МОДЕЛЬ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ ДЛЯ ОЦЕНКИ МАСШТАБИРУЕМОСТИ ИТЕРАЦИОННЫХ АЛГОРИТМОВ НА КЛАСТЕРНЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМАХ

05.13.11 - математическое и программное обеспечение
вычислительных машин, комплексов и компьютерных сетей

Диссертация на соискание ученой степени кандидата физико-математических наук

Ежова Надежда Александровна

Научный руководитель:
СОКОЛИНСКИЙ Леонид Борисович,
доктор физ.-мат. наук, профессор

Исследование выполнено при финансовой поддержке Правительства РФ в соответствии с Постановлением №211 от 16.03.2013 г. (соглашение № 02.A03.21.0011) и Министерства образования и науки РФ (государственное задание 2.7905.2017/8.9).

Цель диссертационной работы

Разработать и исследовать новую модель параллельных вычислений для итерационных алгоритмов применительно к многопроцессорным системам с распределенной памятью, позволяющую предсказывать границу масштабируемости алгоритма на ранних стадиях его проектирования, и построить на ее основе параллельный каркас для кластерных вычислительных систем

Основные задачи

1. Разработать модель параллельных вычислений
2. Создать на языке C++ параллельный каркас, основанный на разработанной модели
3. Выполнить проектирование и реализацию визуального конструктора программ на языке C++ в соответствии с разработанными моделью и параллельным каркасом
4. Провести вычислительные эксперименты для верификации предложенной модели и разработанного программного обеспечения

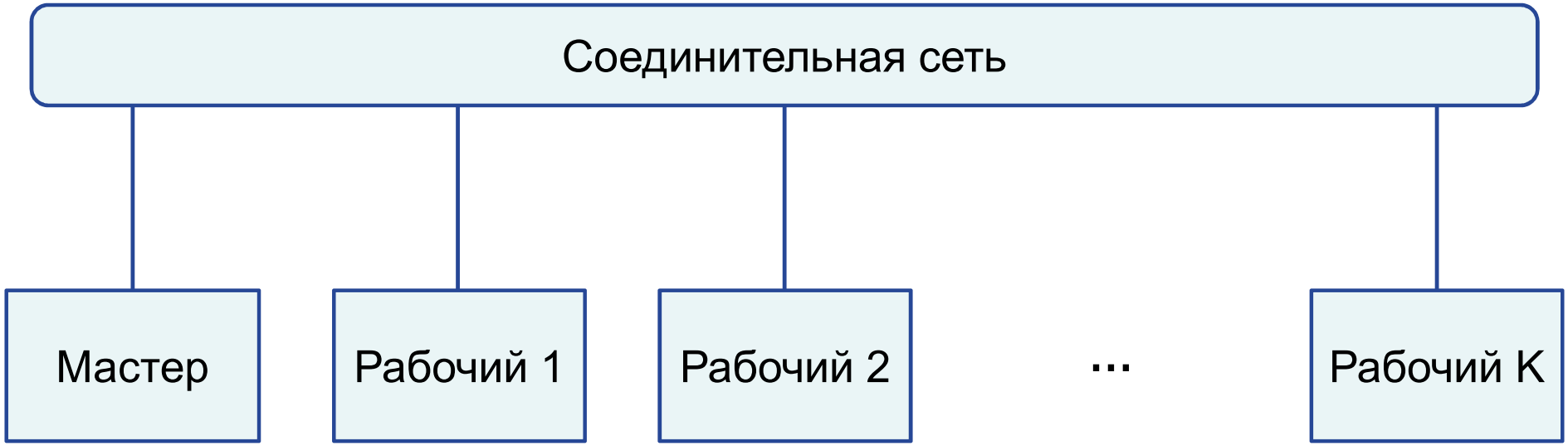
Работы по теме диссертации

1	Amaris M. et al. A Simple BSP-based Model to Predict Execution Time in GPU Applications // 2015 IEEE 22nd International Conference on High Performance Computing (HiPC). IEEE, 2015. P. 285–294. DOI:10.1109/HiPC.2015.34.	*CUDA-BSP - расширение модели BSP для ГПУ под управлением CUDA*
2	Gerbessiotis A. V. Extending the BSP model for multi-core and out-of-core computing: MBSP // Parallel Comput. Elsevier B.V., 2015. Vol. 41. P. 90–102. DOI:10.1016/j.parco.2014.12.002.	*MBSP - расширение модели BSP для многопроцессорных систем с многоуровневой иерархической памятью и многоядерными процессорами*
3	Lu F., Song J., Pang Y. HLognGP: A parallel computation model for GPU clusters // Concurr. Comput. Pract. Exp. 2015. Vol. 27, no. 17. P. 4880–4896. DOI:10.1002/cpe.3475	*$HLog_nGP$ - расширение модели LogGP, ориентированное на гетерогенные вычислительные кластеры*
4	Valiant L.G. A bridging model for multi-core computing // J. Comput. Syst. Sci. Elsevier Inc., 2011. Vol. 77, no. 1. P. 154–166. DOI:10.1016/j.jcss.2010.06.012.	*Multi-BSP - расширение модели BSP, ориентированное на иерархическую память (RAM, CACHE-I, CACHE-II, …)*
5	Hoefler T., Schneider T., Lumsdaine A. LogGOPSim – Simulating Large-Scale Applications in the LogGOPS Model // Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing - HPDC ’10. New York, New York, USA: ACM Press, 2010. P. 597–604. DOI:10.1145/1851476.1851564.	LogGOPS - *расширение модели LogGPS, предназначенное для оценки масштабируемости параллельных алгоритмов, ориентированных на большие вычислительные системы, работающие под управлением MPI-2*

Модель параллельных вычислений BSF (Bulk-Synchronous Farm)

- Область применения:
  - Многопроцессорные системы с распределенной памятью
  - Параллельные итерационные алгоритмы с высокой вычислительной сложностью
- Позволяет предсказать:
  - ускорение параллельного алгоритма
  - **границу масштабируемости параллельного алгоритма** (уникальное качество модели BSF)

BSF-компьютер

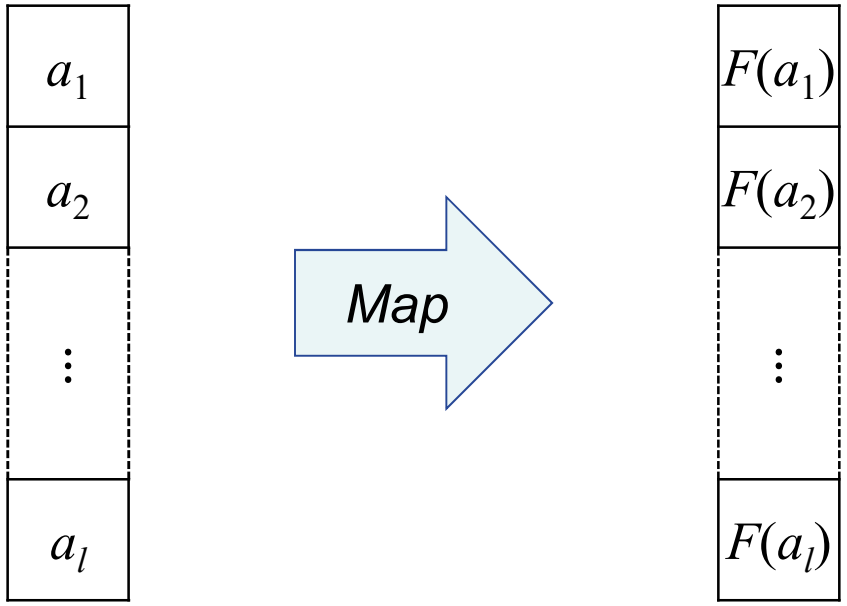


Процессорные узлы

Представление алгоритма в виде операций над списками

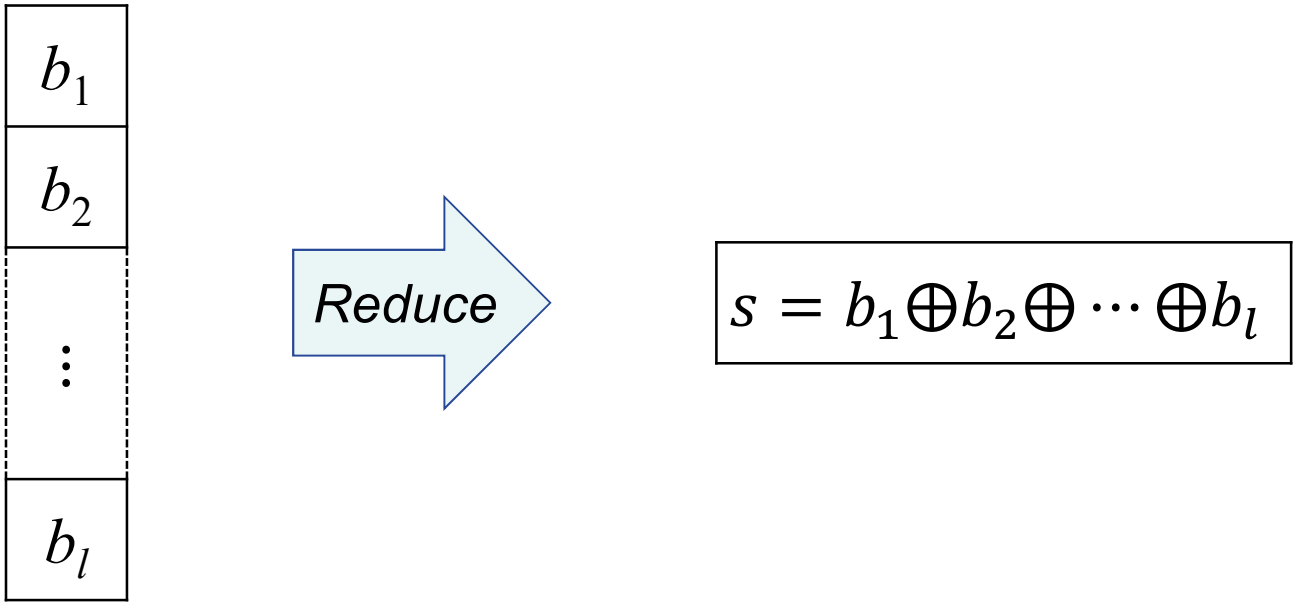
- Функции высшего порядка:
  - Map
  - Reduce

Функция высшего порядка *Map*



$$Map(F, [a_1, \ldots, a_l]) = [F(a_1), \ldots, F(a_l)]$$

Функция высшего порядка *Reduce*



$$Reduce(\oplus, [b_1, \ldots, b_l]) = b_1 \oplus \cdots \oplus b_l$$

Шаблон последовательного алгоритма в модели BSF

1. $i := 0; Input(A, x_0)$
2. $B := Map(F_{x_i}, A)$
3. $s := Reduce(\oplus, B)$
4. $x_{i+1} := Compute(x_i, s);\ i := i + 1$
5. **if** $StopCond(x_{i-1}, x_i)$ **go to** 7
6. **go to** 2
7. $Output(x_i)$; **stop**

i	- номер итерации
$A \in [\mathcal{A}]$	- список исходных элементов данных
x_0	- начальное приближение
$F_x: \mathcal{A} \to \mathcal{B}$	- параметризованная функция
$B \in [\mathcal{B}]$	- список результирующих элементов
$\oplus$	- ассоциативная операция
x_i	- *i*-тое приближение

Шаблон параллельного алгоритма в модели BSF

Шаг	Мастер	Рабочий j (j=1,…,K)
1.	$i \coloneqq 0; Input(x_0)$	$Input\left(A^{[j]}\right)$
2.	$SendToAllWorkers(x_i)$	$RecvFromMaster(x_i)$
3.		$B^{[j]} \coloneqq Map\left(F_{x_i}, A^{[j]}\right)$
4.		$s^{(j)} \coloneqq Reduce\left(\oplus, B^{[j]}\right)$
5.	$RecvFromAllWorkers([s^{(1)}, \dots, s^{(K)}])$	$SendToMaster(s^{(j)})$
6.	$s \coloneqq Reduce\left(\oplus, [s^{(1)}, \dots, s^{(K)}]\right)$	
7.	$x_{i+1} \coloneqq Compute(x_i, s);\ i \coloneqq i + 1$	
8.	**if** $StopCond(x_{i-1}, x_i)$ **then** $exit \coloneqq 1$ **else** $exit \coloneqq 0$	
9.	$SendToAllWorkers(exit)$	$RecvFromMaster(exit)$
10.	**if** $exit \coloneqq 0$ **go to** 2	**if** $exit \coloneqq 0$ **go to** 2
11.	$Output(x_i)$	
12.	**stop**	**stop**

Стоимостная метрика модели BSF

K – количество рабочих

l – длина обрабатываемого списка

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

t_s – время передачи сообщения от мастера рабочему (без учета латентности)

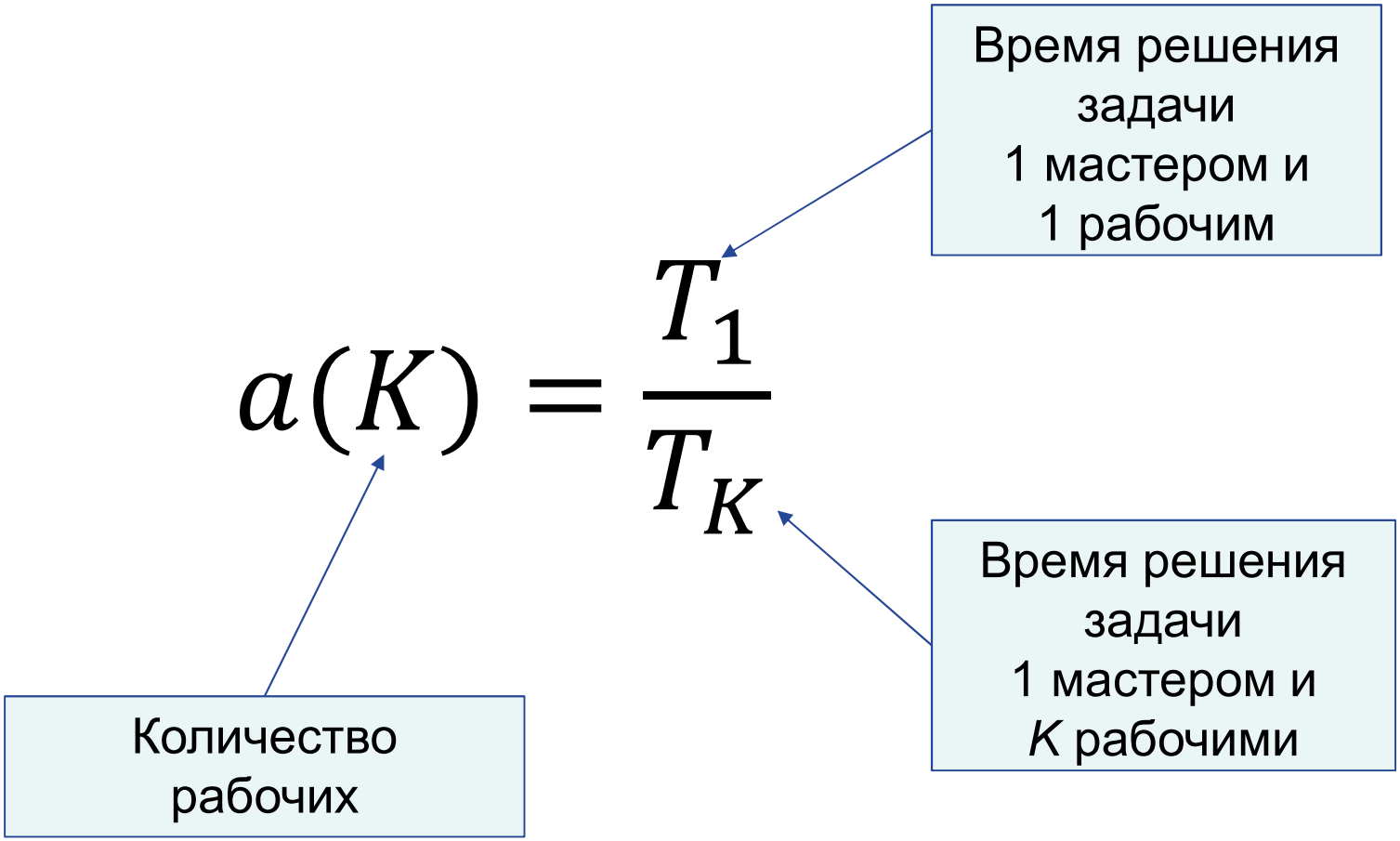
t_r – время передачи сообщения от рабочего мастеру (без учета латентности)

t_{Map} – время выполнения функции Map для всего списка исходных данных

t_a – время выполнения операции $\oplus$

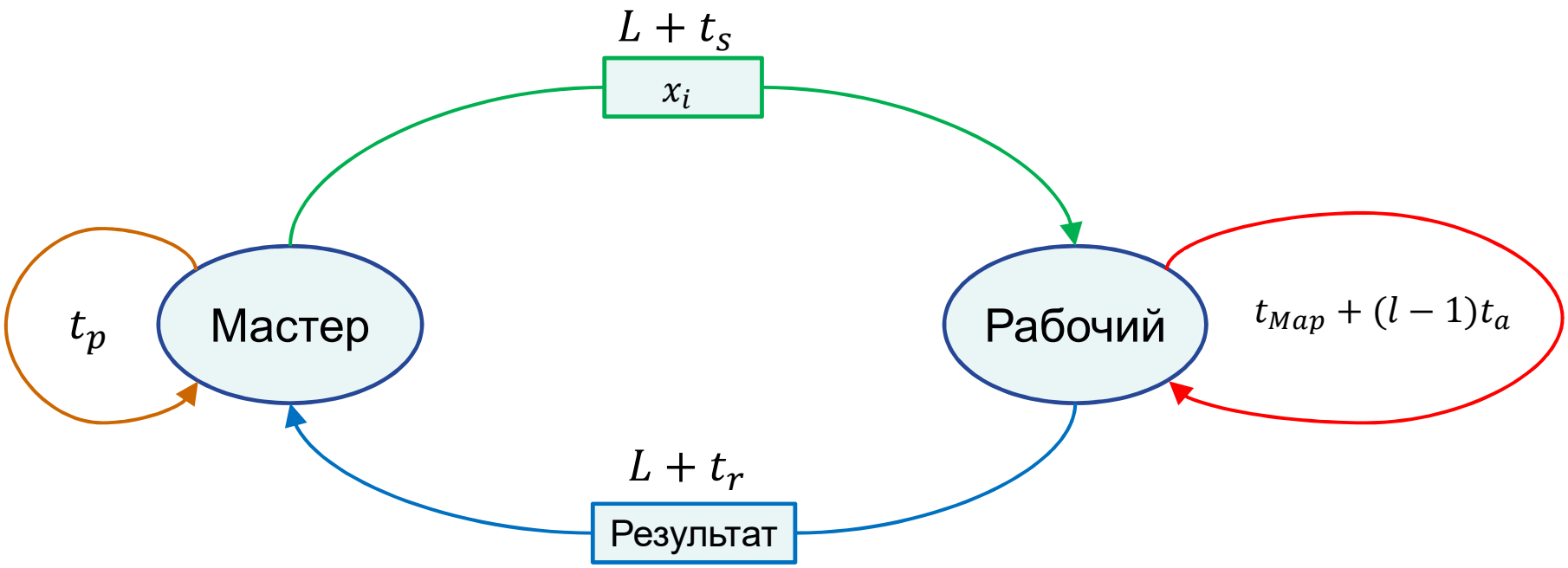
t_p – время на обработку результатов итерации

Ускорение в модели BSF



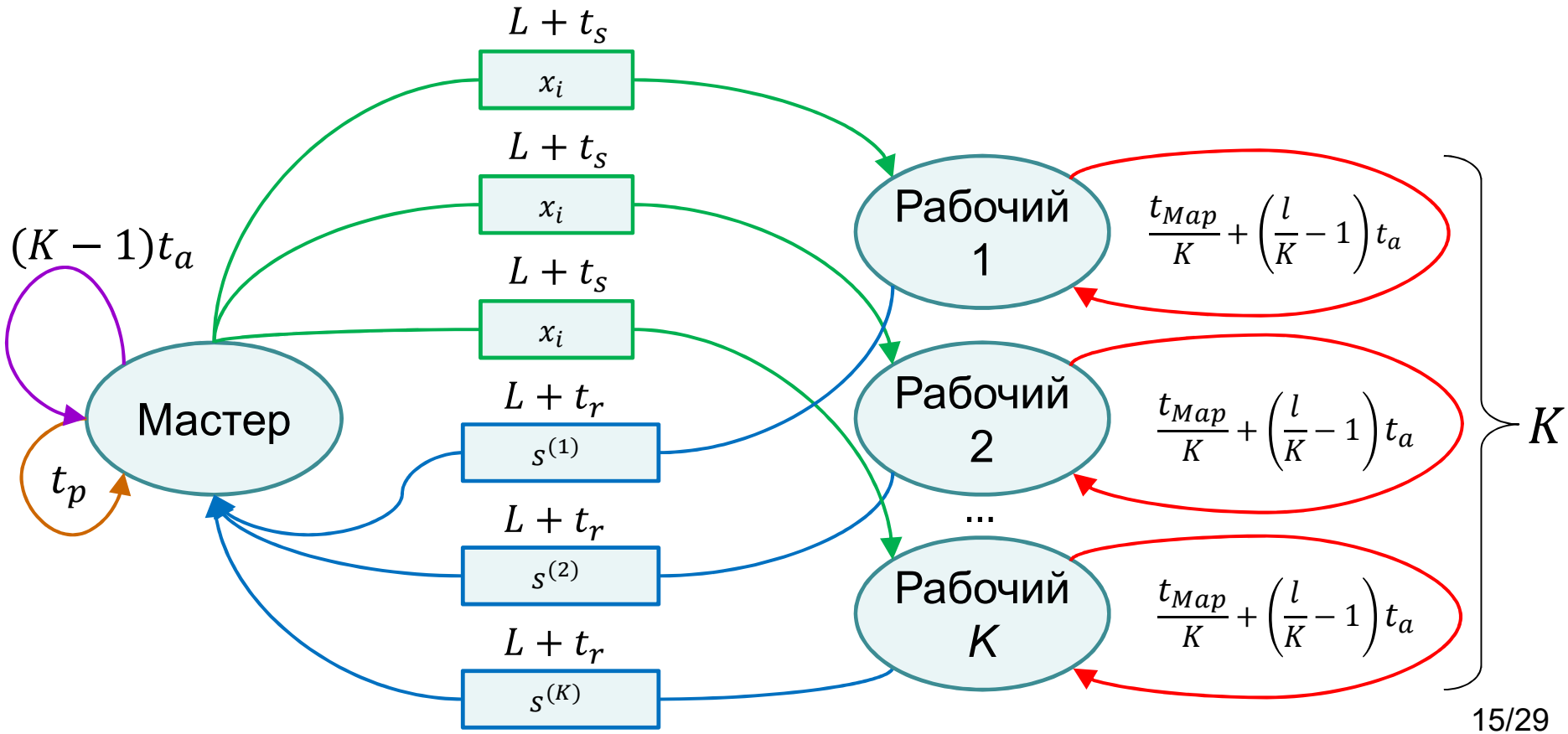
Оценка времени T_1

$$T_1 = L + t_s + t_{Map} + (l-1)t_a + L + t_r + t_p$$

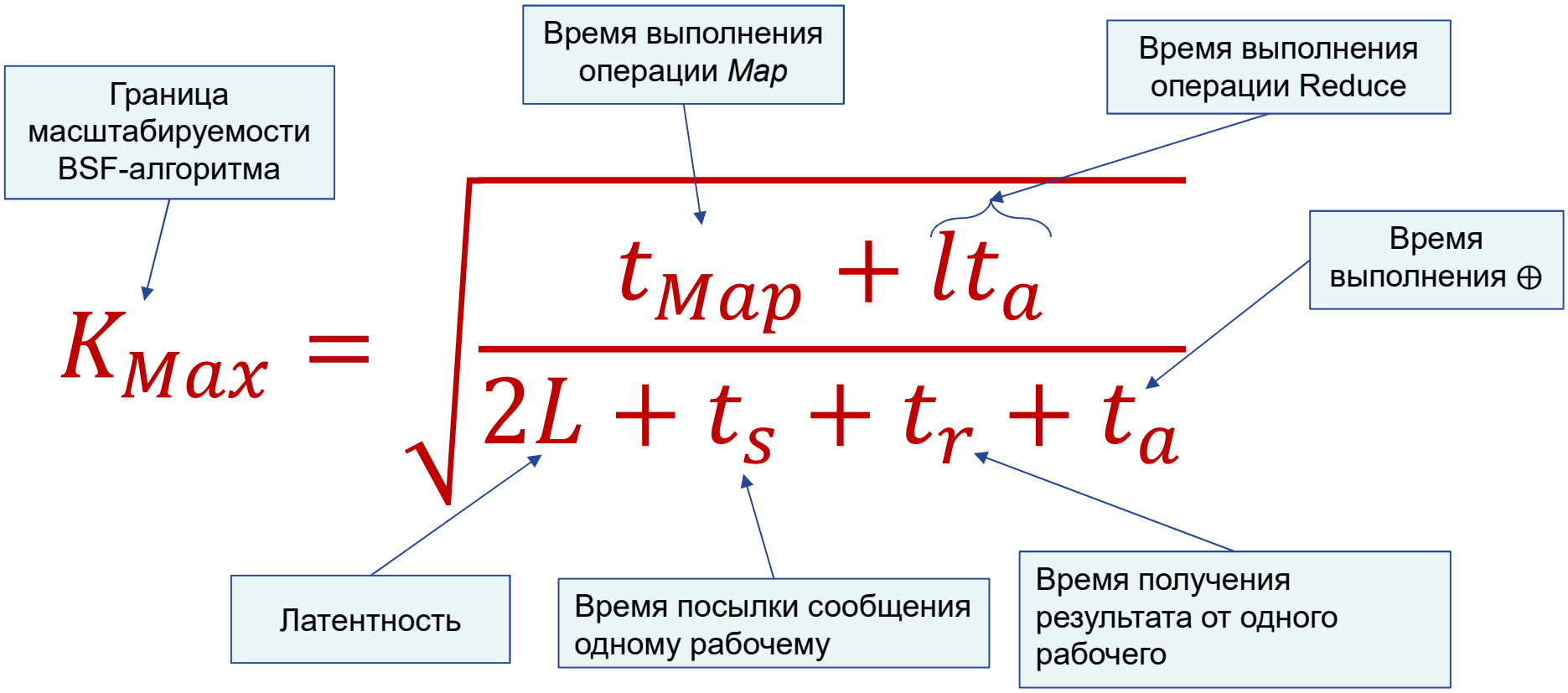


Оценка времени T_K

$$T_K = K(L + t_s) + \frac{t_{Map}}{K} + \left(\frac{l}{K} - 1\right) t_a + K(L + t_r) + (K - 1)t_a + t_p$$



Теорема о границе масштабируемости BSF-алгоритма



Параллельный BSF-каркас

- C++
- MPI + OpenMP
- Исходные коды
  github.com/nadezhda-ezhova/BSF-MR

BSF-Studio

- Веб-приложение для быстрой разработки BSF-программ на языке C++
  bsf-studio.ru
- Базируется на BSF-каркасе
- Исходные коды
  - Серверная часть:
    github.com/ezhova-nadezhda/BSF-Studio-API
  - Клиентская часть:
    github.com/ezhova-nadezhda/BSF-Studio

Верификация модели BSF

- Алгоритм Jacobi
- Гравитационная задача

Алгоритм Jacobi для приближенного решения СЛАУ

$$Ax = b$$

$$A = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix} \quad x = (x_1, \dots, x_n) \quad b = (b_1, \dots, b_n)$$

$$C = \begin{pmatrix} c_{11} & \cdots & c_{1n} \\ \vdots & \ddots & \vdots \\ c_{n1} & \cdots & c_{nn} \end{pmatrix} \quad c_{ij} = \begin{cases} -\dfrac{a_{ij}}{a_{ii}}, \forall j \neq i \\ 0, \forall j = i \end{cases}$$

$$d = (d_1, \dots, d_n) \qquad d_i = b_i / a_{ii}$$

$$x^{(k+1)} = Cx^{(k)} + d$$

Алгоритм Jacobi-BSF

$F_x(j) = \left(c_{1j}x_j, \ldots, c_{nj}x_j\right)$, j – номер столбца матрицы C

1. $Input(C, d); k := 0; x^{(0)} := d$
2. $[g^1, \ldots, g^n] := Map\left(F_{x^{(k)}}, [1, \ldots, n]\right)$
3. $g := Reduce(+, [g^1, \ldots, g^n])$
4. $x^{(k+1)} := g + x^{(k)}; k := k + 1$
5. **if** $\left\|x^{(k)} - x^{(k-1)}\right\|^2 < \varepsilon$ **go to** 7
6. **go to** 2
7. $Output\left(x^{(k)}\right)$; **stop**

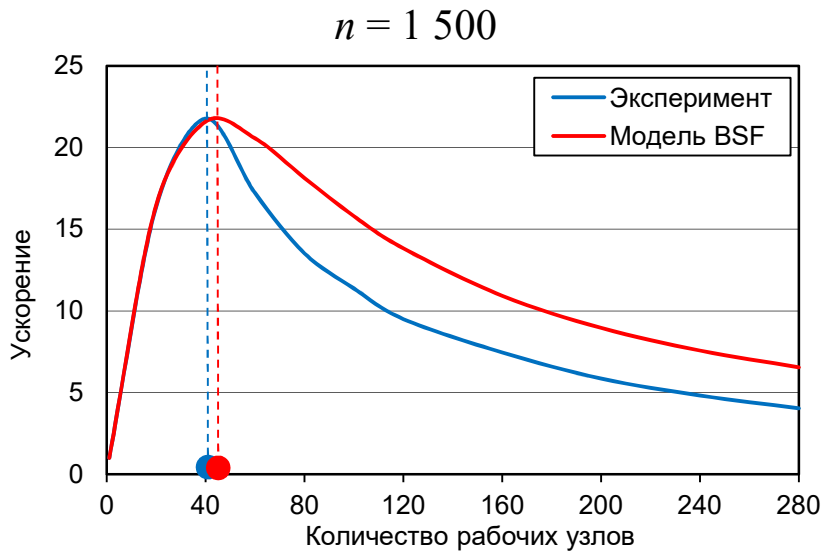
BSF-оценки для алгоритма Jacobi-BSF

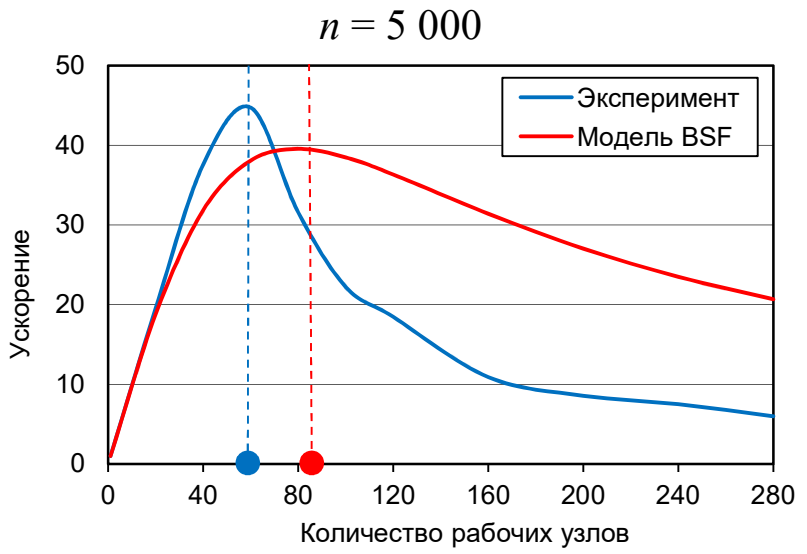
τ_{op} - время выполнения одной операции с плавающей точкой
τ_{tr} - время пересылки вещественного числа (без учета латентности)
n - количество уравнений
L - латентность

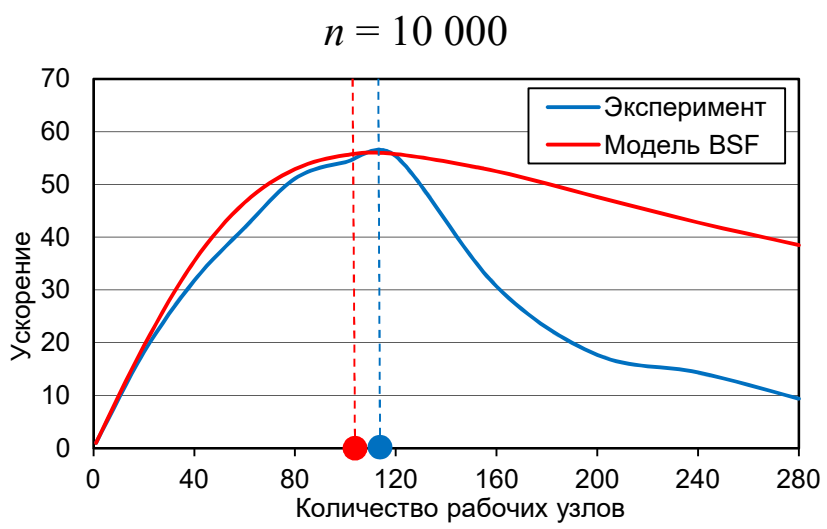
$$\Rightarrow K_{Max} = \sqrt{\frac{\tau_{op} n(3n - K)}{2(L + \tau_{tr} n) + \tau_{op} n}}$$

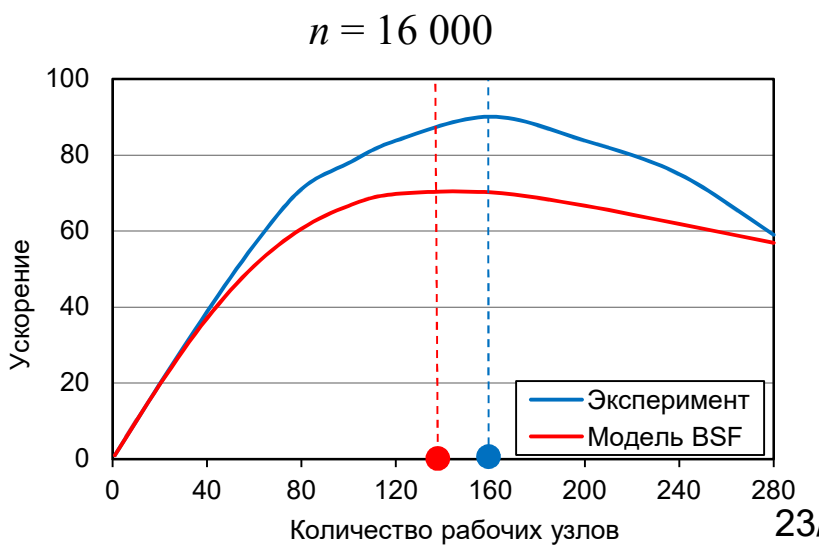
$$\boldsymbol{K_{Jacobi}^{Max} = O\left(\sqrt{n}\right)}$$

Ускорение алгоритма Jacobi-BSF: теория и практика

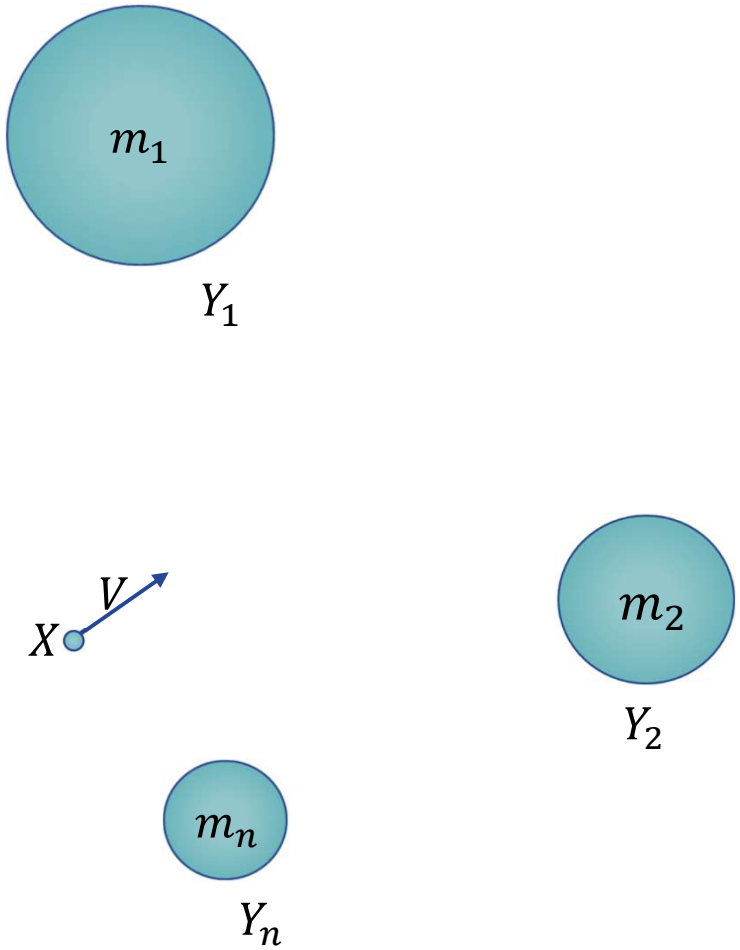






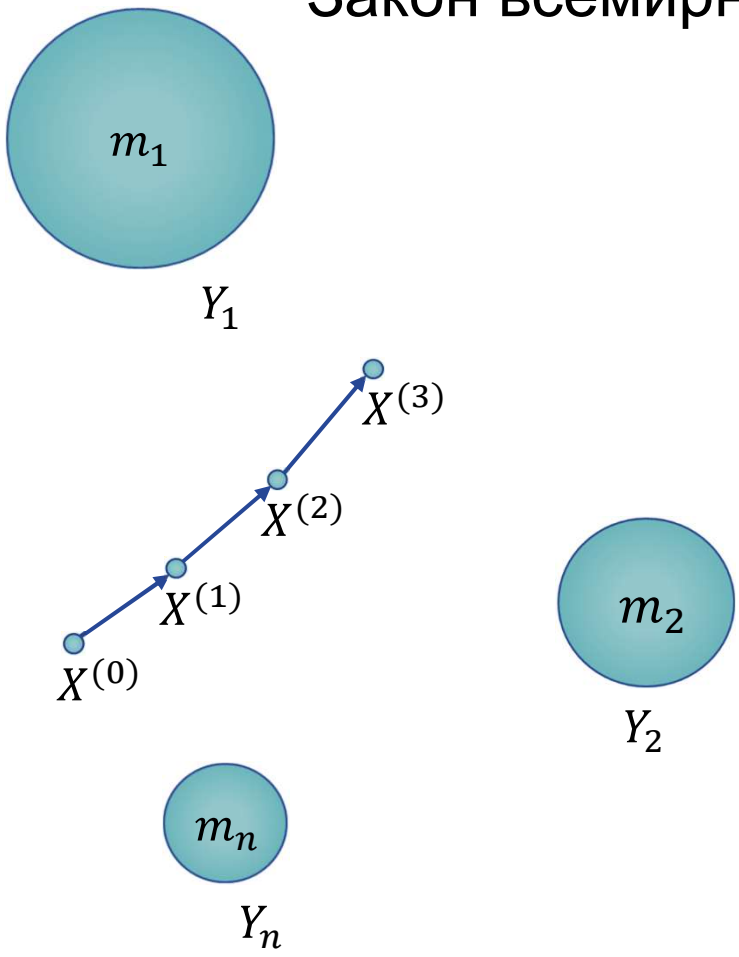


Гравитационная задача



Материальная точка X с малой массой m_x движется с начальной скоростью V среди n неподвижных тел $Y_1, Y_2, \ldots, Y_n$ с большими массами $m_1, m_2, \ldots, m_n$.

Необходимо рассчитать траекторию точки X с шагом Δt



Закон всемирного тяготения: $F_i = G\frac{m_i m_x}{\|Y_i - X\|^3}(Y_i - X)$

Второй закон Ньютона: $A_i = \frac{F_i}{m_x}$

$$\alpha_X(Y_i) = G\frac{m_i}{\|Y_i - X\|^3}(Y_i - X)$$

$$A_i = \alpha_X(Y_i)$$

$$A = \sum_{i=1}^{n} A_i$$

$$V^{(k+1)} = V^{(k)} + A \cdot \Delta t$$

$$X^{(k+1)} = X^{(k)} + V^{(k+1)} \cdot \Delta t$$

Алгоритм решения гравитационной задачи по шаблону BSF

(*T* – конечный момент времени, Δt - шаг по времени)

1. $t := t_0; X^{(0)} := 0; V^{(0)} := 0;\ Input([(Y_1, m_1), \dots, (Y_n, m_n)])$
2. $[A_1, \dots, A_n] := Map(\alpha_X, [(Y_1, m_1), \dots, (Y_n, m_n)])$
3. $A := Reduce(+, [A_1, \dots, A_n])$
4. $V^{(t+\Delta t)} := V^{(t)} + A \cdot \Delta t$

   $X^{(t+\Delta t)} := X^{(t)} + V^{(t+\Delta t)} \Delta t$
5. Если $t \geq T$, перейти на шаг 7
6. $t := t + \Delta t$; перейти на шаг 2
7. Стоп

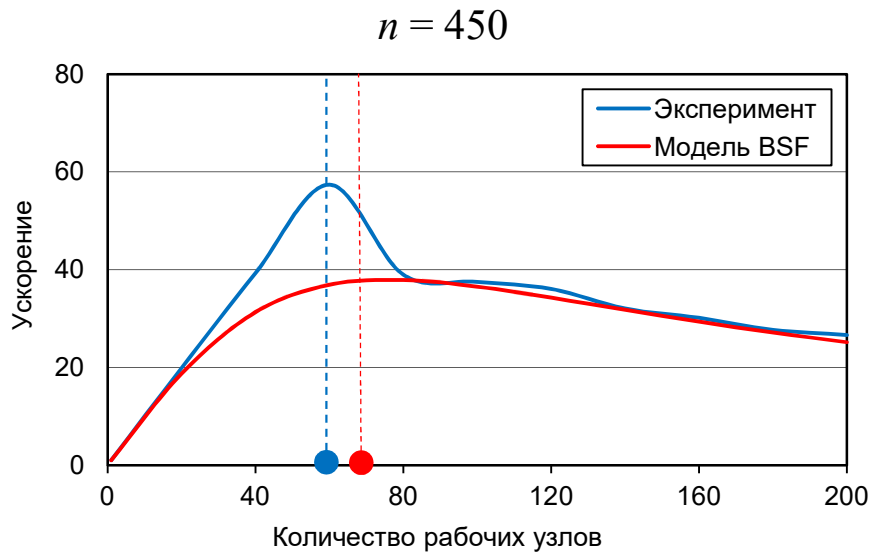
BSF-оценки для алгоритма Gravitation-BSF

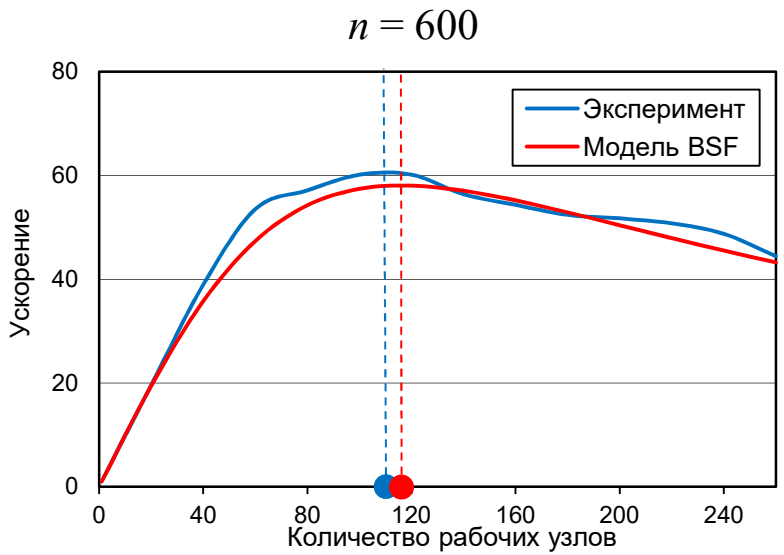
τ_{op} - время выполнения одной операции с плавающей точкой
τ_{tr} - время пересылки вещественного числа (без учета латентности)
n - количество больших тел
L - латентность

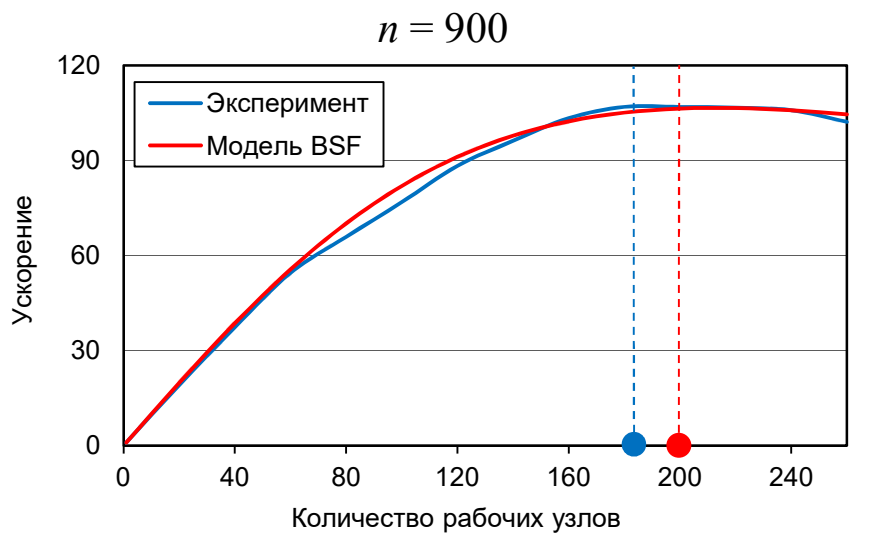
$$\Rightarrow K_{Max} = \sqrt{\frac{20n\tau_{op} + 3l\tau_{op}}{2L + 6\tau_{tr} + 14\tau_{op}}}$$

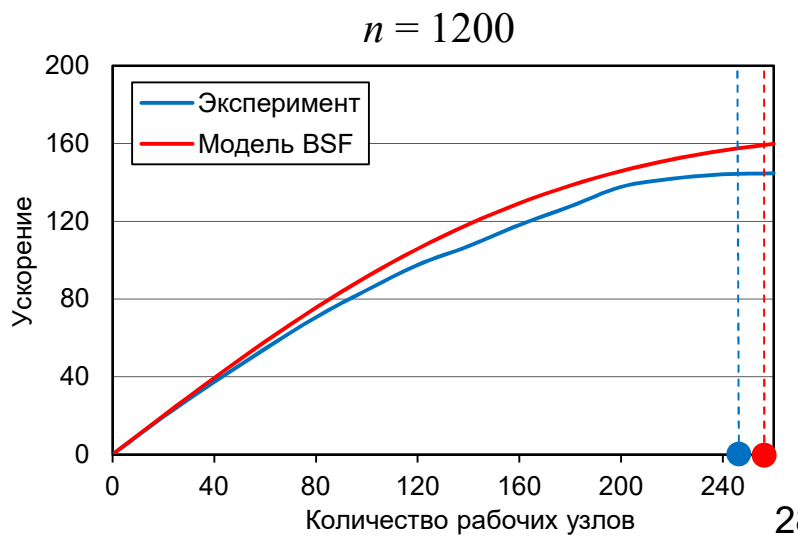
$$K^{Max}_{Gravitation} = O\left(\sqrt{n}\right)$$

Ускорение гравитационного алгоритма: теория и практика









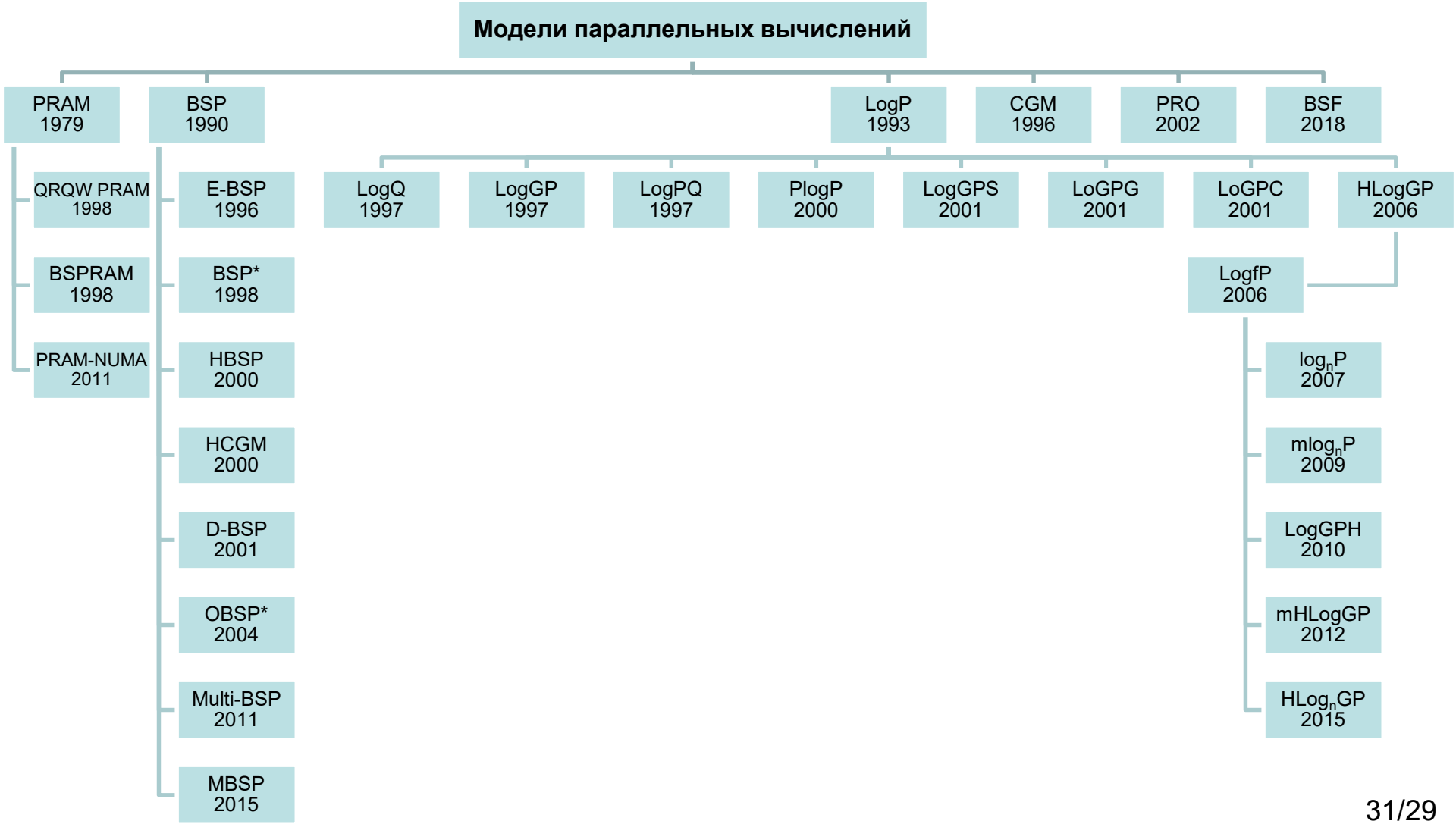
Основные результаты, выносимые на защиту

1. Разработана модель параллельных вычислений BSF, позволяющая получить аналитическую оценку границы масштабируемости алгоритма
2. Разработан компилируемый BSF-каркас на языке C++ с использованием технологий параллельного программирования MPI и OpenMP
3. Выполнены проектирование и реализация визуального конструктора программ BSF-Studio
4. С использованием BSF-каркаса созданы BSF-программы для известных численных итерационных алгоритмов

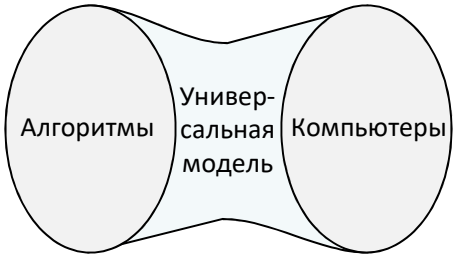
Применения модели BSF

- *Sokolinskaya I., Sokolinsky L.B.* Scalability Evaluation of NSLP Algorithm for Solving Non-Stationary Linear Programming Problems on Cluster Computing Systems // Supercomputing. RuSCDays 2017. Communications in Computer and Information Science. 2017. Vol. 793. P. 40-53. DOI: 10.1007/978-3-319-71255-0_4
- *Sokolinskaya I.M., Sokolinsky L.B.* Scalability Evaluation of Cimmino Algorithm for Solving Linear Inequality Systems on Multiprocessors with Distributed Memory // Supercomputing Frontiers and Innovations. 2018. Vol. 5, No. 2. P. 11-22. DOI: 10.14529/jsfi180202

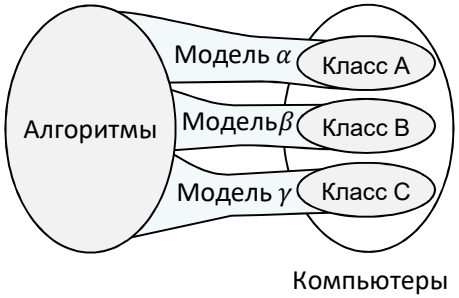
Дерево моделей параллельных вычислений



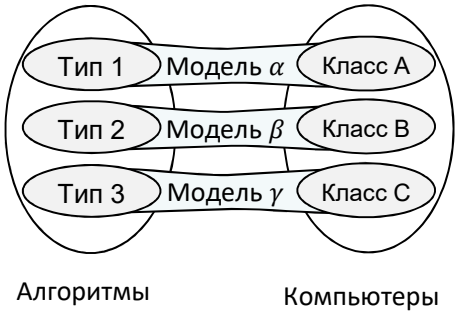
Модель, соединяющая алгоритмы и компьютеры



+ Универсальность
- Невозможно создать ***адекватную*** универсальную модель для современных компьютеров

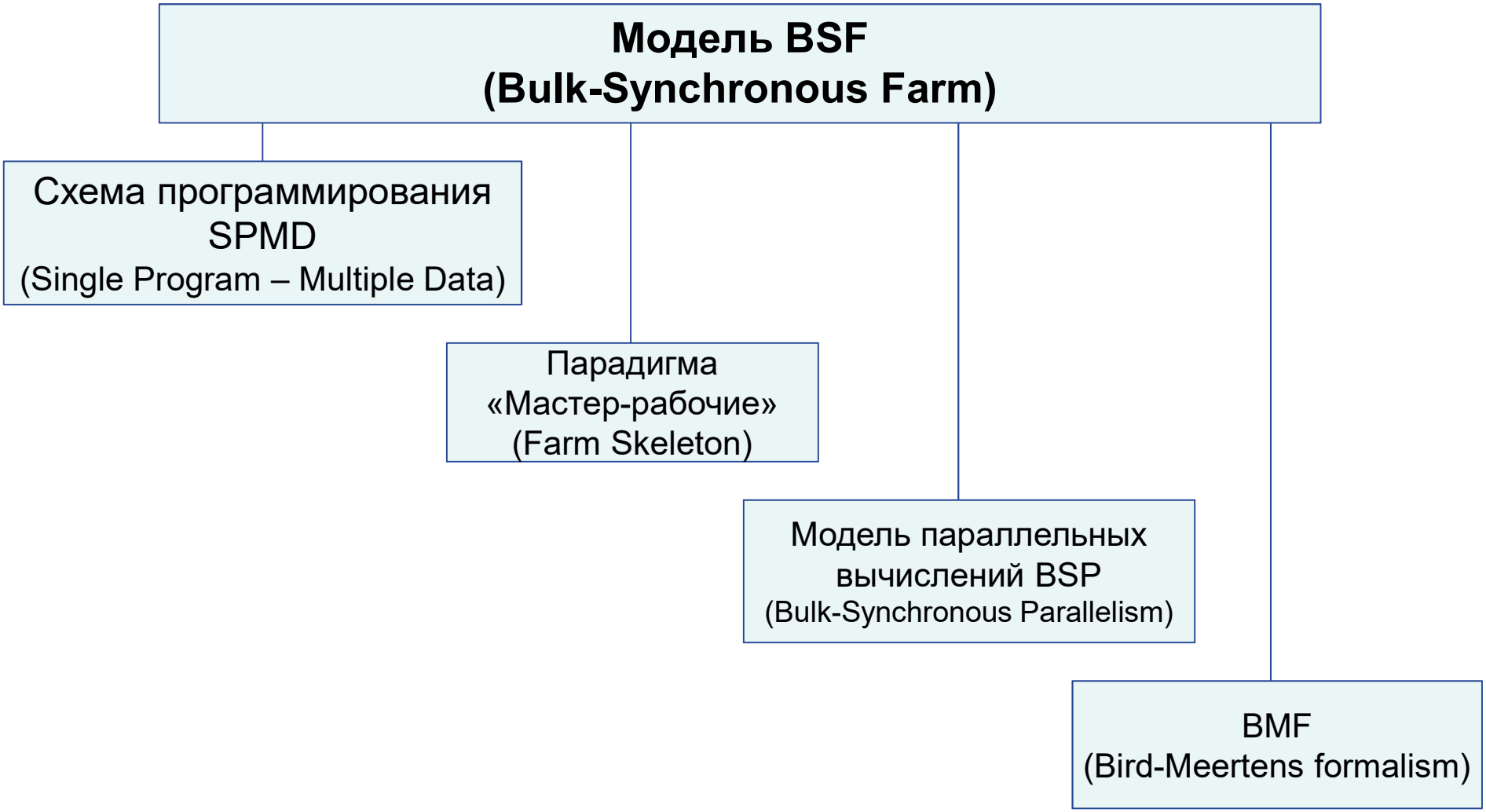


+ Адекватность
± Ограниченная универсальность
- Сложность практического применения



+ Адекватность
+ Простота использования
- Универсальность

Фундамент модели BSF



Модель параллельных вычислений

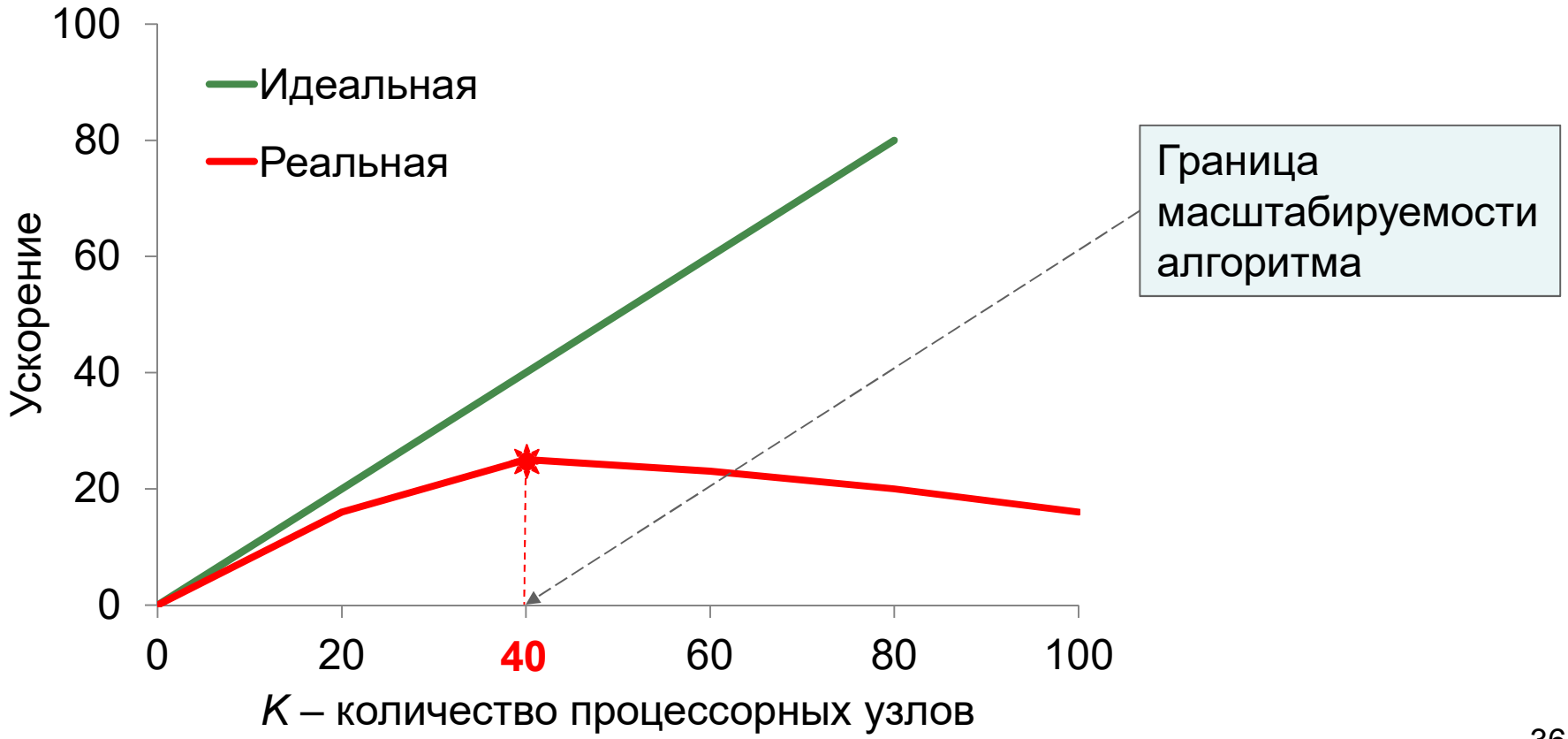
- Модель параллельных вычислений – фреймворк (система правил и ограничений), включающий в себя:
  - абстрактное описание компьютера
  - правила написания алгоритмов и программ
  - способ их параллельного выполнения
  - стоимостную метрику для оценки времени выполнения
- Используется для разработки эффективных алгоритмов и программ
- Позволяет оценить реальную вычислительную сложность алгоритма и прогнозируемое быстродействие программы на различных вычислительных системах

Итерационные алгоритмы с высокой вычислительной сложностью

- Решение систем линейных и нелинейных уравнений [Gonzalez-Gutierrez E., Todorov M.I. A relaxation method for solving systems with infinitely many linear inequalities // Optim. Lett. Springer-Verlag, 2012. Vol. 6, no. 2. P. 291–298.]
- Решение систем линейных и нелинейных неравенств [Censor Y. et al. New Methods for Linear Inequalities // SIAM J. Sci. Comput. Society for Industrial and Applied Mathematics, 2008. Vol. 30, no. 1. P. 473–504]
- Декомпозиция многомерных матриц [Agarwal A., Negahban S., Wainwright M.J. Noisy matrix decomposition via convex relaxation: Optimal rates in high dimensions // Ann. Stat. Institute of Mathematical Statistics, 2012. Vol. 40, no. 2. P. 1171–1197]
- Решение эллиптических дифференциальных уравнений в частных производных [Adsuara J.E. et al. Scheduled Relaxation Jacobi method: Improvements and applications // J. Comput. Phys. Academic Press, 2016. Vol. 321. P. 369–413]
- Решение задач математического программирования и оптимизации [Соколинская И.М., Соколинский Л.Б. Масштабируемый алгоритм для решения нестационарных задач линейного программирования // Вычислительные методы и программирование новые вычислительные технологии. 2018. Vol. 19, no. 4. P. 540–550.]
- Решение задач выпуклой разрешимости [Gubin L.G., Polyak B.T., Raik E.V. The method of projections for finding the common point of convex sets // USSR Comput. Math. Math. Phys. No longer published by Elsevier, 1967. Vol. 7, no. 6. P. 1–24.]

Кривая ускорения – основной индикатор масштабитруемости

$$a(K) = \frac{t_1}{t_K}$$



Ускорение для модели BSF

$$a(K) = \frac{T_1}{T_K}$$

$$a_{BSF}(K) = \frac{2L + t_s + t_r + t_p + t_{Map} + lt_a}{K(2L + t_s + t_r + t_a) + \frac{(t_{Map} + lt_a)}{K} - t_a + t_p}$$

$$K_{Max} = \sqrt{\frac{t_{Map} + lt_a}{2L + t_s + t_r + t_a}}$$

Параметры экспериментов

- $L = 1.5 \cdot 10^{-5}$ сек
  (посылка «точка-точка» одного байта)
- $\tau_{op} = 2.9 \cdot 10^{-8}$ сек
  (в цикле выполнялось 10^6 умножений;
  затраченное время делилось на 10^6)
- $\tau_{tr} = 1.9 \cdot 10^{-7}$ сек
  (пересылалось сообщение длиной 10^6 байт;
  затраченное время делилось на 10^6)

Стоимостные параметры алгоритма Gravitation-BSF

τ_{op} - время выполнения одной операции с плавающей точкой
τ_{tr} - время пересылки вещественного числа (без учета латентности)

$$t_s = 3\tau_{tr}$$

$$t_r = 3\tau_{tr}$$

$$t_{Map} = 20n\tau_{op}$$

$$t_{Reduce} = 3\tau_{op}$$

$$t_p = 14\tau_{op}$$