Динамическое децентрализованное управление распределенными вычислениями для блочно-рекурсивных алгоритмов в компьютерной алгебре.

Ильченко Евгений

Примеры блочно-рекурсивных алгоритмов компьютерной алгебры

1) Блочное матричное умножение:

$$\begin{pmatrix} A_0 & A_1 \\ A_2 & A_3 \end{pmatrix} \times \begin{pmatrix} B_0 & B_1 \\ B_2 & B_3 \end{pmatrix} = \begin{pmatrix} C_0 & C_1 \\ C_2 & C_3 \end{pmatrix}$$

$$\begin{aligned} C_0 &= A_0 \times B_0 + A_1 \times B_2 \\ C_1 &= A_0 \times B_1 + A_1 \times B_3 \\ C_2 &= A_2 \times B_0 + A_3 \times B_2 \\ C_3 &= A_2 \times B_1 + A_3 \times B_3 \end{aligned}$$

2) Умножение матриц по алгоритму Штрассена:

$$\begin{aligned} \mathbf{P}_1 &:= (\mathbf{A}_{1,1} + \mathbf{A}_{2,2})(\mathbf{B}_{1,1} + \mathbf{B}_{2,2}) \\ \mathbf{P}_2 &:= (\mathbf{A}_{2,1} + \mathbf{A}_{2,2})\mathbf{B}_{1,1} \\ \mathbf{P}_3 &:= \mathbf{A}_{1,1}(\mathbf{B}_{1,2} - \mathbf{B}_{2,2}) \\ \mathbf{P}_4 &:= \mathbf{A}_{2,2}(\mathbf{B}_{2,1} - \mathbf{B}_{1,1}) \\ \mathbf{P}_5 &:= (\mathbf{A}_{1,1} + \mathbf{A}_{1,2})\mathbf{B}_{2,2} \\ \mathbf{P}_6 &:= (\mathbf{A}_{2,1} - \mathbf{A}_{1,1})(\mathbf{B}_{1,1} + \mathbf{B}_{1,2}) \\ \mathbf{P}_7 &:= (\mathbf{A}_{1,2} - \mathbf{A}_{2,2})(\mathbf{B}_{2,1} + \mathbf{B}_{2,2}) \end{aligned}$$

$$\begin{aligned} \mathbf{C}_{1,1} &= \mathbf{P}_1 + \mathbf{P}_4 - \mathbf{P}_5 + \mathbf{P}_7 \\ \mathbf{C}_{1,2} &= \mathbf{P}_3 + \mathbf{P}_5 \\ \mathbf{C}_{2,1} &= \mathbf{P}_2 + \mathbf{P}_4 \\ \mathbf{C}_{2,2} &= \mathbf{P}_1 - \mathbf{P}_2 + \mathbf{P}_3 + \mathbf{P}_6 \end{aligned}$$

3) Обращение матриц по алгоритму Шура-Штрассена:

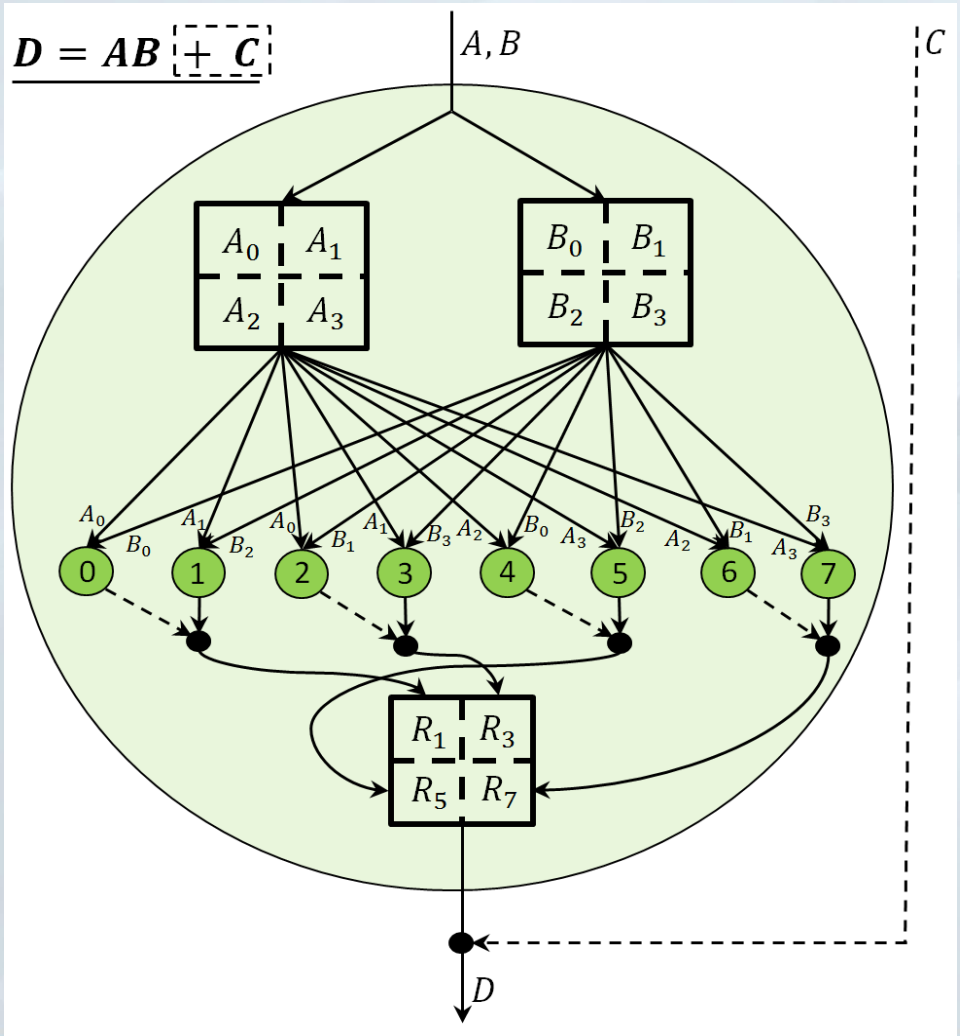
$$A = \begin{pmatrix} A_0 & A_1 \\ A_2 & A_3 \end{pmatrix}$$

$$A^{-1} = \begin{pmatrix} \boldsymbol{I} & -A_0^{-1}A_1 \\ 0 & \boldsymbol{I} \end{pmatrix} \times \begin{pmatrix} \boldsymbol{I} & 0 \\ 0 & (A_3 - A_2A_0^{-1}A_1)^{-1} \end{pmatrix} \times \begin{pmatrix} \boldsymbol{I} & 0 \\ -A_2 & \boldsymbol{I} \end{pmatrix} \times \begin{pmatrix} A_0^{-1} & 0 \\ 0 & \boldsymbol{I} \end{pmatrix}$$

4) Дихотомическое умножение полиномов.

5) Умножение полиномов алгоритмом Карацубы.

Блочный рекурсивный алгоритм умножения двух матриц



(0,2,4,6) $\underbrace{A \cdot B}_{\text{исходные данные}}$

(1,3,5,7) $\underbrace{A \cdot B}_{\text{исходные данные}} + \underbrace{C}_{\text{пост-данные}}$

Информационные зависимости данных

Информационные зависимости постданных

Параллельная реализация блочного рекурсивного алгоритма

Два основных подхода к реализации блочно-рекурсивных алгоритмов:

- статическая реализация;
- динамическая реализация.

Статическая реализация предполагает явное отображение всех подзадач, возникающих при рекурсивном разбиении, на вычислительные модули (узлы вычислительного кластера, MPI-процессы, ядра многоядерного процессора) ***до начала работы*** программы. В процессе вычисления никаких изменений не происходит.

Динамическая реализация подразумевает отображение подзадач на вычислительные модули ***в процессе работы*** параллельной программы. Задается только вычислительный модуль *(root_node)*, на котором происходит инициализация исходных данных. Всё дальнейшее распределение нагрузки будет осуществляться только во время работы программы.

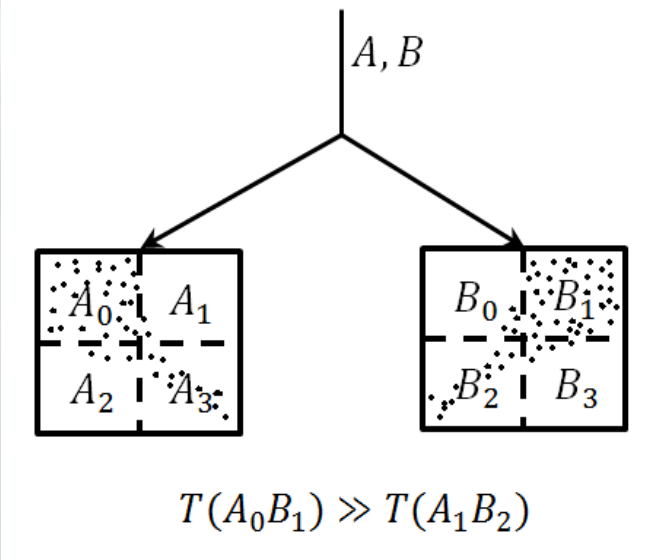
Плюсы и минусы *статической* параллельной реализации блочных рекурсивных алгоритмов

Положительные стороны статического подхода:

- отсутствие дополнительных расходов на перестройку вычислительного процесса;
- простота реализации.

Главный минус статического подхода к распараллеливанию алгоритмов:

- требование к однородности входных данных.



Плюсы и минусы *динамической* параллельной реализации блочных рекурсивных алгоритмов

Положительные стороны динамического распараллеливания:

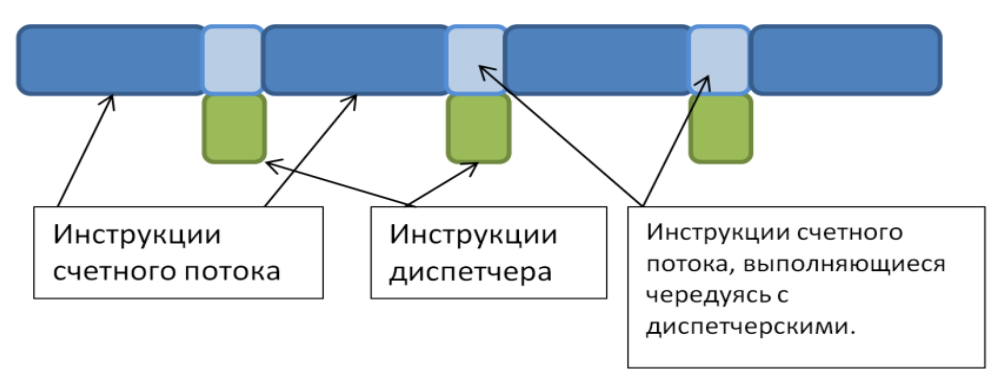
- отсутствие проблемы перепрограммирования при изменении числа вычислительных модулей или размера входных данных для алгоритма;
- правильная динамическая реализация обеспечивает загрузку всех вычислительных модулей и не допускает их простоя при работе программы, даже в случае неоднородных данных.

Минус динамического распараллеливания:

- требуются дополнительные расходы на функционирование диспетчерского потока выполнения инструкций программы.

Динамическая реализация параллельного алгоритма с несколькими диспетчерами

Реализация параллельного алгоритма с одним диспетчером и одним счетных потоком на каждом MPI-процессе:

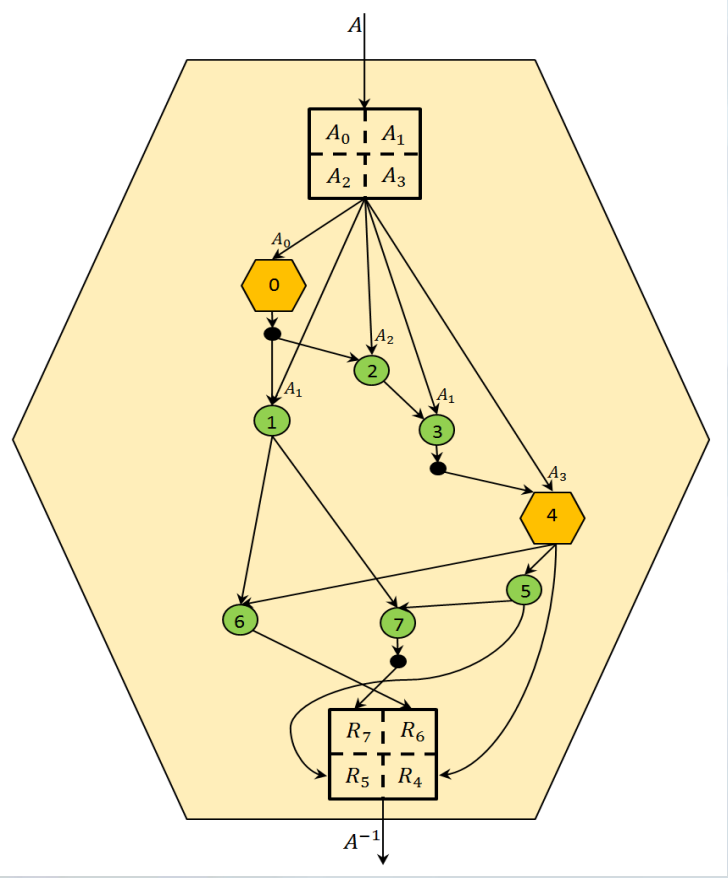


Реализация параллельного алгоритма с одним диспетчером для каждого MPI-процессе и количеством счетных потоков равным *m-1*, где *m* – количество ядер, имеющихся на вычислительном узле супер-ЭВМ:

D	P	P	P	P	P	P	P
P	P	P	P	P	P	P	P
P	P	P	P	P	P	P	P
P	P	P	P	P	P	P	P
P	P	P	P	P	P	P	P
P	P	P	P	P	P	P	P
P	P	P	P	P	P	P	P
P	P	P	P	P	P	P	P

Граф алгоритма нахождения обратной матрицы по методу Шура-Штрассена

$$A^{-1} = \begin{pmatrix} A_0^{-1} + A_0^{-1}A_1(A_3 - A_2A_0^{-1}A_1)^{-1}A_2A_0^{-1} & -A_0^{-1}A_1(A_3 - A_2A_0^{-1}A_1)^{-1} \\ -(A_3 - A_2A_0^{-1}A_1)^{-1}A_2A_0^{-1} & (A_3 - A_2A_0^{-1}A_1)^{-1} \end{pmatrix}$$



- матричное обращение

- матричное умножение

Используемая терминология

Блочно-рекурсивные алгебраические алгоритмы будем представлять в виде направленных ациклических графов, вершинам которых будут сопоставляться некоторые вычисления (формулы), а дуги будут обозначать информационные зависимости.

Вершины имеют два типа:

- **«тяжелые»** – вершины, которые *могут* быть посчитаны параллельно (разными MPI-процессами);
- **«легкие»** – вершины, которые всегда будут считаться в последовательном однопоточном режиме на том MPI-процессе, где был инициализирован граф, к которому принадлежит рассматриваемая вершина.

Любая вершина имеет:

- ***исходные данные*** – входные данные для алгоритма, представляющего данную вершину;
- ***результат*** – выходные данные алгоритма, представляющего данную вершину.

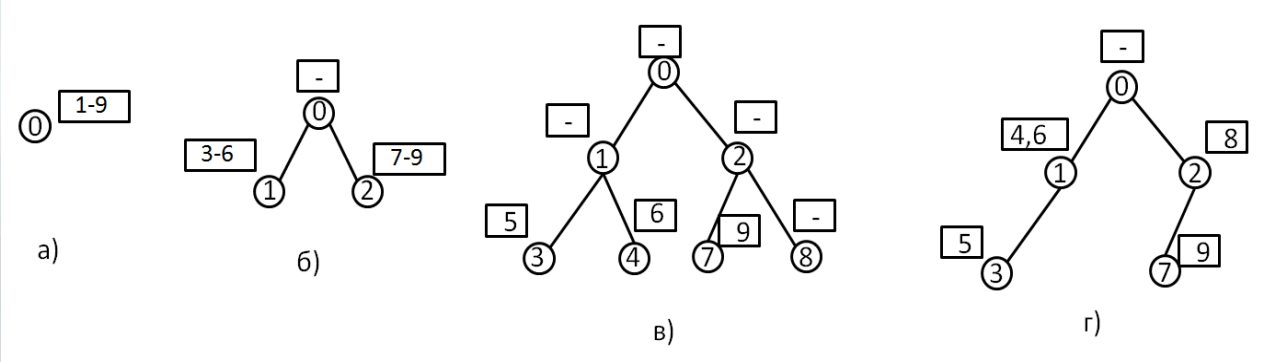
Тяжелые вершины *в общем случае* имеют четыре процедуры обработки:

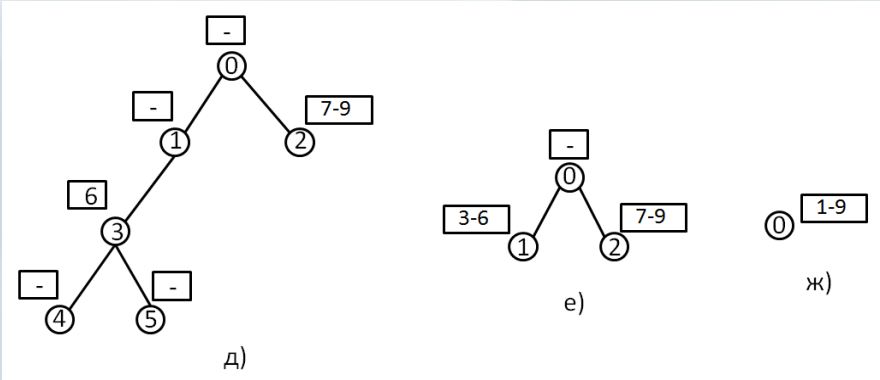
- ***инициализация*** – подготовка исходных данных вершины;
- ***счет*** – параллельное выполнение алгоритма для этой вершины;
- ***сборка графа*** – процедура получения выходных данных рассматриваемого графа из результатов вершин этого графа;
- ***постобработка результата*** – некоторое преобразование выходных данных, выполняемое последовательно в однопоточном режиме на том MPI-процессе, где велся счет рассматриваемого графа.

Легкие вершины *всегда* имеют две процедуры обработки:

- ***инициализация*** – подготовка исходных данных вершины;
- ***счет*** – последовательный однопоточный счет вершины.

Пример организации параллельных вычислений при использовании DDP-схемы





Описание работы счетных потоков DDP (*dynamic decentralized parallelization*)

Задания, которые выполняют счетные потоки:

1) *выполнить сериализацию данных задачи;*
2) *выполнить сериализацию данных для постобработки некоторой тяжелой вершины;*
3) *выполнить сериализацию результата задачи;*
4) *выполнить десериализацию данных задачи;*
5) *выполнить десериализацию данных для постобработки некоторой тяжелой вершины;*
6) *выполнить десериализацию результата задачи;*
7) *выполнить инициализацию некоторой тяжелой вершины графа задачи;*
8) *выполнить счет некоторой тяжелой вершины графа задачи;*
9) *выполнить счет некоторой легкой вершины графа задачи;*
10) *выполнить постобработку некоторой тяжелой вершины графа задачи;*
11) *выполнить завершающую сборку некоторого графа задачи;*
12) *выполнить счет кустовой задачи (в многопоточном режиме).*

Блочный алгоритм нахождения расширенной присоединенной матрицы [3]

$$A_{ext}(M, d_0) = (\mathbf{A}, \mathbf{S}, \mathbf{E}, \mathbf{d}), \quad M \in R^{n\times n},\ n = 2^k.$$

Если $M = 0 : A_{ext}(0, d_0) = (d_0\boldsymbol{I}, 0, 0, d_0)$, где $\boldsymbol{I}$ - единичная матрица.

Если $k = 0$ и $M = a \neq 0$: $A_{ext}(a, d_0) = (d_0, a, 1, a)$.

Если $k > 0$ и $M \neq 0$ разобьем матрицы M, A, S, E на четыре равных блока:

$$M = \begin{pmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{pmatrix}, A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}, S = \begin{pmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{pmatrix}, E = \begin{pmatrix} E_{11} & E_{12} \\ E_{21} & E_{22} \end{pmatrix}.$$

Введем обозначения: $I_{ij} = E_{ij}E_{ij}^T, \bar{I}_{ij} = \boldsymbol{I} - I_{ij}, Y_{ij} = E_{ij}^T S_{ij} - d_{ij}\boldsymbol{I}, \quad i, j \in 1, 2.$

$$\mathbf{A_{ext}}(\mathbf{M_{11}}, \mathbf{d_0}) = (\mathbf{A_{11}}, \mathbf{S_{11}}, \mathbf{E_{11}}, \mathbf{d_{11}}),$$

$$M_{12}^1 = \frac{A_{11}M_{12}}{d_0}, M_{21}^1 = -\frac{M_{21}Y_{11}}{d_0}, M_{22}^1 = \frac{M_{22}d_{11} - M_{21}E_{11}^T M_{12}^1}{d_0},$$

$$\mathbf{A_{ext}}(\bar{\mathbf{I}}_{\mathbf{11}}\mathbf{M_{12}^1}, \mathbf{d_{11}}) = (\mathbf{A_{12}}, \mathbf{S_{12}}, \mathbf{E_{12}}, \mathbf{d_{12}}), \quad \mathbf{A_{ext}}(\mathbf{M_{21}^1}, \mathbf{d_{11}}) = (\mathbf{A_{21}}, \mathbf{S_{21}}, \mathbf{E_{21}}, \mathbf{d_{21}}),$$

$$M_{22}^2 = -\frac{A_{21}M_{22}^1 Y_{12}}{(d_{11})^2}, \ d_s = \frac{d_{21}d_{12}}{d_{11}},$$

$$\mathbf{A_{ext}}(\bar{\mathbf{I}}_{\mathbf{21}}\mathbf{M_{22}^2}, \mathbf{d_s}) = (\mathbf{A_{22}}, \mathbf{S_{22}}, \mathbf{E_{22}}, \mathbf{d_{22}}),$$

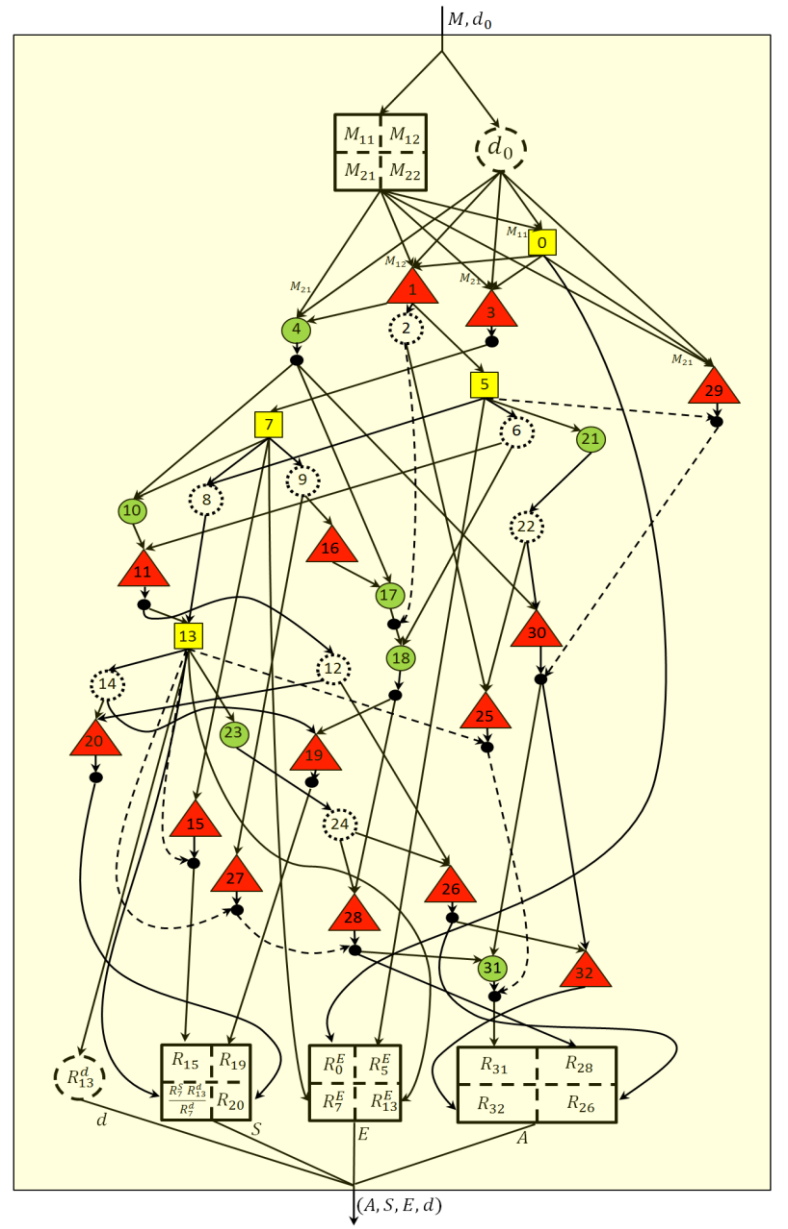
$$M_{11}^2 = -\frac{S_{11}Y_{21}}{d_{11}}, \ M_{12}^2 = \frac{\left(\frac{S_{11}E_{21}^T A_{21}}{d_{11}} M_{22}^1 - I_{11}M_{12}^1 d_{21}}{d_{11}}\right) Y_{12} + S_{12}d_{21}}{d_{11}}, \ M_{12}^3 = -\frac{M_{12}^2 Y_{22}}{d_s},$$

$$M_{22}^3 = S_{22} - \frac{I_{21}M_{22}^2 Y_{22}}{d_s}, \ A^1 = A_{12}A_{11}, \ A^2 = A_{22}A_{21}, \quad L = \left(\frac{A^1 - \frac{I_{11}M_{12}^1 E_{12}^T A^1}{d_{11}}}{d_{11}}\right) d_{22},$$

$$P = \frac{A^2 - \frac{I_{21}M_{22}^2 E_{22}^T A^2}{d_s}}{d_{21}}, F = -\frac{\left(\frac{S_{11}E_{21}^T A_{21}}{d_{11}}\right) d_{22} + \frac{M_{12}^2 E_{22}^T A^2}{d_s}}{d_{21}}, \ G = -\frac{\left(\frac{M_{21}E_{11}^T A_{11}}{d_0}\right) d_{12} + \frac{M_{22}^1 E_{12}^T A^1}{d_{11}}}{d_{11}},$$

$$\mathbf{A} = \begin{pmatrix} \frac{L+FG}{d_{12}} & F \\ \frac{PG}{d_{12}} & P \end{pmatrix}, \mathbf{S} = \begin{pmatrix} \frac{M_{11}^2 d_{22}}{d_{21}} & M_{12}^3 \\ \frac{S_{21}d_{22}}{d_{21}} & M_{22}^3 \end{pmatrix}, \mathbf{E} = \begin{pmatrix} E_{11} & E_{12} \\ E_{21} & E_{22} \end{pmatrix}, \mathbf{d} = d_{22}.$$

Граф алгоритма нахождения расширенной присоединенной матрицы



$A \cdot B$

$\frac{A \cdot B}{d}$

$A_{ext}(M, d_0) = (A, S, E, d)$

«Легкие» вершины

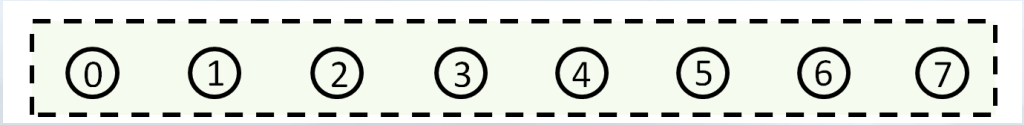
Информационные зависимости данных

Информационные зависимости постданных

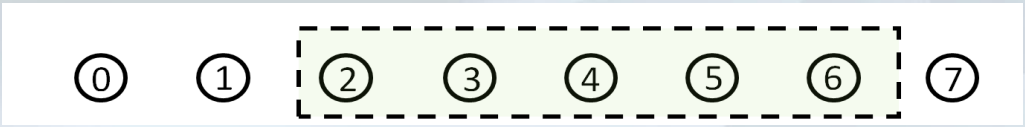
Режимы запуска DDP

Если параллельная программа запущена на N узлах с m вычислительных ядер на каждом, то возможны следующие режимы работы DDP:

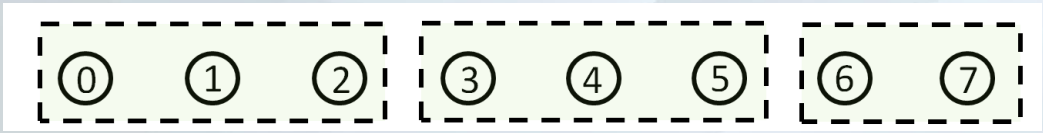
- запуск на всех N узлах:



- запуск на некотором подмножестве имеющихся узлов:



- q одновременных запусков нескольких экземпляров DDP, если N было разбито на q непересекающихся подмножеств:

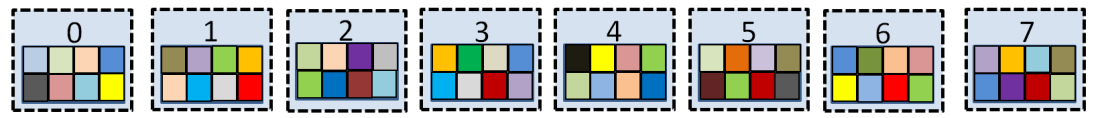


- любая последовательная комбинация описанных выше случаев.

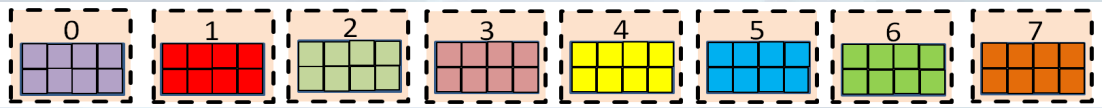
Объединение статического управления и DDP

Параллельные модулярные алгоритмы предполагают независимое вычисление результатов по нескольких модулям (необходимое количество которых может быть неизвестно заранее). Запускается цикл статического управления, который в зависимости от оставшегося количества необработанных модулей выполняет запуск одного из следующих режимов:

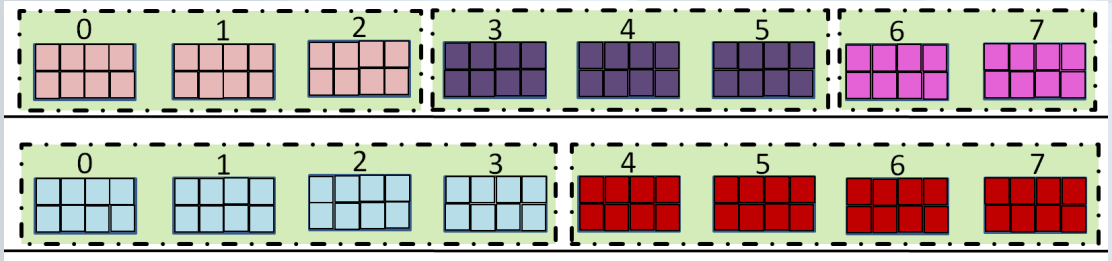
- запуск последовательной версии алгоритма на каждом доступном ядре (будет произведен одновременный счет по $N \cdot m$ модулям):



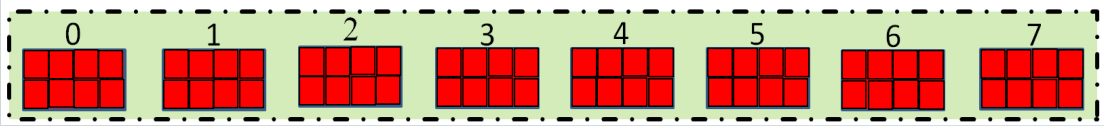
- запуск многопоточной версии алгоритма на каждом вычислительном узле (будет произведен счет по *N* модулям, в случае отсутствия многопоточной версии алгоритма возможен запуск DDP для каждого узла):



- разбиение множества N на q непересекающихся подмножеств, отличающихся мощностью не более чем на единицу и запуск DDP на каждом из этих подмножеств:



- запуск единственного экземпляра DDP на всех доступных узлах:



Основные характеристики программной платформы DDP

1. DDP является фреймворком, позволяющим реализовывать блочно-рекурсивные алгоритмы, выполняя явное отображение графа алгоритма и используемых при его счете формул на заранее определенные абстрактные классы.

2. Реализация выполнена на языке программирования C++, объем исходного кода составил более 10000 строк.

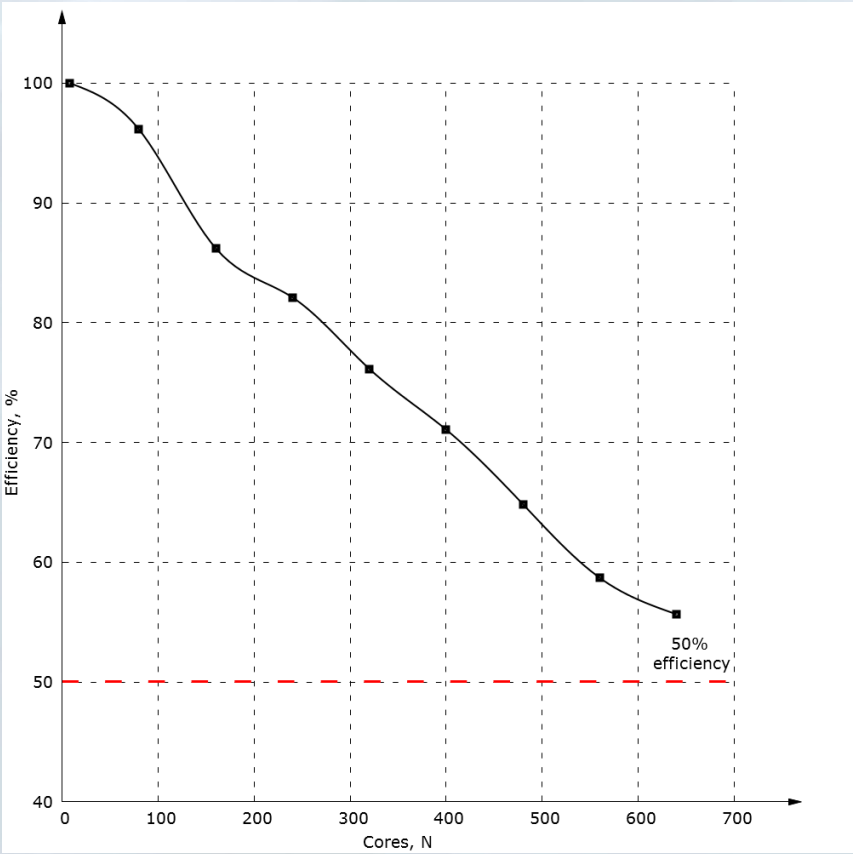
3. DDP позволяет тесную интеграцию с системами компьютерной алгебры в целях получения возможности производить счет крупных задач на супер-ЭВМ.

- C++
- GMP
- Pthreads
- Java (MathPartner)

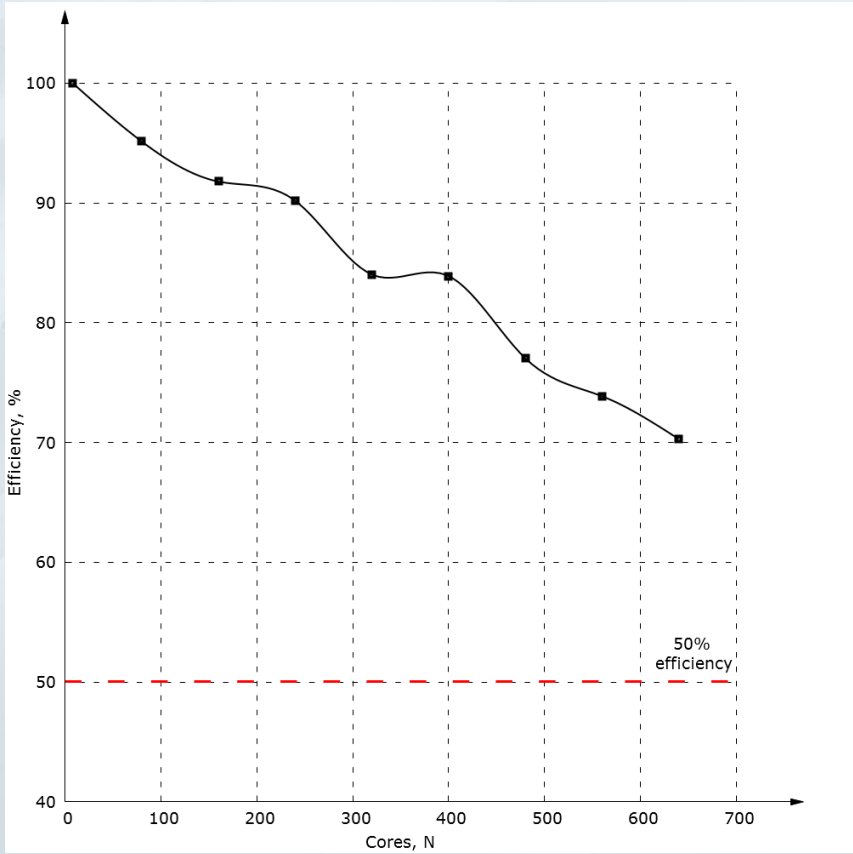
Результаты экспериментов на вычислительном кластере МВС-10П

Графики эффективности параллельной программы, выполняющей умножение двух плотных матриц с 15-битными коэффициентами:

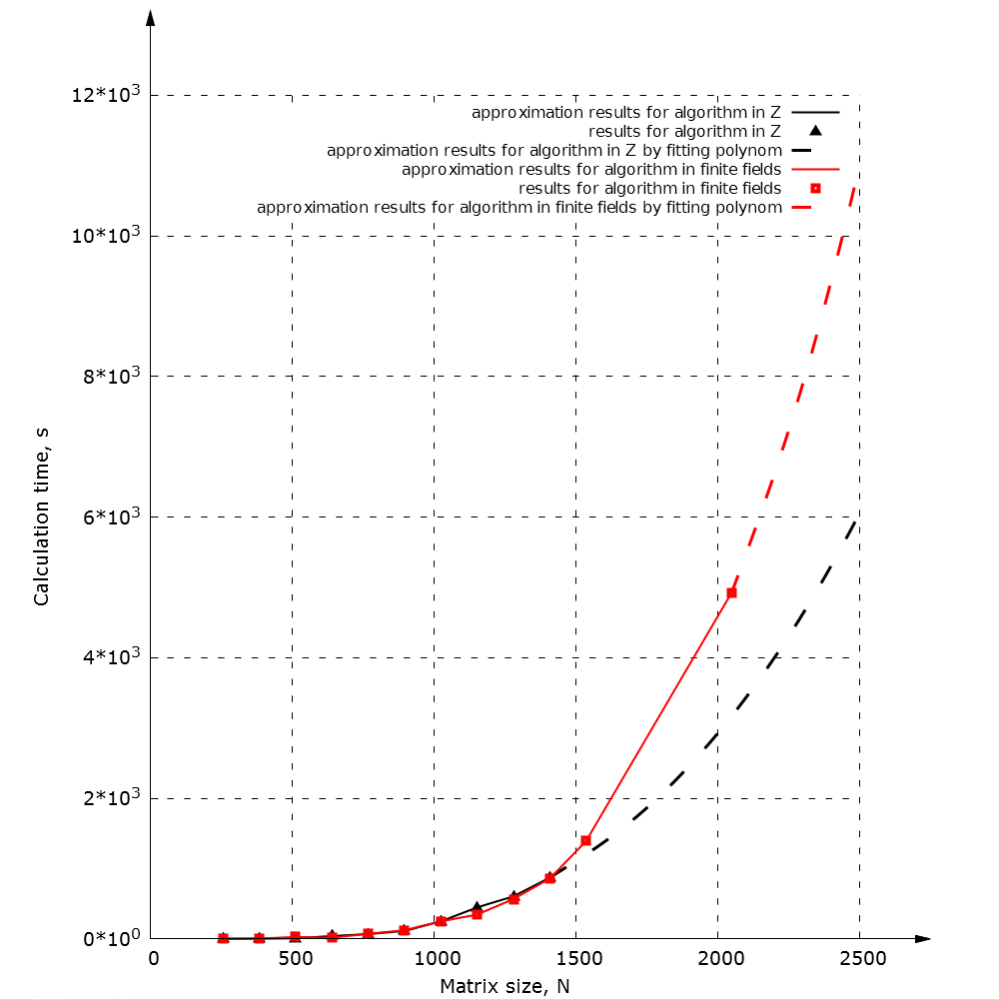
размерность матриц 8192,
7 счетных потоков:



размерность матриц 14336,
7 счетных потоков:



Результаты экспериментов на вычислительном кластере МВС-10П



Сравнение прямого и модулярного параллельных алгоритмов вычисления расширенной присоединенной матрицы. Количество узлов для всех точек равнялось 20, использовалось по 8 ядер на узле. Входные матрицы имели плотность 1% и 15-битные коэффициенты.

Аппроксимация результатов приближающими полиномами, n – размерность входной матрицы:

$$FitZ^a = 4.52 * 10^{-7} n^3 - 1.02 * 10^{-4} n^2 - 0.16n + 46$$

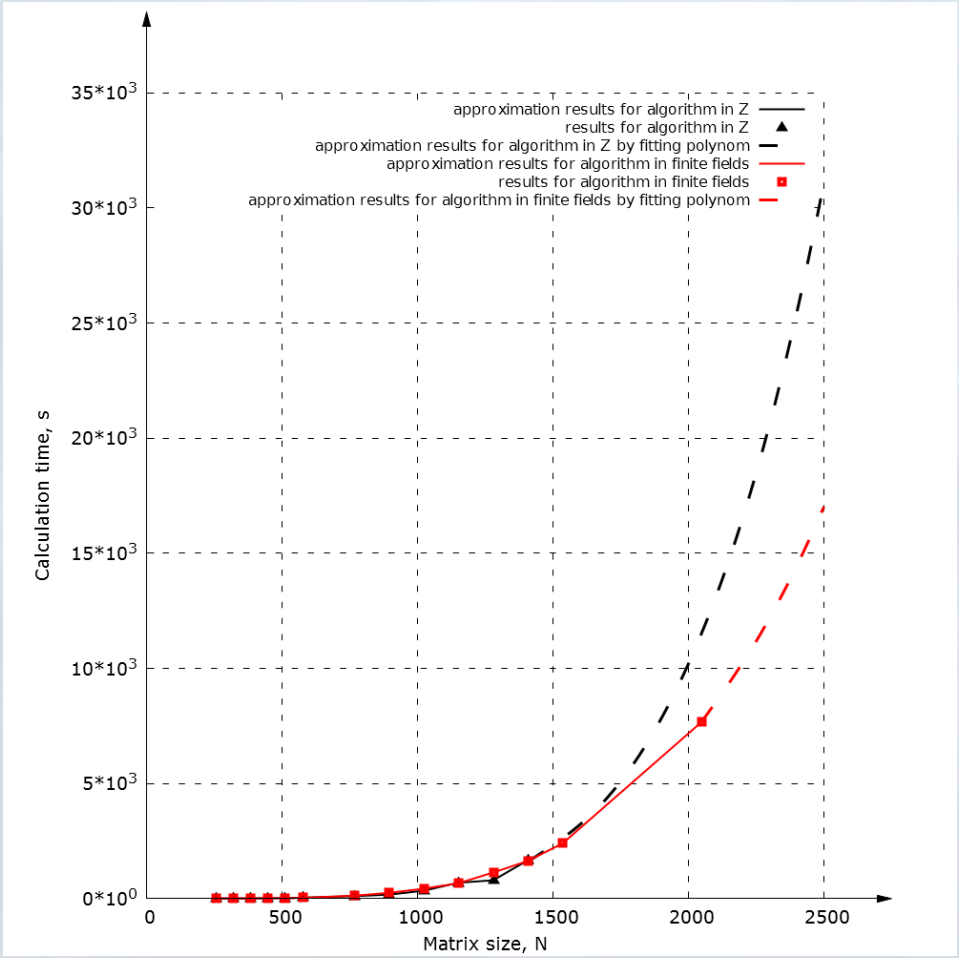
$$FitZp^a = 1.596 * 10^{-6} n^3 - 2.886 * 10^{-3} n^2 + 1.768n - 308$$

Если размер матриц дан в тысячах: $n = 1000 \cdot N$, то

$$FitZ^a = 452N^3 - 102N^2 - 164N + 46$$

$$FitZp^a = 1596N^3 - 2886N^2 + 1768N - 308$$

Результаты экспериментов на вычислительном кластере МВС-10П



Сравнение прямого и модулярного параллельных алгоритмов вычисления расширенной присоединенной матрицы. Количество узлов для всех точек равнялось 20, использовалось по 8 ядер на узле. Входные матрицы имели плотность 100% и 15-битные коэффициенты.

Аппроксимация результатов приближающими полиномами, n – размерность входной матрицы:

$$FitZ^b = 1.837 * 10^{-9}n^4 - 3.514 * 10^{-6}n^3 + 2.487 * 10^{-3}n^2 - 0.519n$$

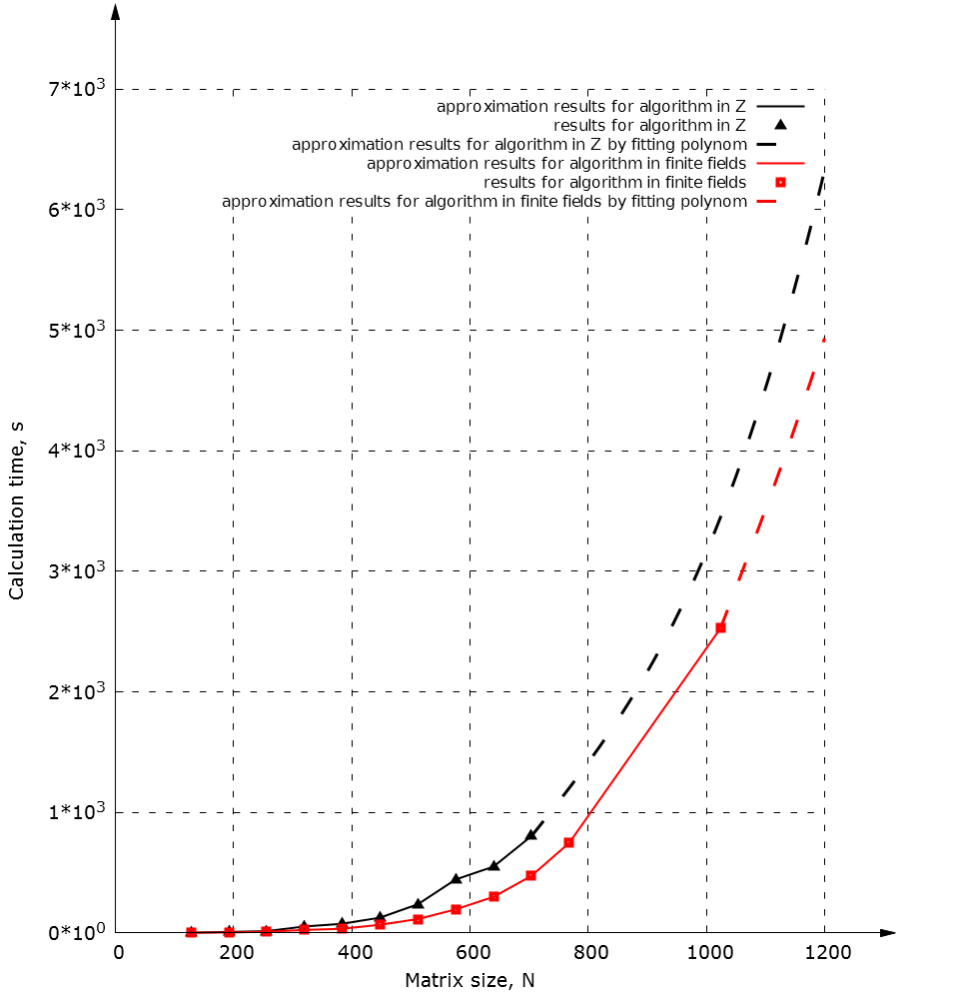
$$FitZp^b = 3.76 * 10^{-10}n^4 + 2.92 * 10^{-7}n^3 - 4.13 * 10^{-4}n^2 + 0.131n$$

Если размер матриц дан в тысячах: $n = 1000 \cdot N$, то

$$FitZ^b = 1837N^4 - 3514N^3 + 2487N^2 - 519N$$

$$FitZp^b = 376N^4 + 292N^3 - 413N^2 + 131N$$

Результаты экспериментов на вычислительном кластере МВС-10П



Сравнение прямого и модулярного параллельных алгоритмов вычисления расширенной присоединенной матрицы. Количество узлов для всех точек равнялось 20, использовалось по 8 ядер на узле. Входные матрицы имели плотность 100% и 100-битные коэффициенты.

Аппроксимация результатов приближающими полиномами, n – размерность входной матрицы:

$$FitZ^c = 2.439 * 10^{-9} n^4 + 1.083 * 10^{-6} n^3 - 4.36 * 10^{-4} n^2 + 6.26 * 10^{-2} n$$

$$FitZp^c = 1.078 * 10^{-9} n^4 + 3.425 * 10^{-6} n^3 - 2.819 * 10^{-3} n^2 + 0.61 n$$

Если размер матриц дан в тысячах: $n = 1000 \cdot N$, то

$$FitZ^c = 2439 N^4 + 1083 N^3 - 436 N^2 + 63 N$$

$$FitZp^c = 1078 N^4 + 3425 N^3 - 2819 N^2 + 606 N$$

Результаты

1. Разработаны алгоритмы динамического децентрализованного управления распределенными вычислениями.
2. Разработана программная платформа DDP для создания эффективных масштабируемых параллельных программ, реализующих блочно-рекурсивные алгоритмы компьютерной алгебры.
3. Разработаны алгоритмы и параллельные программы, выполняющие блочно-рекурсивное матричное умножение, умножение матриц по алгоритму Штрассена, обращение матриц методом Шура-Штрассена, нахождение ядра оператора и обратной матрицы. Сделан сравнительный анализ параллельных программ, выполняющих вычисления блочно-рекурсивным методом с целочисленными операциями и методом гомоморфных образов с отображением в конечные поля и восстановлением в рациональных числах.
4. Осуществлена алгоритмическая и программная интеграция платформы DDP и системы компьютерной алгебры MathPartner.

Публикация результатов работы

Публикации в журналах, входящих в перечень ВАК:

1. Ильченко Е.А. *Об одном алгоритме управления параллельными вычислениями с децентрализованным управлением* // Вестник Тамбовского университета. Серия: Естественные и технические науки, 2013, том 18, вып 4, с. 1198-1206.

2. Ильченко Е.А. *Сокетный сервер для удаленного взаимодействия пользователя с системой управления распределенными вычислениями* // Вестник Тамбовского университета. Серия: Естественные и технические науки, 2014, том 19, вып 2, с. 551-557.

3. Ильченко Е.А. *Об эффективном методе распараллеливания блочных рекурсивных алгоритмов* // Вестник Тамбовского университета. Серия: Естественные и технические науки, 2015, том 20, вып 5, с. 1173-1186.

4. Ильченко Е.А. *Инструменты математического сервиса «MathPartner» для выполнения параллельных вычислений на кластере* // Труды ИСП РАН, 2016, том 1, вып. 3, с. 173-188.

Публикации в изданиях, не входящих в перечень ВАК:

1. Gennadi Malaschonok and Evgeni Ilchenko. *Decentralized control of parallel computing* // International conference Polynomial Computer Algebra. St.Petersburg, PDMI RAS, 2012. P. 57-58.

2. Ilchenko E.A. *One approach to algorithm of the decentralized management for the distributed computing process* // International conference MathParCA-2015, Turku-Abo, Finland, 2015. P. 8.

3. Evgeny A. Ilchenko. *About Effective Methods of Parallelizing Block Recursive Algorithms* // International Conference on Mathematical Partnership, Parallel Computing and Computer Algebra: MathParCA-2016, Loutra, Agia Paraskevi, Greece, August, 5 - 15, 2016. P.51-52.

4. Е. А. Ильченко. *О распараллеливании рекурсивных блочных алгебраических алгоритмов: алгоритмы и эксперименты* // Компьютерная алгебра. Материалы международной конференции. Москва, 29 июня - 2 июля 2016, ВЦ РАН им. А.А.Дородницына. Москва: 2016. С. 60-62.

5. Е.А. Ильченко, М.А. Рыбаков, С.А. Хворов, Г.И. Малашонок. *Параллельное программирование на OpenMPI Java с приложением в Math Partner. Учеб. пособие.* ФГБОУ ВПО «Тамбовский государственный университет имени Г.Р. Державина». Тамбов: Изд. дом ТГУ им. Г.Р. Державина, 2016.

Литература

1. Малашонок Г.И. *Управление параллельным вычислительным процессом* // Вестник Тамбовского университета. Сер. Естественные и технические науки. Тамбов, 2009. Том 14. Вып. 1. С. 269-274.
2. Малашонок Г.И. *Матричные методы вычислений в коммутативных кольцах*. Тамбов: Изд-во Тамбовского университета, 2002.
3. Малашонок Г.И. *О вычислении ядра оператора действующего в модуле* // Вестник Тамбовского университета. Сер. Естественные и технические науки. Тамбов, 2008. Том 13, вып. 1. С. 129-131
4. Malaschonok G.I. *Effective Matrix Methods in Commutative Domains* // Formal Power Series and Algebraic Combinatorics. Berlin: Springer, 2000. P.506-517.
5. Strassen V. Gaussian Elimination is not optimal // Numerische Mathematik. 1969, 13, 354-356.
6. Jean-Guillaume Dumasa, Thierry Gautierb, Clement Pernet, Jean-Louis Rochb, Ziad Sultana. *Recursion based parallelization of exact dense linear algebra routines for Gaussian elimination* // Parallel Computing, Volume 57, September 2016, Pages 235-249.
7. A. R. Benson and G. Ballard. *A framework for practical parallel fast matrix multiplication* // In Proceedings of the 20th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, PPoPP 2015, pages 42-53, New York, NY, USA, 2015. ACM.
8. P. D'alberto, M. Bodrato, and A. Nicolau. *Exploiting parallelism in matrix-computation kernels for symmetric multiprocessor systems: Matrix-multiplication and matrix-addition algorithm optimizations by software pipelining and threads allocation* // ACM Trans. Math. Softw., 38(1):2:1-2:30, Dec. 2011.
9. J.-G. Dumas, P. Giorgi, and C. Pernet. *Dense linear algebra over word-size prime fields: the fflas and ffpack packages* // ACM Trans. on Mathematical Software(TOMS), 35(3):1-42, 2008.
10. Бетин А.А. *Эксперименты с параллельным алгоритмом вычисления присоединённой матрицы и параллельным умножением файловых матриц* // Вестник Тамбовского университета. Сер. Естественные и технические науки. Тамбов, 2010, том 15, вып. 1, с. 341-345.
11. Бетин А.А. *Эксперименты с параллельным алгоритмом вычисления присоединённой матрицы* // Вестник Тамбовского университета. Сер. Естественные и технические науки. Тамбов, 2010. Том 15. Вып. 6. С. 1748-1754.

Литература

12. Малашонок Г.И., Валеев Ю.Д. *Организация параллельных вычислений в рекурсивных символьно-численных алгоритмах* // Труды конференции ПаВТ'2008 (Санкт-Петербург). Челябинск: Изд-во ЮУрГУ, 2008. С. 153-165.
13. Г.И. Малашонок, Ю.Д. Валеев. *Рекурсивное распараллеливание символьно-численных алгоритмов* Вестник Тамбовского университета. Сер. Естественные и технические науки. Тамбов, 2006. Том 11. Вып. 4. С. 536-549
14. Малашонок Г.И., Валеев Ю.Д. *Динамическое распределение заданий в вычислительном кластере по графу алгоритма* // XI Державинские чтения. Тезисы докладов. Тамбов: Изд-во ТГУ им. Г.Р. Державина, 2006. С. 38-47.
15. Г.И. Малашонок, Ю.Д. Валеев. О некоторых подходах к построению параллельных программ // Вестник тамбовского университета. Тамбов, 2005. Том~10. Вып. 1. С.154-156.

Спасибо за внимание!