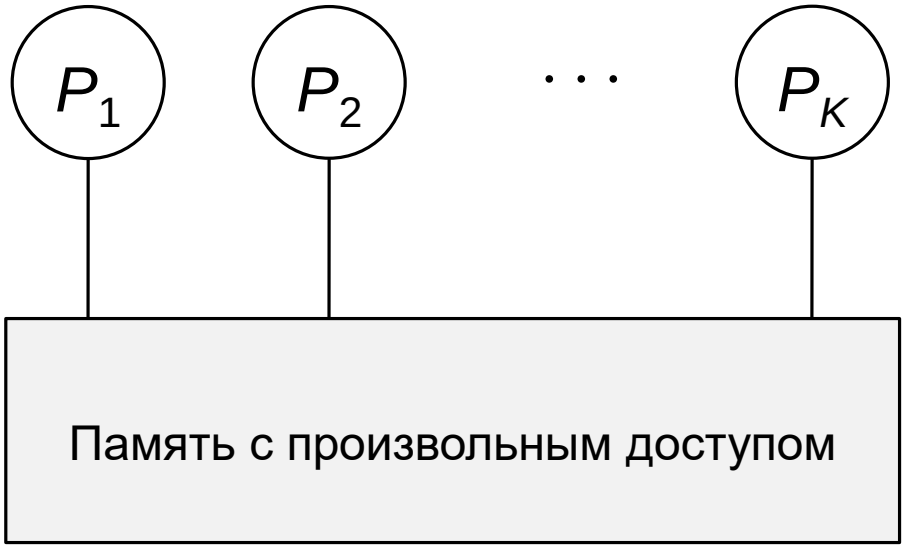
Классификация моделей параллельных вычислений

- Модели для многопроцессорных систем с общей памятью
- Модели для многопроцессорных систем с распределенной памятью
- Гибридные модели

PRAM – модель параллельных вычислений над общей памятью

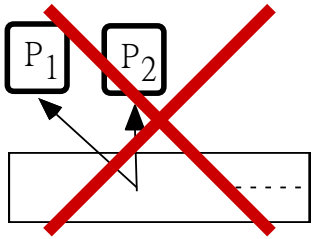
- PRAM (*Parallel Random Access Machine*) появилась в конце 1970-х годов как естественное расширение модели RAM (Random Access Machine)
- Все процессоры работают синхронно под управлением одного тактового генератора и имеют произвольный доступ к общему адресному пространству оперативной памяти
- Каждый процессор может выполнить арифметическую операцию, логическую операцию или операцию доступа к памяти за один машинный такт

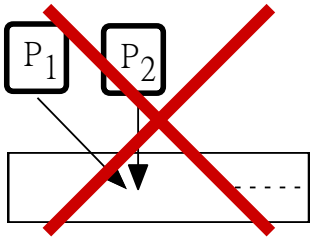
PRAM-компьютер



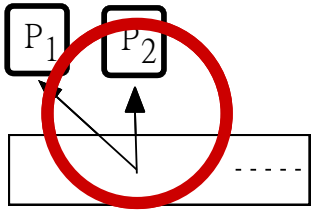
Три варианта модели PRAM

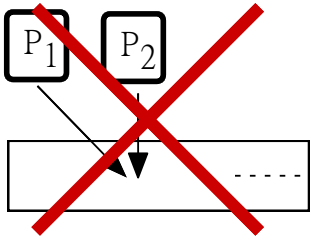
- ***EREW*** (*Exclusive Read Exclusive Write*) запрещает параллельный доступ к ячейке памяти и по чтению, и по записи



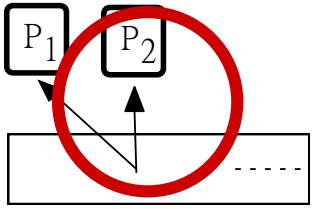


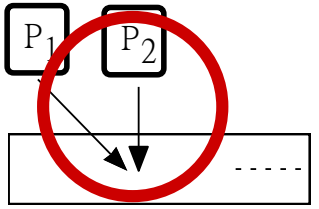
- ***CREW*** (*Concurrent Read Exclusive Write*) разрешает параллельные чтения значения из ячейки, но запрещает параллельную запись в одну и ту же ячейку памяти





- ***CRCW*** (*Concurrent Read Concurrent Write*) разрешает параллельный доступ к ячейке памяти как по чтению, так и по записи





Задача: являются ли все элементы двоичного массива M нулями?

Алгоритм 1

Все элементы M нули?

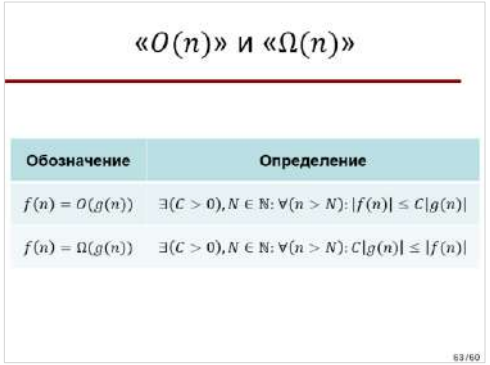
```
input
    бинарный массив M[1..n]
output
    бинарная переменная A:
        A=1 – «Да»;
        A=0 – «Нет».
begin
    A := 1
    for i = 1 to n pardo
        if M[i]=1 then A := 0
    end for
    return A
end
```

Время работы алгоритма на *n* процессорах

CREW	CRCW
$O(n)$	$O(1)$

Для любой CREW-реализации

$$t = \Omega(\log n)$$



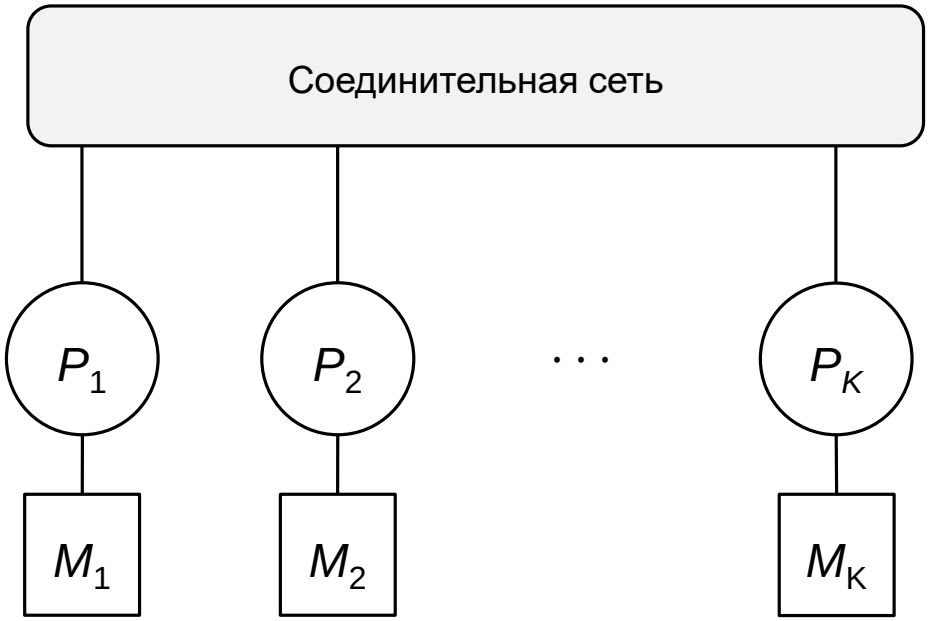
Недостатки модели PRAM

1. Временная сложность алгоритма зависит от аппаратной реализации (EREW, CREW или CRCW)
   - PRAM является низкоуровневой моделью
2. Не учитываются расходы на межпроцессорные коммуникации
   - PRAM не применима к многопроцессорным системам с распределенной памятью

BSP – модель параллельных вычислений над распределенной памятью

- *Модель BSP* (*Bulk-Synchronous Parallelism*) предложена Валиантом (Valiant) в 1990 г.
- BSP широко используется при разработке и анализе параллельных алгоритмов и программ
- Время измеряется в машинных тактах

BSP-компьютер



Параметры BSP-модели

g – время, необходимое для передачи по сети одного машинного слова

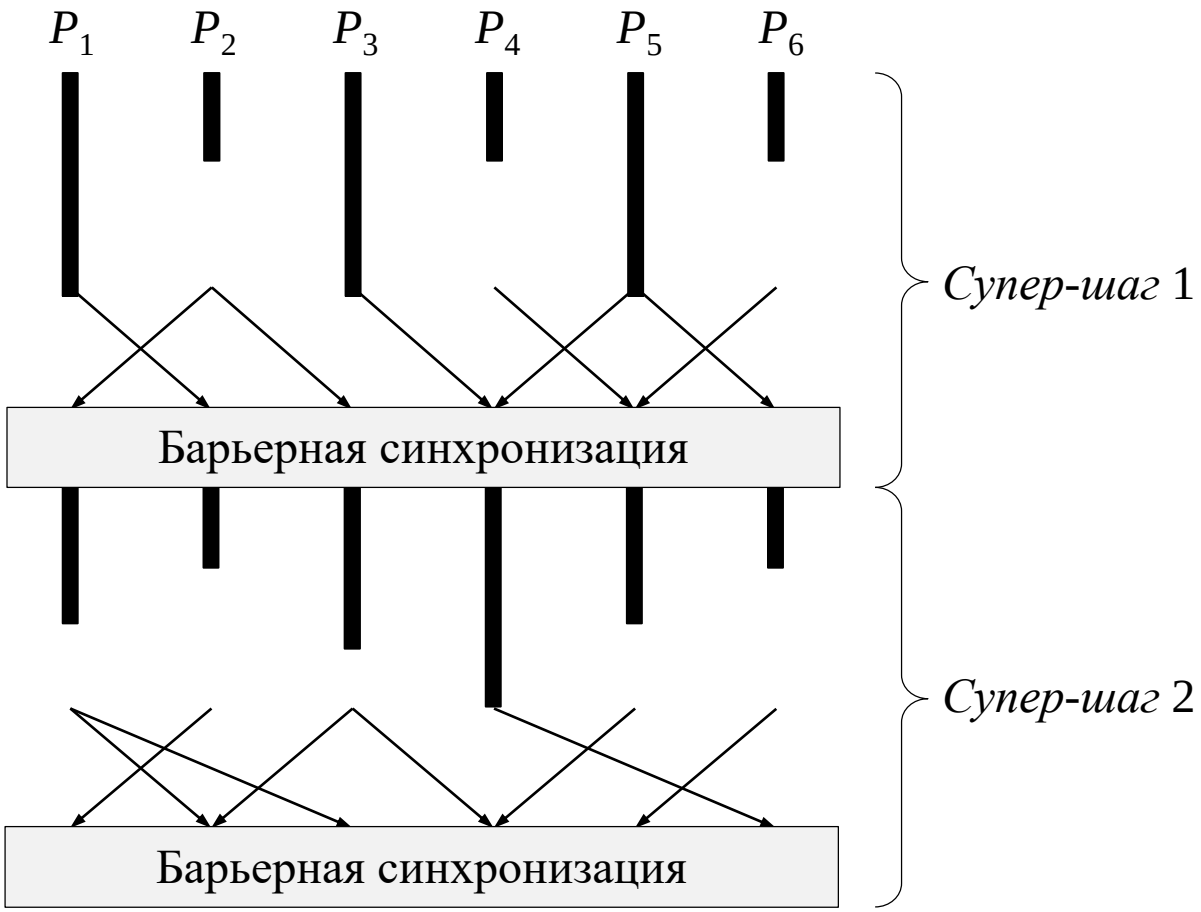
L – время, необходимое для выполнения глобальной синхронизации

h-сессия: каждый процессор передает не более *h* машинных слов и получает не более *h* машинных слов

$h \cdot g$ – время выполнения h-сессии

Схема выполнения BSP-программы

- *BSP-программа* состоит из *n* потоков команд
- Каждый поток команд делится на последовательные *супершаги*
- Супершаг:
  1) Вычисления на каждом процессоре с использованием только локальных данных
  2) h-сессия
  3) Глобальная барьерная синхронизация



Стоимостная функция модели BSP

- BSP-программа состоит из S супершагов
- Каждый процессор на i-том супершаге выполняет максимум w_i тактов в ходе локальных вычислений
- Время выполнения i-того супершага:

$$t_i = w_i + h \cdot g + L$$

- Время выполнения всей программы:

$$T = h \cdot g \cdot S + L \cdot S + \sum_{i=1}^{S} w_i$$

Пример использования модели BSP

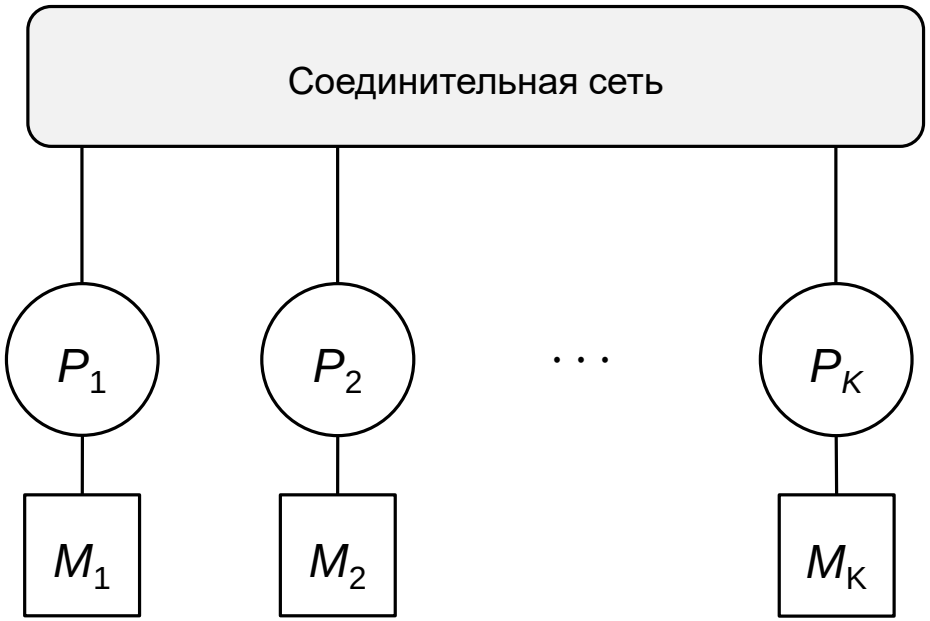
- Умножение матрицы $A_{n\times n}$ на вектор b
- Задействуем $K = n + 1$ процессоров $P_0, P_1, \dots, P_n$
- Строка a_i хранится в M_i $(i = 1, \dots, n)$
- Вектор b хранится во всех M_i $(i = 1, \dots, n)$
- Процессоры P_i $(i = 1, \dots, n)$ параллельно вычисляют $\langle a_i, b\rangle$ и пересылают результат на P_0, формирующий результирующий вектор
- Вычисление $\langle a_i, b\rangle$ состоит из n умножений и $(n-1)$ сложений $\Rightarrow w = O(n) + O(n-1) \approx O(n)$
- Затраты на пересылку $= n \cdot g + L$
- Общее время: $T = n \cdot g + L + O(n)$

Недостатки модели BSP

- Допускаются только короткие сообщения (в одно машинное слово)
- Сообщения, передаваемые в конце выполнения супершага, не могут быть использованы реципиентом до начала следующего супершага
- BSP предполагает аппаратную поддержку механизма глобальной синхронизации
- Проблемы с балансировкой загрузки процессоров

Модель LogP – усовершенствование модели BSP

- *Модель LogP* предложена в 1993 г.
- Архитектура LogP-компьютера такая же как и в модели BSP



Параметры модели LogP

L – (*latency*) время передачи сообщения в одно машинное слово

o – (*overhead*) промежуток времени, в течении которого процессорный модуль занят приемом или передачей сообщения

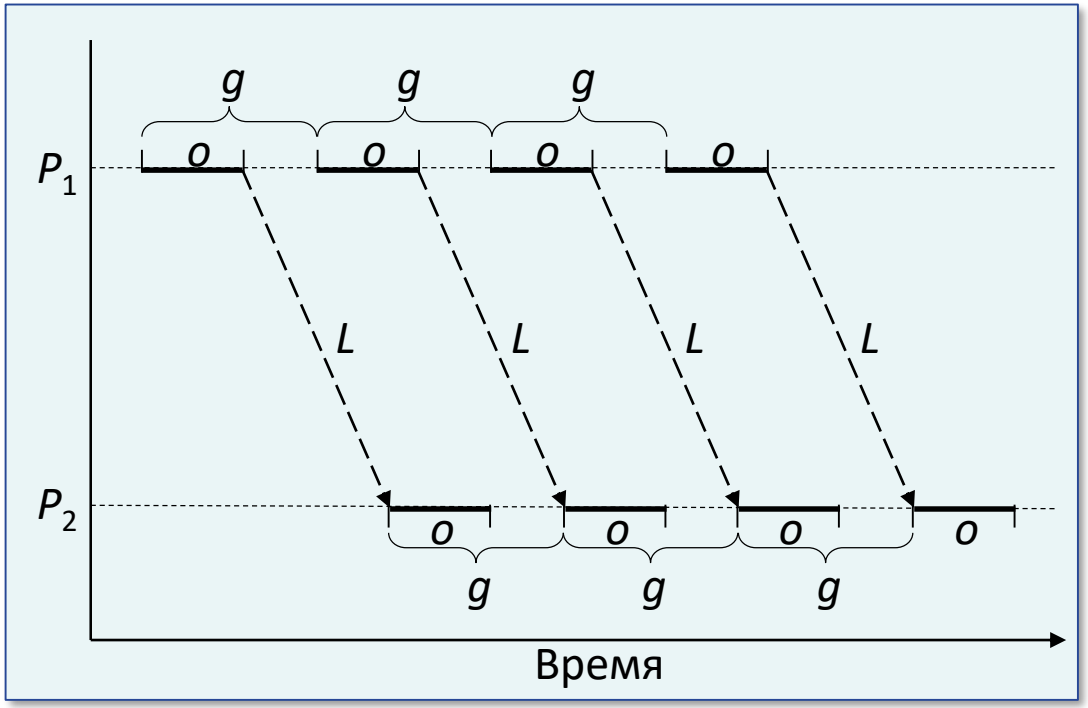
g – (*gap*) минимально время между двумя последовательными операциями чтения или передачи сообщений

P – количество процессорных модулей

- Один процессор не может одновременно посылать/получать более $\lceil L/g \rceil$ сообщений
- Допускаются только короткие сообщения фиксированной длины
- Время выполнения алгоритма определяется как максимум временных затрат среди всех процессорных модулей, участвующих в вычислениях

Стоимость передачи данных в LogP

- Время передачи сообщения от P_1 к P_2 $= o + L + o$
- Время доступа к ячейке памяти на другом процессорном модуле $= 2L + 4o$
- Время конвейерной передачи n сообщений от P_1 к P_2 $= (n-1)g + o + L + o$



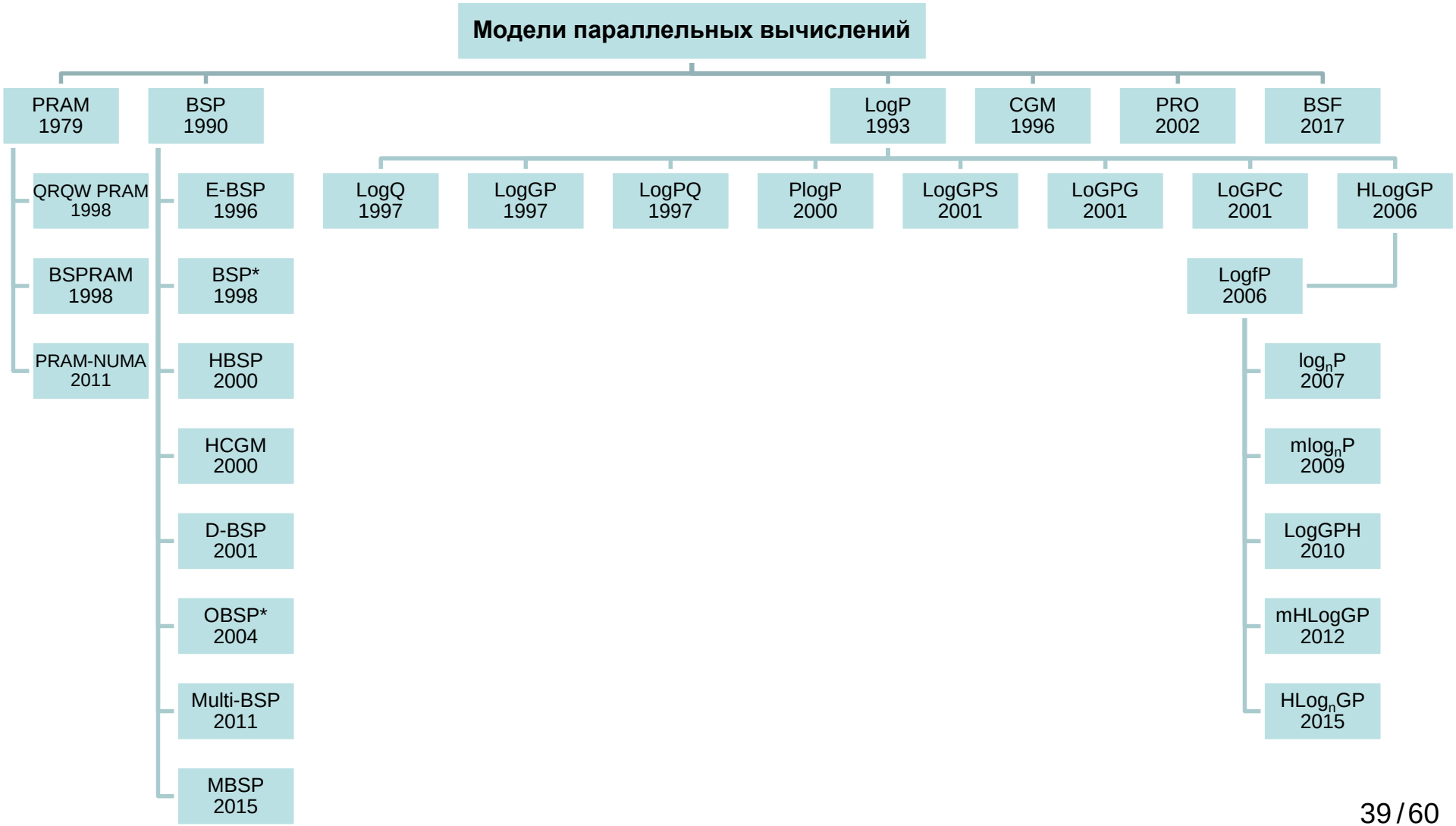
Конвейерная передача сообщений между P_1 и P_2

Особенности модели LogP

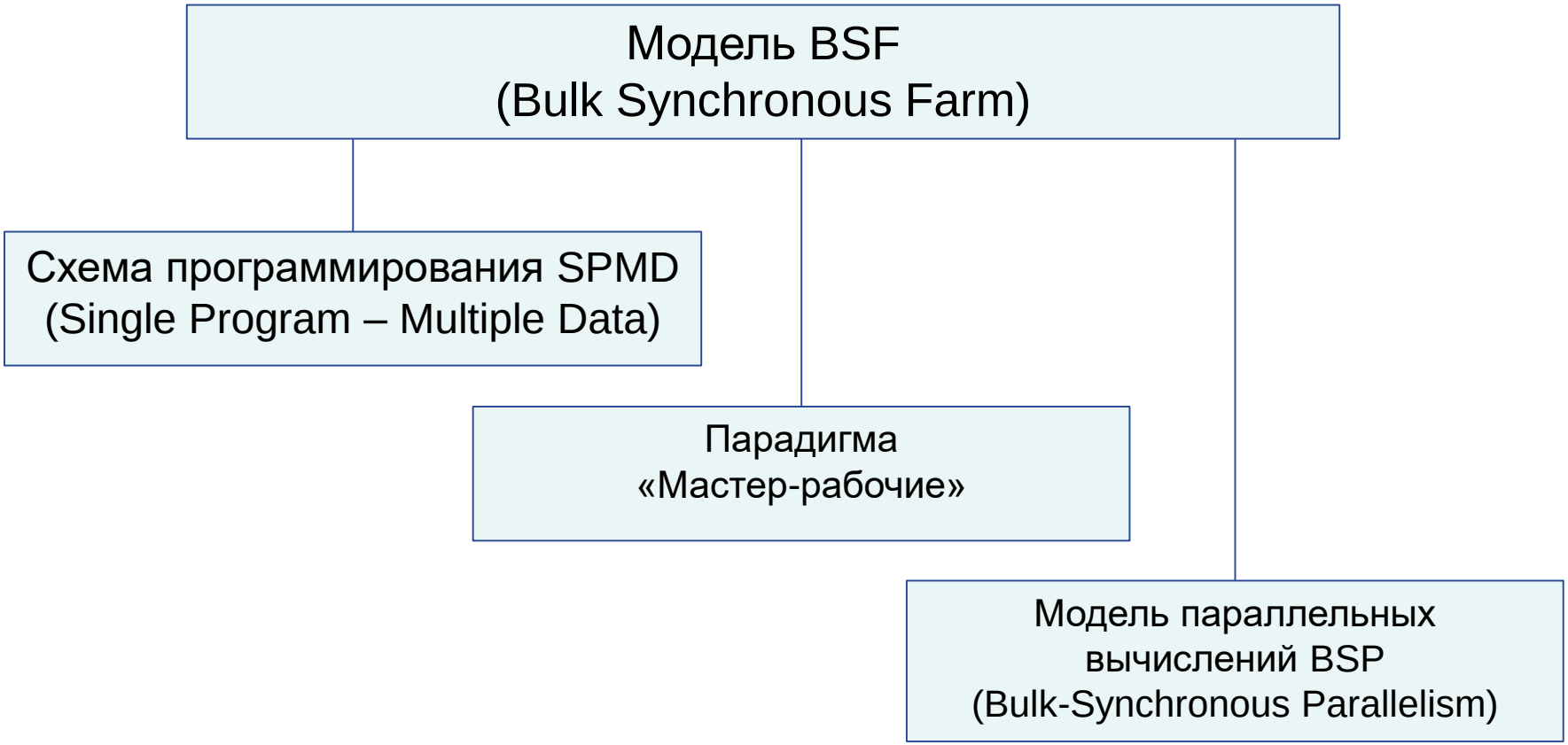
+Повышение точности прогноза

- Понижение уровня абстракции

Дерево моделей параллельных вычислений



Модель параллельных вычислений BSF (2017)



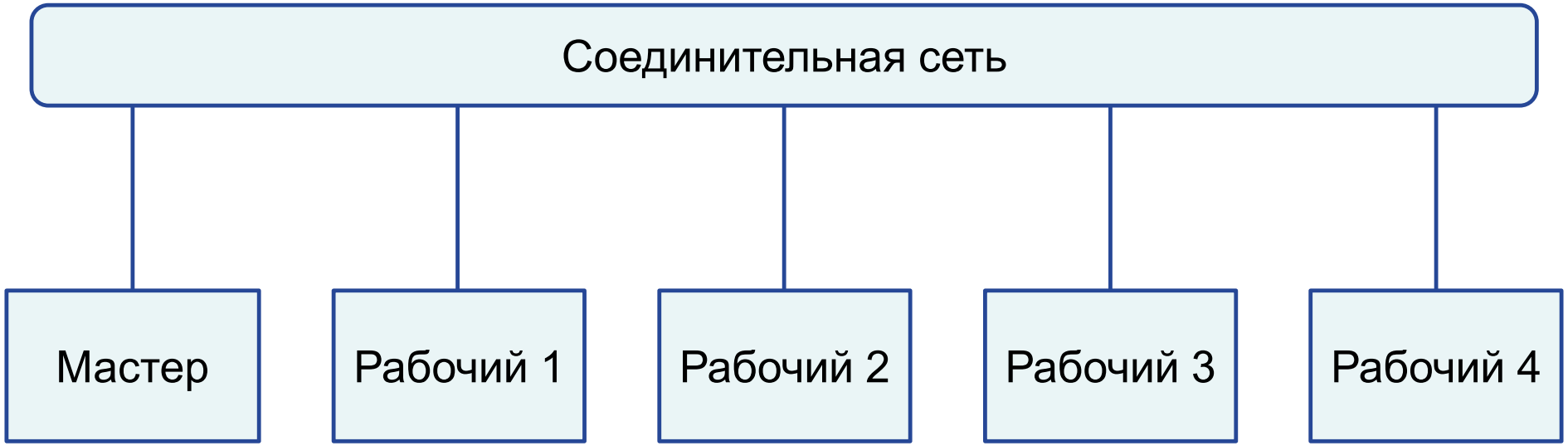
Область применения модели BSF

- Многопроцессорные системы с распределенной памятью
- Параллельные итерационные алгоритмы с высокой вычислительной сложностью
- Исследование масштабируемости алгоритма

Модель BSF позволяет **предсказать**:

- границу масштабируемости параллельного алгоритма
- ускорение алгоритма
- параллельную эффективность алгоритма

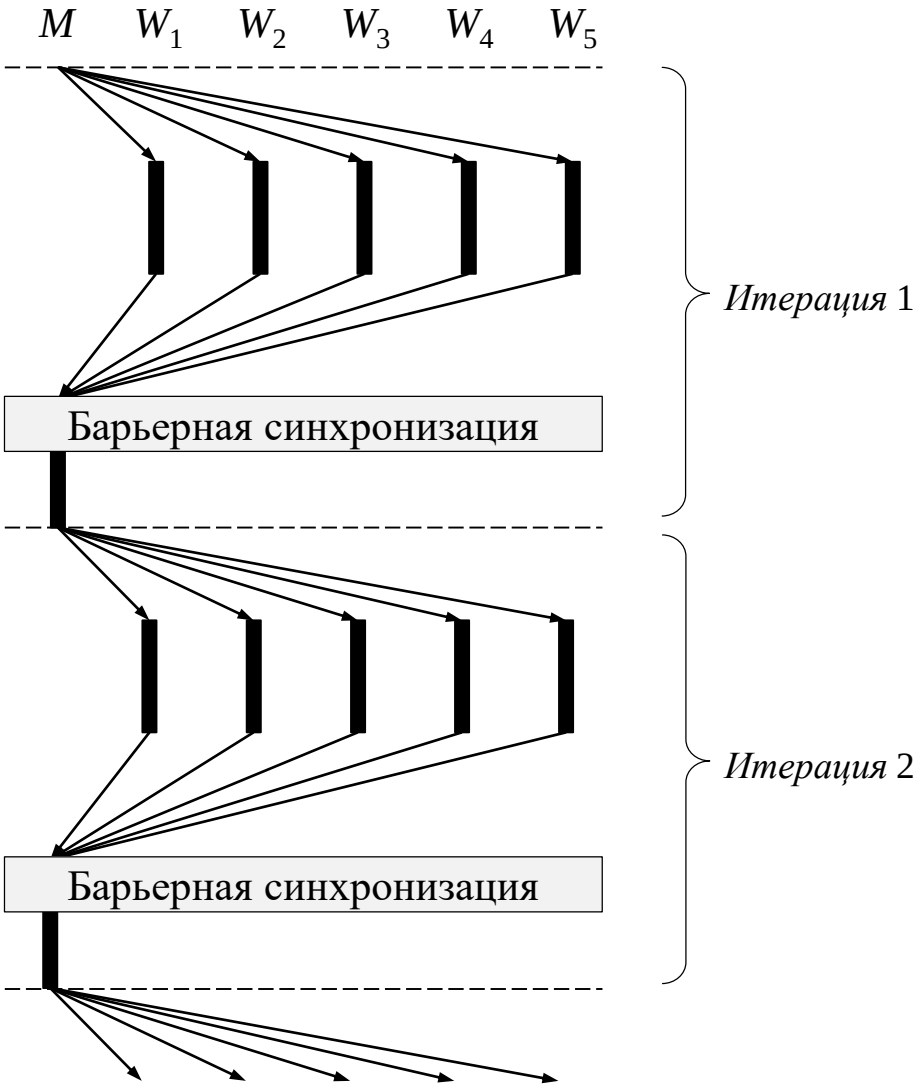
BSF-компьютер



Процессорные узлы

Структура BSF-программы

Схема работы *BSF*-алгоритма



Параметры BSF-модели

t_s – время, необходимое для посылки сообщения одному рабочему

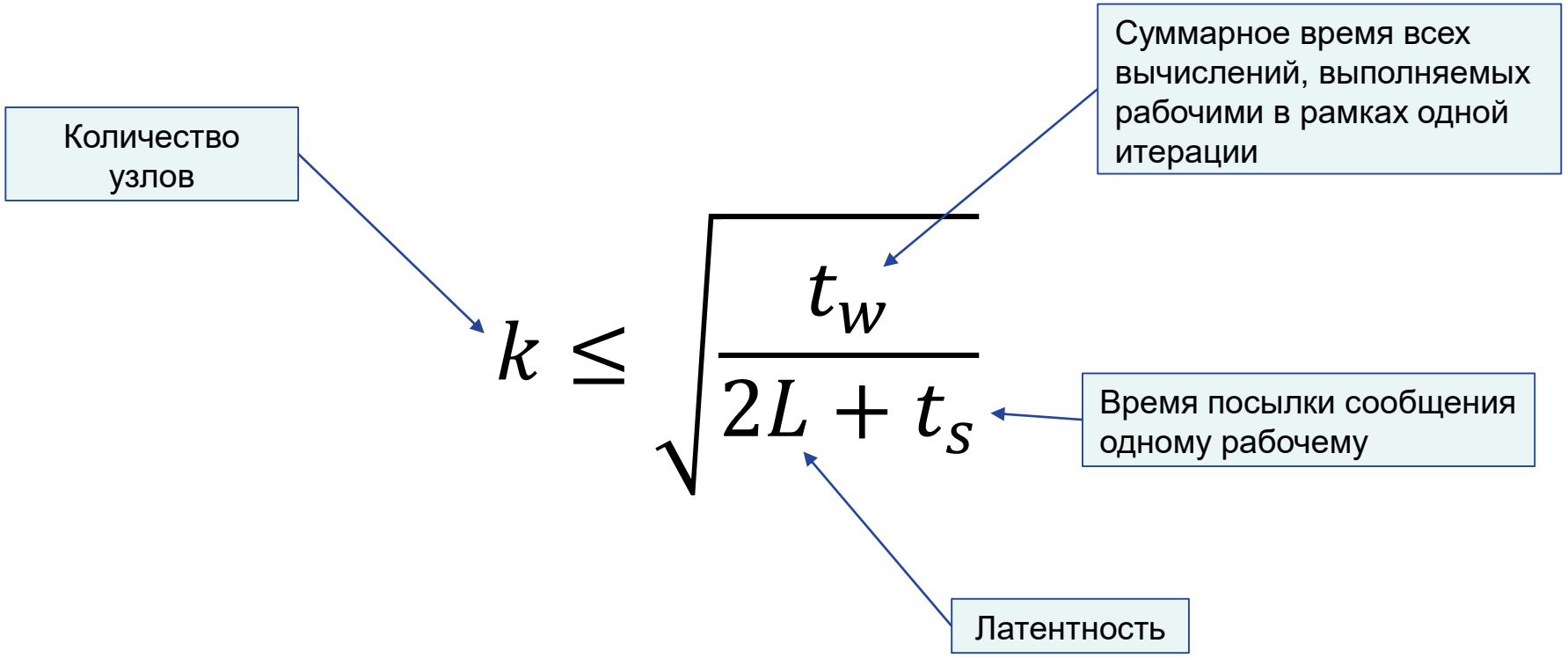
L – латентность (время посылки сообщения длиной в 1 байт)

t_w – суммарное время всех вычислений, выполняемых рабочими в рамках одной итерации

t_r – время, необходимое для передачи результатов мастеру от рабочих

t_p – время обработки мастером результатов, полученных от рабочих

Граница масштабируемости BSF-алгоритма



Граница масштабируемости не зависит от:

t_r – время передачи результатов мастеру от рабочих

t_p – время обработки мастером результатов, полученных от рабочих

Ускорение BSF-алгоритма

$$a = \frac{k(2L + t_s + t_r + t_p + t_w)}{k^2(2L + t_s) + k(t_r + t_p) + t_w}$$

k – количество процессорных узлов

t_s – время посылки сообщения одному рабочему

L – латентность

t_w – время вычислений, выполняемых рабочими

t_r – время передачи результатов мастеру от рабочих

t_p – время обработки результатов, полученных от рабочих

Параллельная эффективность BSF-алгоритма

$$e = \frac{1}{1 + \left(k^2(2L + t_s) + k(t_r + t_p)\right)/t_w}$$

- k – количество процессорных узлов
- t_s – время для посылки сообщения одному рабочему
- L – латентность
- t_w – время вычислений, выполняемых рабочими
- t_r – время передачи результатов мастеру от рабочих
- t_p – время обработки результатов, полученных от рабочих

Применение модели BSF для исследования алгоритма NSLP

Нестационарная задача линейного программирования

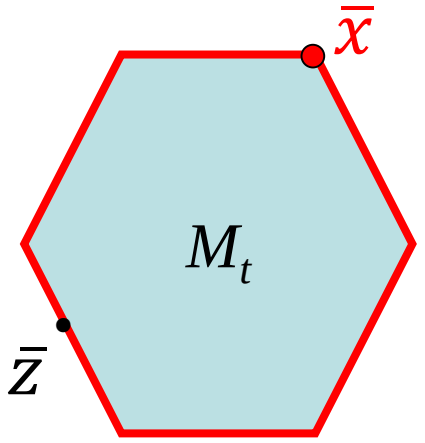
$$\max\{\langle c_t, x\rangle | A_t x \leq b_t, x \geq 0\}$$

- $x \in \mathbb{R}_n$
- A_t – матрица $m \times n$
- c_t, b_t – векторы размерности n
- $t \in \mathbb{R}_{\geq 0}$ – время

Алгоритм NSLP (Non Stationary Linear Programming)

Фазы алгоритма:

- *Quest* – поиск точки $\bar{z} \in M_t$
- *Targeting* – перемещение точки $\bar{z}$ таким образом, чтобы точное решение $\bar{x}$ задачи ЛП находилось в ее ε-окрестности

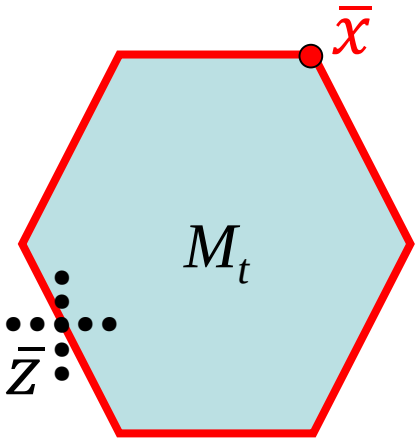


$$A_t x \leq b_t \Leftrightarrow x \in M_t$$

Алгоритм NSLP (Non Stationary Linear Programming)

Фазы алгоритма:

- *Quest* – поиск точки $\bar{z} \in M_t$
- *Targeting* – перемещение точки $\bar{z}$ таким образом, чтобы точное решение $\bar{x}$ задачи ЛП находилось в ее ε-окрестности



$$A_t x \leq b_t \Leftrightarrow x \in M_t$$

Граница масштабируемости BSF-реализации алгоритма NSLP

Меняются все элементы матрицы A:

$$k \leq O\left(\sqrt{n}\right)$$

Меняется один элемент матрицы A:

$$k \leq O(n)$$

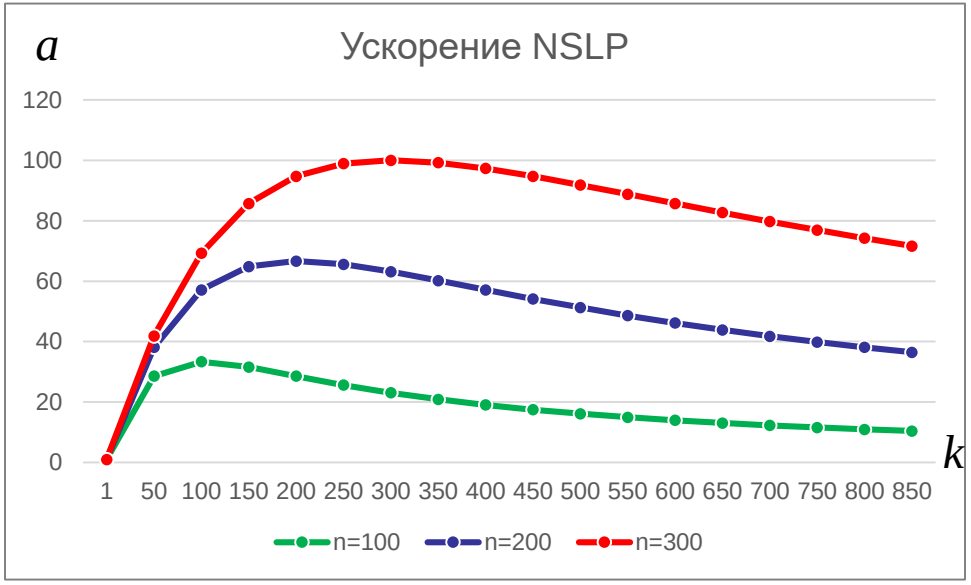
k – количество процессорных узлов

n – размерность задачи

Ускорение для BSF-реализации алгоритма NSLP

$$a = \frac{O(n^3)k}{O(n)k^2 + O(n^2)k + O(n^3)}$$

(меняется один элемент матрицы A)



k – количество процессорных узлов

n – размерность задачи

Параллельная эффективность BSF-реализации алгоритма NSLP

Меняются все элементы матрицы A:

$$e = \frac{1}{1 + k^2/O(n)}$$

Меняется один элемент матрицы A:

$$e = \frac{1}{1 + k^2/O(n^2) + k/O(n)}$$

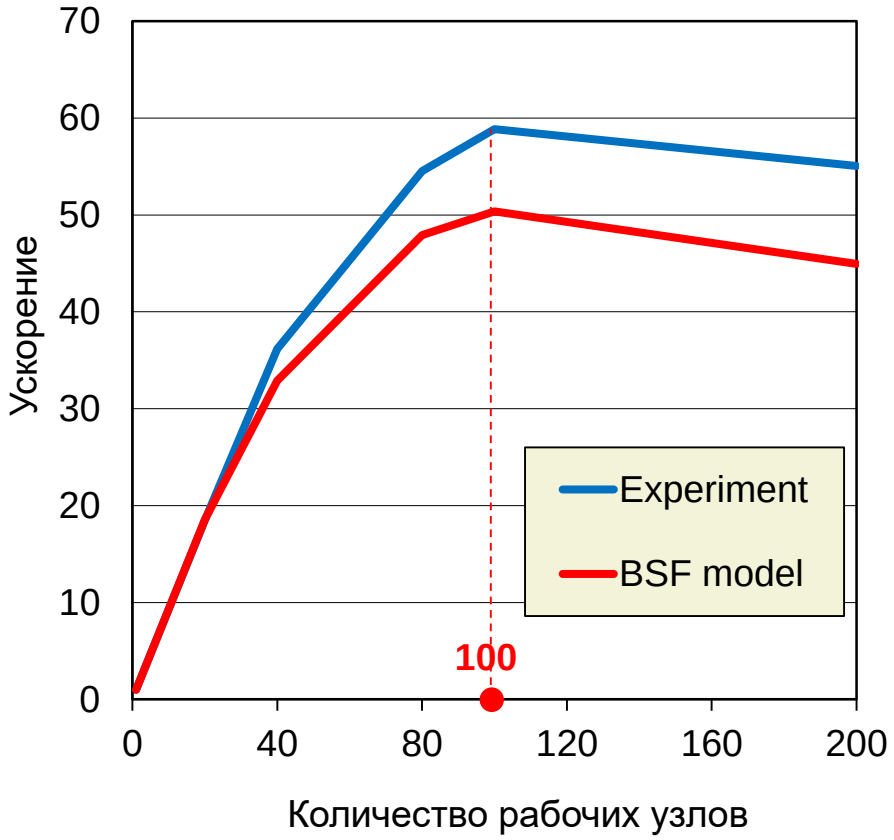
k – количество процессорных узлов

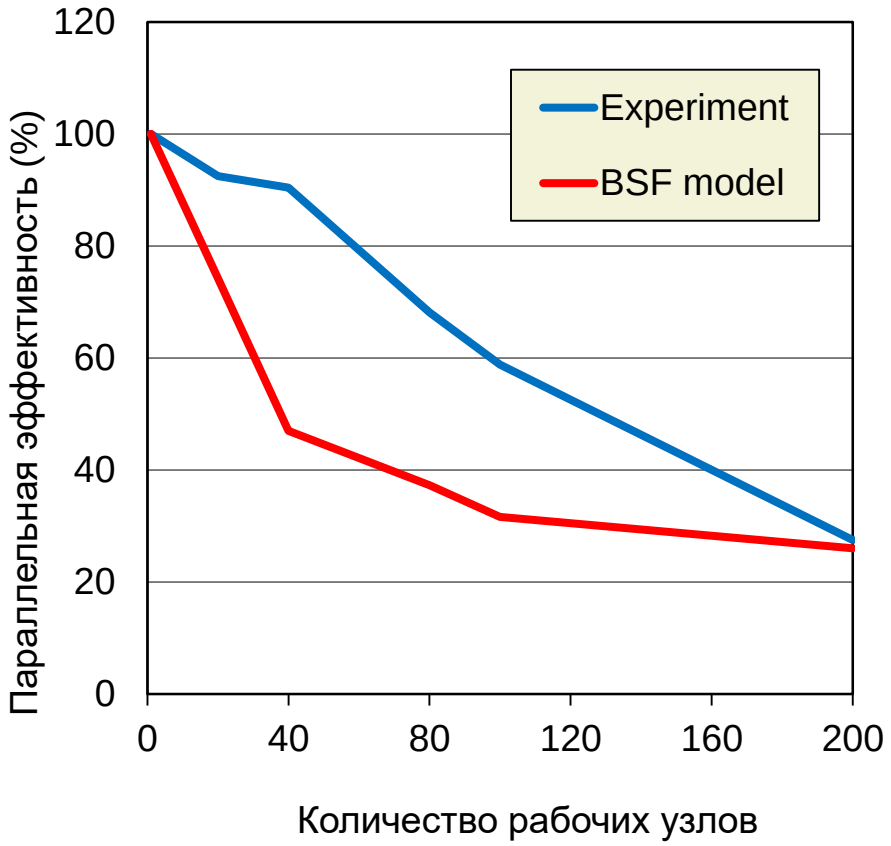
n – размерность задачи

BSF-реализация алгоритма NSLP на C++ и MPI

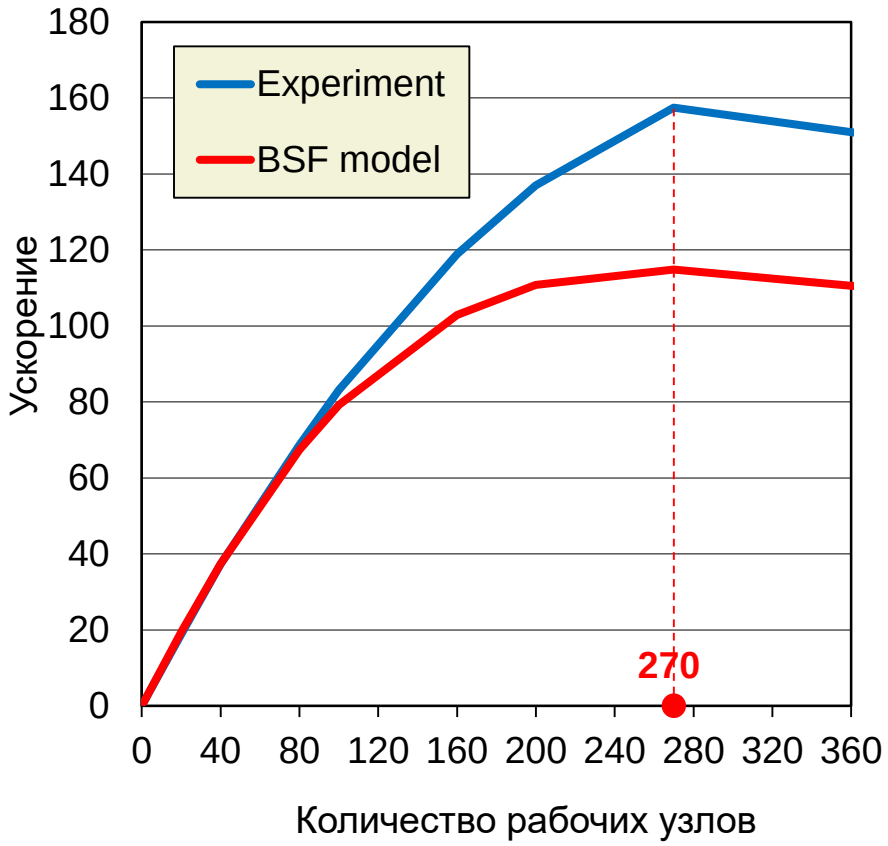
https://github.com/leonid-sokolinsky/BSF-NSLP

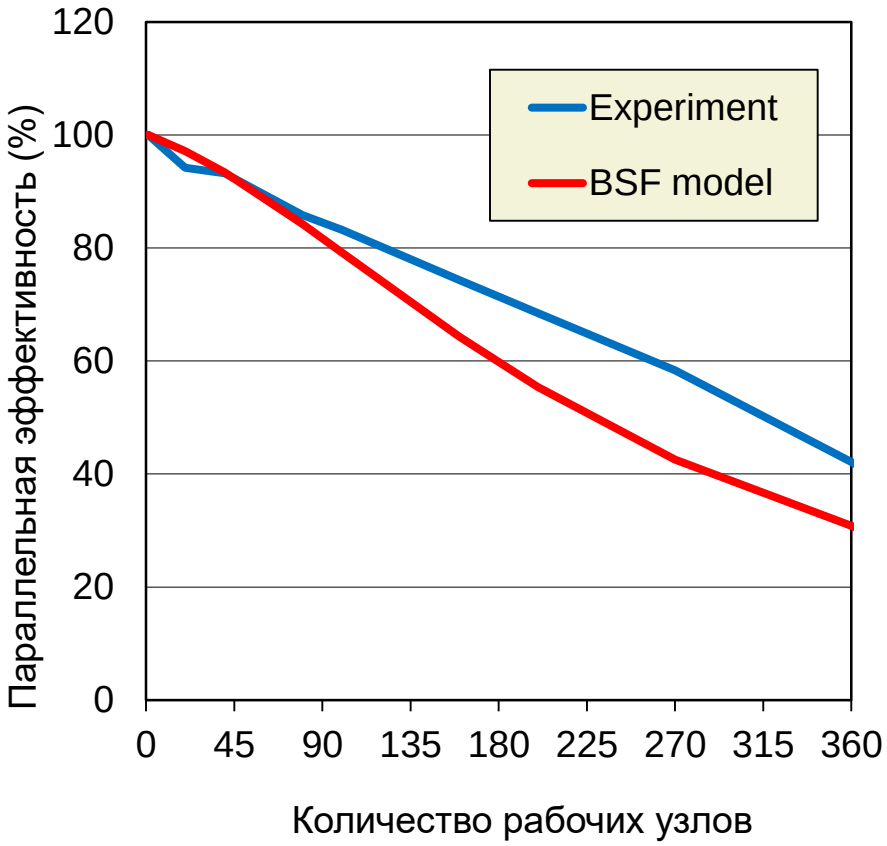
Сравнение теории с экспериментом (n = 400)



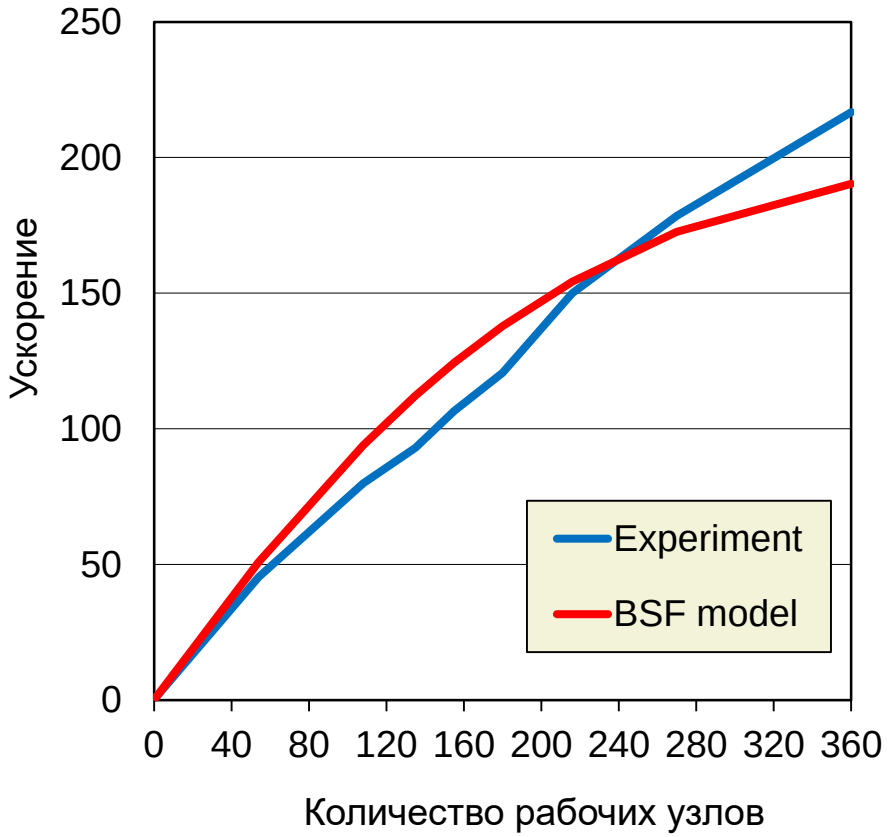


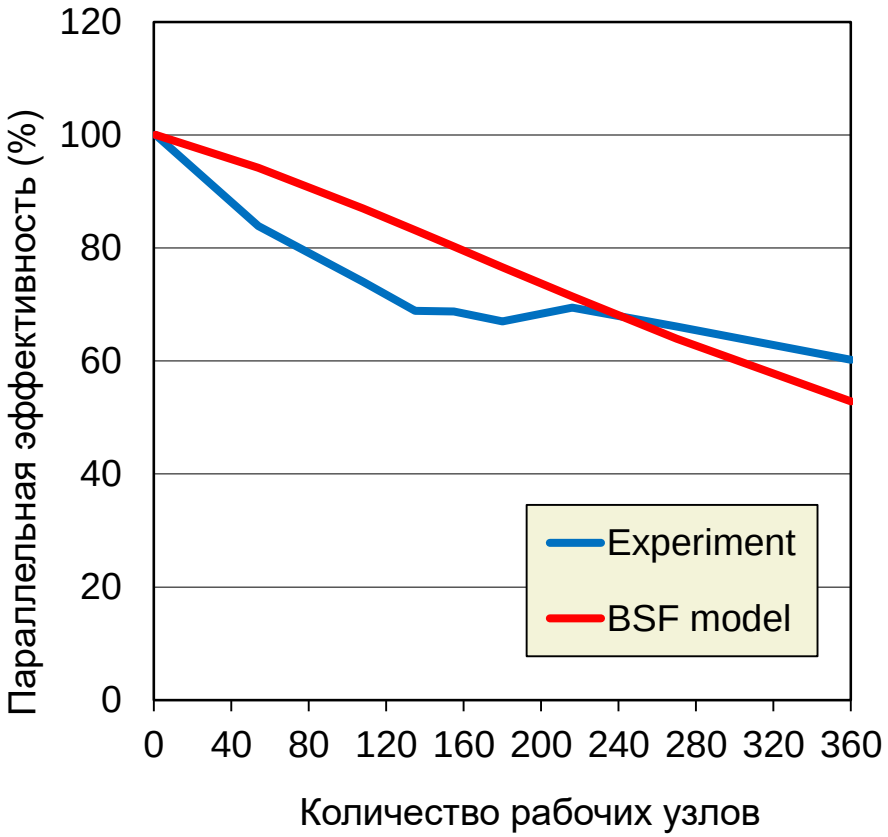
Сравнение теории с экспериментом (n = 800)





Сравнение теории с экспериментом (n = 1080)





Библиография

1. *Bilardi G., Pietracaprina A.* Models of Computation, Theoretical // Encyclopedia of Parallel Computing. Boston, MA: Springer US, 2011. P. 1150–1158.
2. JaJa J.F. PRAM (Parallel Random Access Machines) // Encyclopedia of Parallel Computing. Boston, MA: Springer US, 2011. P. 1608–1615.
3. Tiskin A. BSP (Bulk Synchronous Parallelism) // Encyclopedia of Parallel Computing. Boston, MA: Springer US, 2011. P. 192–199.
4. Kielmann T., Gorlatch S. Bandwidth-Latency Models (BSP, LogP) // Encyclopedia of Parallel Computing. Boston, MA: Springer US, 2011. P. 107–112.
5. *Sokolinsky L.B.* Analytical study of the “master-worker” framework scalability on multiprocessors with distributed memory [Electronic resource] // arXiv:1704.05816 [cs.DC]. 2017. P. 15. (in Russian). http://arxiv.org/abs/1704.05816
6. Sokolinskaya I., Sokolinsky L.B. Scalability Evaluation of NSLP Algorithm for Solving Non-Stationary Linear Programming Problems on Cluster Computing Systems // Communications in Computer and Information Science. 2017. Vol. 793. P. 40-53. DOI: 10.1007/978-3-319-71255-0_4. http://arxiv.org/abs/1709.04640

Спасибо за внимание!

Вопросы?

Вспомогательные слайды

«$O(n)$» и «$\Omega(n)$»

Обозначение	Определение
$f(n) = O(g(n))$	$\exists(C > 0), N \in \mathbb{N}: \forall(n > N): \lvert f(n)\rvert \leq C\lvert g(n)\rvert$
$f(n) = \Omega(g(n))$	$\exists(C > 0), N \in \mathbb{N}: \forall(n > N): C\lvert g(n)\rvert \leq \lvert f(n)\rvert$