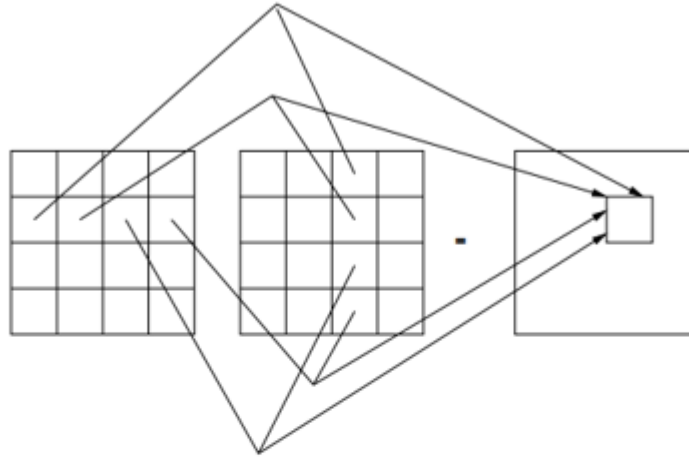


Структура программы

В программе реализован двухшаговый алгоритм умножения матриц с использованием блоков.



1й шаг

map: ('A', i, j, A_{ij}) → [((l, J, K), ('A', i, j, A_{ij}) для K=1,...,g]

('B', j, k, B_{jk}) → [((l, J, K), ('B', j, k, B_{jk}) для l=1,...,g]

reduce: $x_{lk} = \sum_{j \in J} A_{ij} B_{jk}$
emit ((i, k), x_{lk})

2й шаг

Сумма в reducer

Обработка файлов, содержащих размеры матриц, происходит в main. Программа считывает размер матрицы и, используя значение mmGroups, вычисляет размер блока. Размер матрицы и размер блока записываются в конфигурацию, таким образом, эти два параметра доступны в каждом Mapper и Reducer, как и остальные параметры конфигурации. Для определённости в отчёте левая матрица в умножении будет иметь имя 'A', правая - 'B'.

Job 1

mmMapper

Mapper первого шага

- получает на вход четвёрку элементов (name, i, j, e[i,j]), где name - имя матрицы, e[i,j] - элемент, находящийся на пересечении i-ой строки и j-ого столбца матрицы name;

- используя размер блока, вычисляет номера блоков для текущего элемента матрицы;
- если подано значение из левой (первой) матрицы, то возвращает пары <ключ, значение> вида $\langle (I, J, K), ('A', i, j, a[i,j]) \rangle$ для $K = 1 \dots \text{mmGroups}$,
если подано значение из правой (второй) матрицы, то возвращает пары $\langle (I, J, K), ('B', i, j, b[i,j]) \rangle$ для $I = 1 \dots \text{mmGroups}$.

MultReducer

Каждый вызов `reduce` на первом шаге обрабатывает значения $(name, i, j, e[i,j])$, относящиеся к определенному значению тройки (I, J, K) . Его задача перемножить полученные блоки исходных матриц. Итак, Reducer первого шага

- итерируется по значениям и восстанавливает блоки в матрицы соответствующего размера;
- перемножает блоки (каждый элемент матрицы-результата умножения блоков является одним из слагаемых в соответствующей ячейке матрицы-результата умножения исходных матриц);
- используя номера групп (I, K) , размер группы и расположение элемента внутри блока, вычисляет индексы (i_global, k_global) ячейки слагаемого `res` в результирующей матрице;
- возвращает пару <ключ, значение> вида $\langle (i_global, k_global), res \rangle$, если `res` отлично от нуля.

Job 2

IdentityMapper

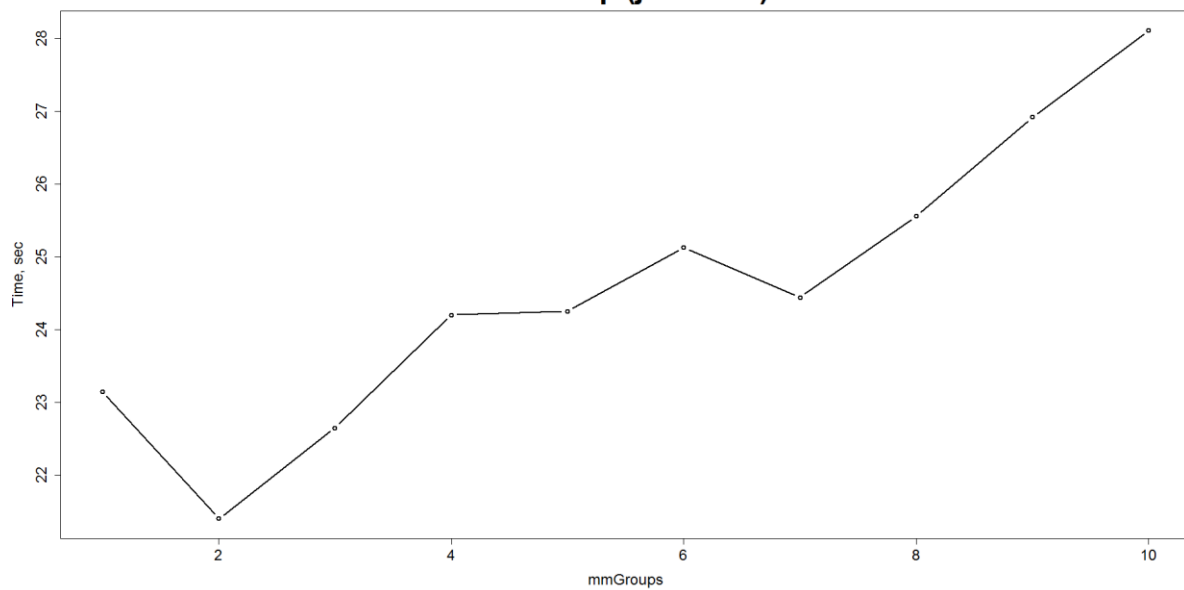
Mapper второго шага - тождественный, он получает результаты MultReducer и выдаёт их в том же виде $\langle (i_global, k_global), res \rangle$.

SumReducer

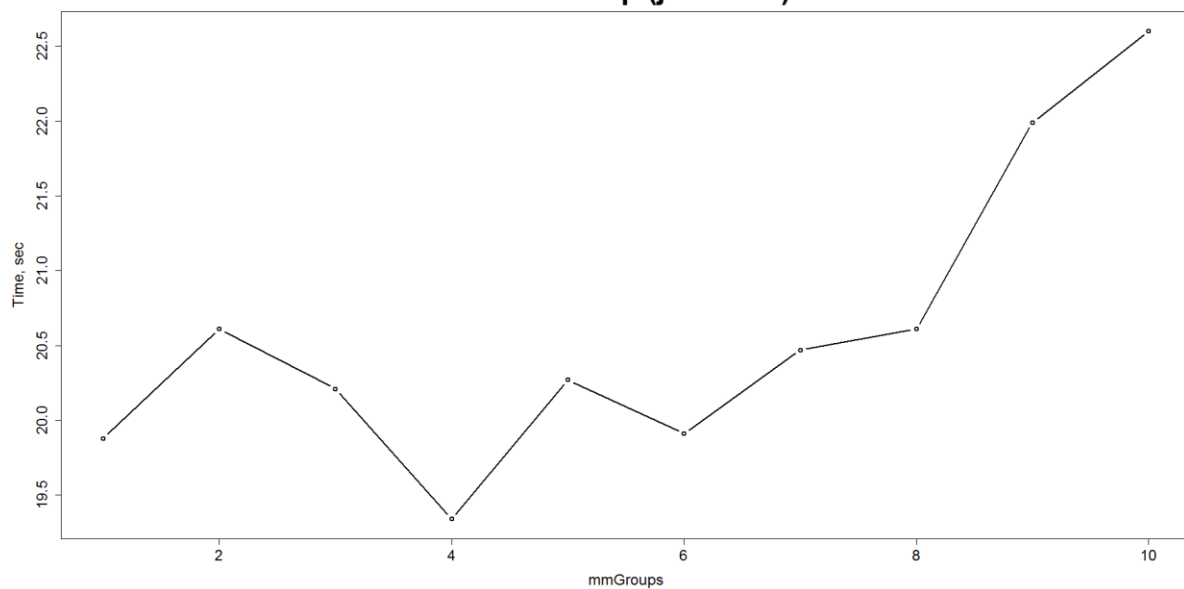
Каждый вызов `reduce` на втором шаге обрабатывает значения `res`, относящиеся к определенному значению пары индексов в матрице-результате умножения (i_global, k_global) . Reducer второго шага суммирует все значения и возвращает полученную сумму, если она отлична от нуля.

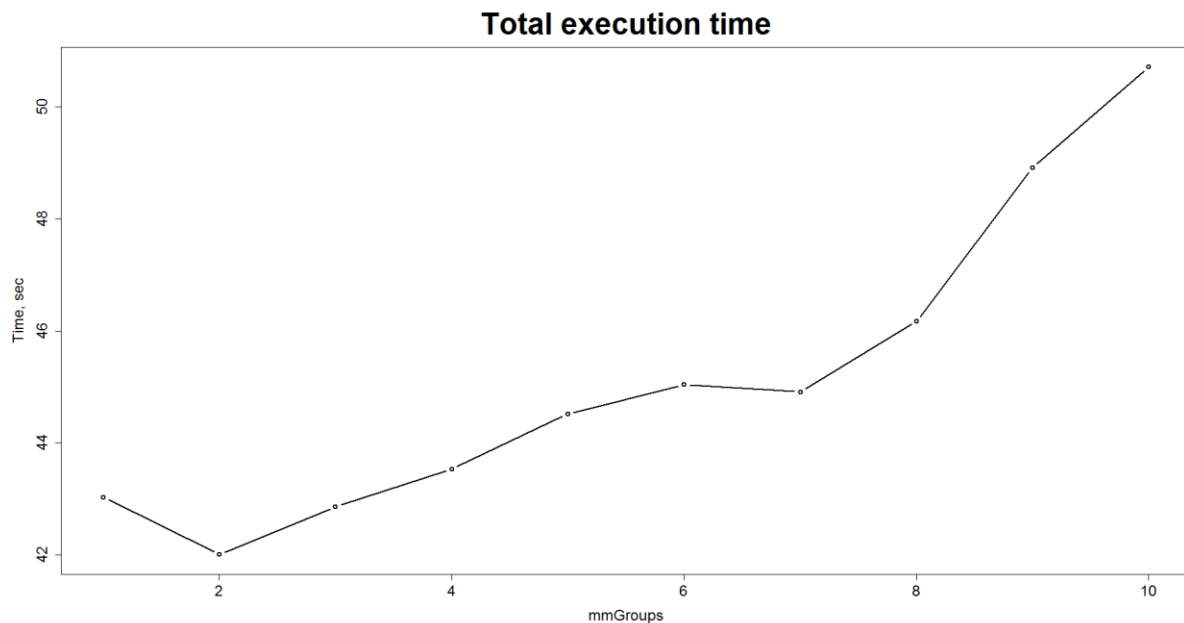
Оптимизация физического времени выполнения на матрицах 800x800 по параметру mmGroups (при numReducers = 32)

First step (job mm1)



Second step (job mm2)



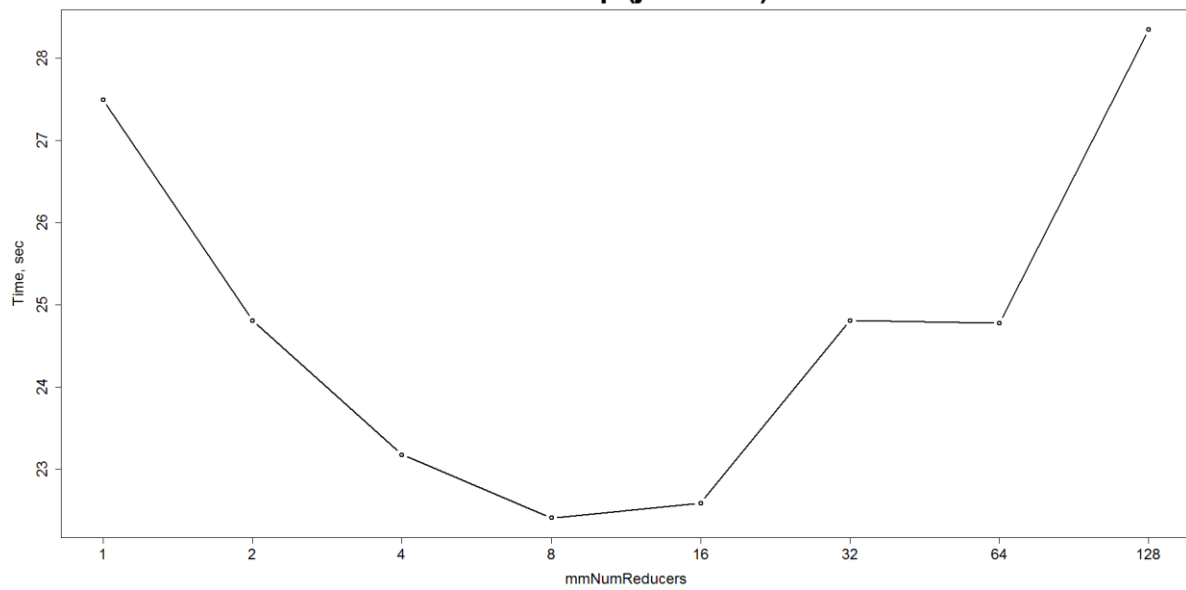


Минимальное общее время выполнения достигается на значении $mmGroups = 2$. Зависимости остальных счетчиков от параметра $mmGroups$:

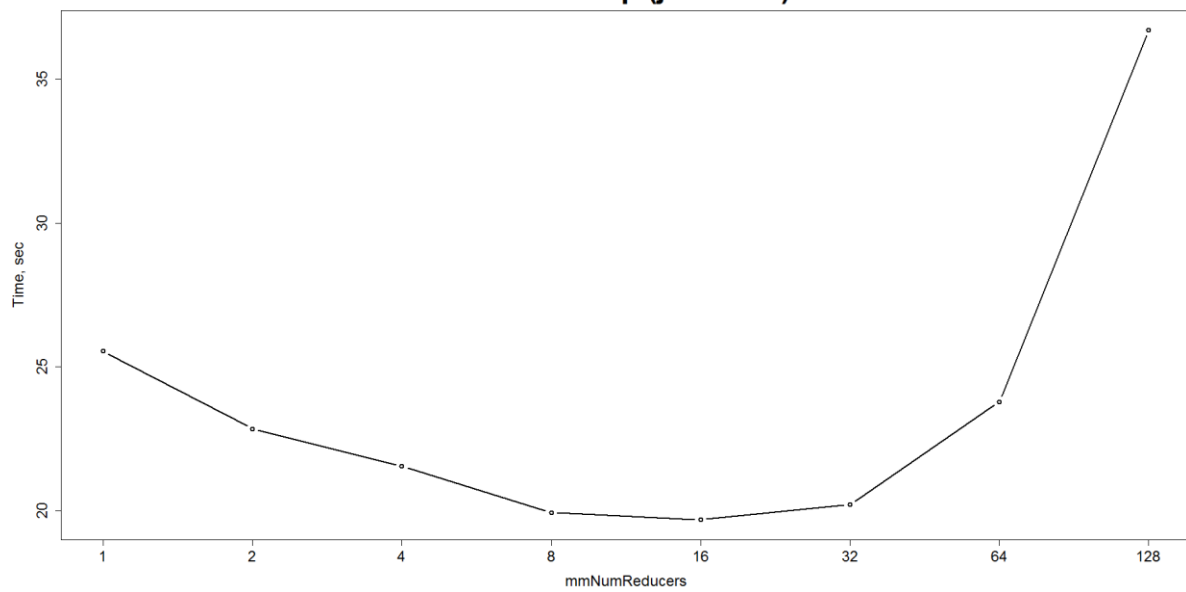
- $MAP_INPUT_RECORDS = 1024000 \text{ (const)}$;
- $MAP_OUTPUT_RECORDS = 1024000 * mmGroups$;
- $MAP_OUTPUT_BYTES = 22246400 * mmGroups$;
- $REDUCE1_INPUT_GROUPS = mmGroups^3$;
- $REDUCE1_SHUFFLE_BYTES \approx 8857942 * mmGroups$ (зависимость очень близка к линейной);
- $REDUCE1_INPUT_RECORDS = 1024000 * mmGroups$;
- $REDUCE1_OUTPUT_RECORDS = 512000 * mmGroups$;
- $REDUCE2_INPUT_GROUPS = 512000 \text{ (const)}$;
- $REDUCE2_SHUFFLE_BYTES \approx 8444953 * mmGroups$ (зависимость очень близка к линейной);
- $REDUCE2_INPUT_RECORDS = 512000 * mmGroups$;
- $REDUCE2_OUTPUT_RECORDS = 512000 \text{ (const)}$.

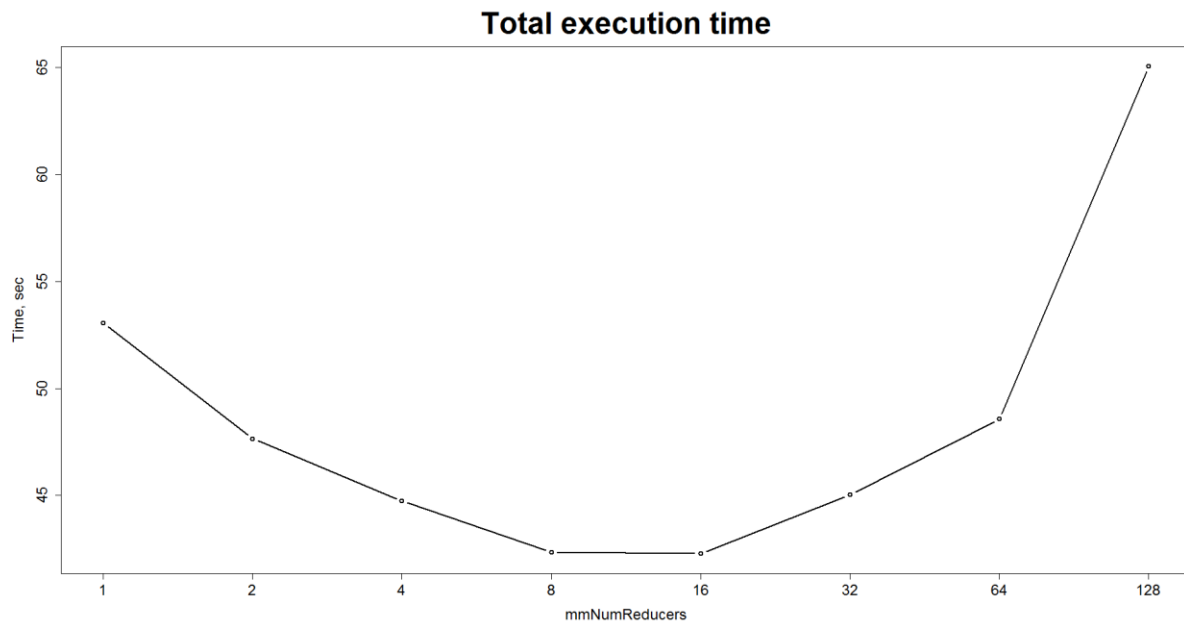
Оптимизация физического времени выполнения на матрицах 800x800 по параметру mmNumReducers (при mmGroups = 2)

First step (job mm1)



Second step (job mm2)





Минимальное общее время выполнения достигается на значении `mmNumReducers = 16`. Зависимости остальных счетчиков от параметра `mmNumReducers`:

`MAP_INPUT_RECORDS = 1024000 (const);`

`MAP_OUTPUT_RECORDS = 2048000 (const);`

`MAP_OUTPUT_BYTES = 44492800 (const);`

`REDUCE1_INPUT_GROUPS = 8 (const);`

`REDUCE1_SHUFFLE_BYTES` ≈ 17671424 (зависимость очень близка к постоянной);

`REDUCE1_INPUT_RECORDS = 2048000 (const);`

`REDUCE1_OUTPUT_RECORDS = 1024000 (const);`

`REDUCE2_INPUT_GROUPS = 512000 (const);`

`REDUCE2_SHUFFLE_BYTES` ≈ 17449121 (зависимость очень близка к постоянной);

`REDUCE2_INPUT_RECORDS = 1024000 (const);`

`REDUCE2_OUTPUT_RECORDS = 512000 (const).`

Коэффициент репликации Mapper первого шага = `mmGroups`;

Размер редукции Reducer первого шага = $2 * groupSize^2$;

Коэффициент репликации Mapper второго шага = 1;

Размер редукции Reducer второго шага = `mmGroups`.