Московский Государственный Университет им. М.В. Ломоносова
Факультет Вычислительной Математики и Кибернетики
Кафедра Суперкомпьютеров и Квантовой Информатики

Спецкурс: системы и средства параллельного программирования.

Отчёт № 2.
Анализ влияния размера блока матрицы при блочном умножении на показания системных счётчиков.

Работу выполнил
Мокров К.С.

Москва 2018

Постановка задачи и формат данных.

Задача: Реализовать блочный алгоритм матричного умножения и оценить влияние размера блока на показания системных счётчиков(Такты процессора, L1 и L2 промахи данных, время работы программы, MFLOPS).
Формат командной строки: <имя файла матрицы A> <имя файла матрицы B> <режим запуска программы>[<номер набора счётчиков>требуется, если до этого был выбран пункт 1].
Режимы: 0 – вывод в терминал, 1 – вывод в файл.
Наборы счётчиков: 0 – Такты процессора, L1 и L2 промахи данных, 1 – время работы программы, MFLOPS.
Формат файла-матрицы: Матрица представляются в виде бинарного файла следующего формата:

Тип	Значение	Описание
Число типа char	T – f (float)	Тип элементов
Число типа size_t	N – натуральное число	Число строк матрицы
Число типа size_t	M – натуральное число	Число столбцов матрицы
Массив чисел типа T	$N \times M$ элементов	Массив элементов матрицы

Элементы матрицы хранятся построчно.

Описание алгоритма.

Математическая постановка: Алгоритм блочного матричного умножения ($A \times B = C$) можно представить в простейшем случае в виде:

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} = \begin{bmatrix} A_{11} * B_{11} + A_{12} * B_{21} & A_{11} * B_{12} + A_{12} * B_{22} \\ A_{21} * B_{11} + A_{22} * B_{21} & A_{21} * B_{12} + A_{22} * B_{22} \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & B_{22} \end{bmatrix}$$

Оценка влияния размера блока и порядка суммирования на показания системных счётчиков осуществляется за счёт изменения размера блока и перестановки индексов суммирования. Оптимальный размер блока определяется из формулы:
3 * b * b = L / sizeof(float), где L — размер L1 кэша, b — оптимальный размер.
Анализ показаний счётчиков: Для снятия показания с системных счётчиков использовалась утилита PAPI. Функции:
int PAPI_flops(float * rtime, float * ptime, long_long * flpops, float * mflops);
int PAPI_start_counters(int * events, int array_len);
int PAPI_stop_counters(long long * values, int array_len);
Верификация: Для проверки корректности работы программы использовались тестовые данные.
Основные функции:

- **Чтение файлов матриц.** В рамках функции осуществляется анализ совместимости входных матриц и их чтение.
- **Перемножение матриц.** В рамках функции осуществляется перемножение матриц в соответствие с выбранным порядком индексов суммирования.
- **Снятие показаний с счётчиков.** В соответствии с требованиями пользователя.

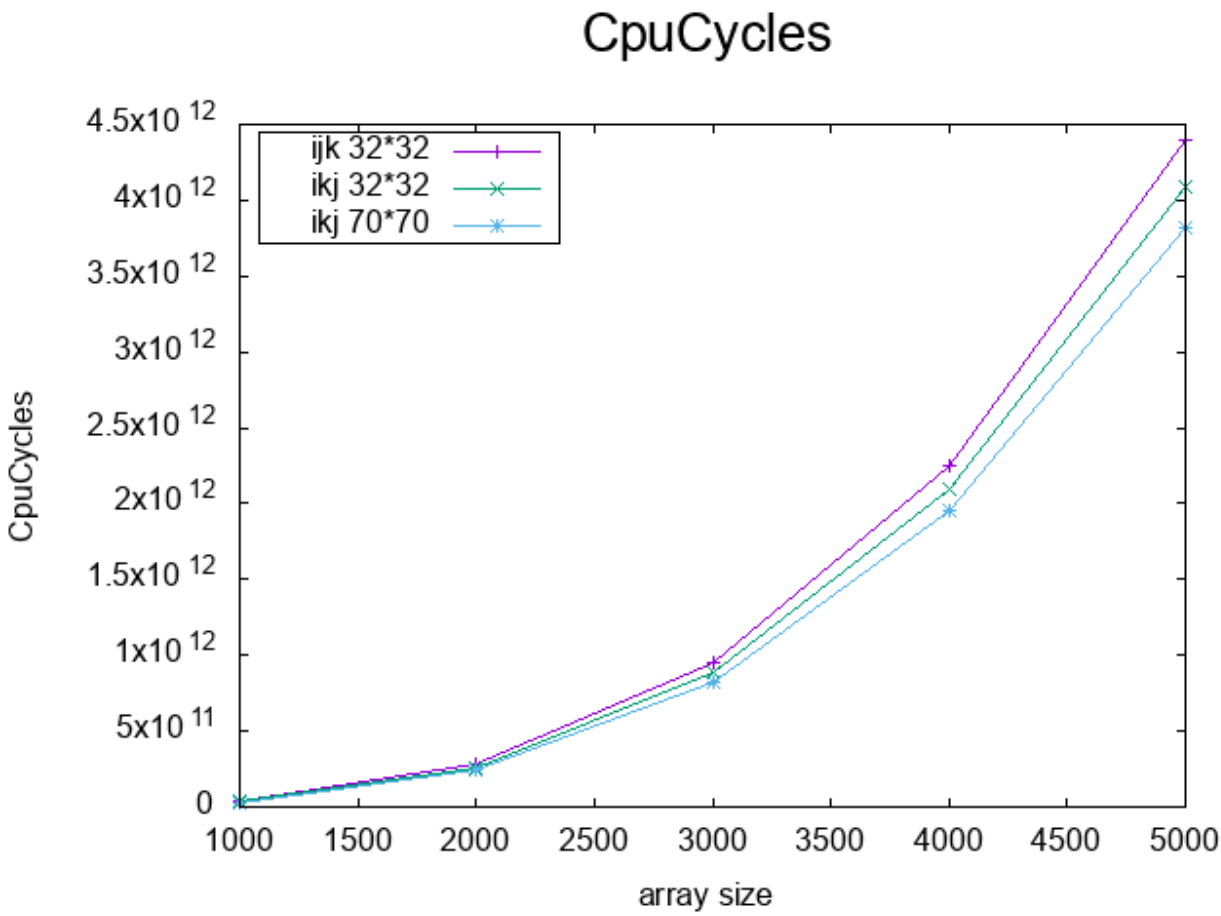
Результаты выполнения.

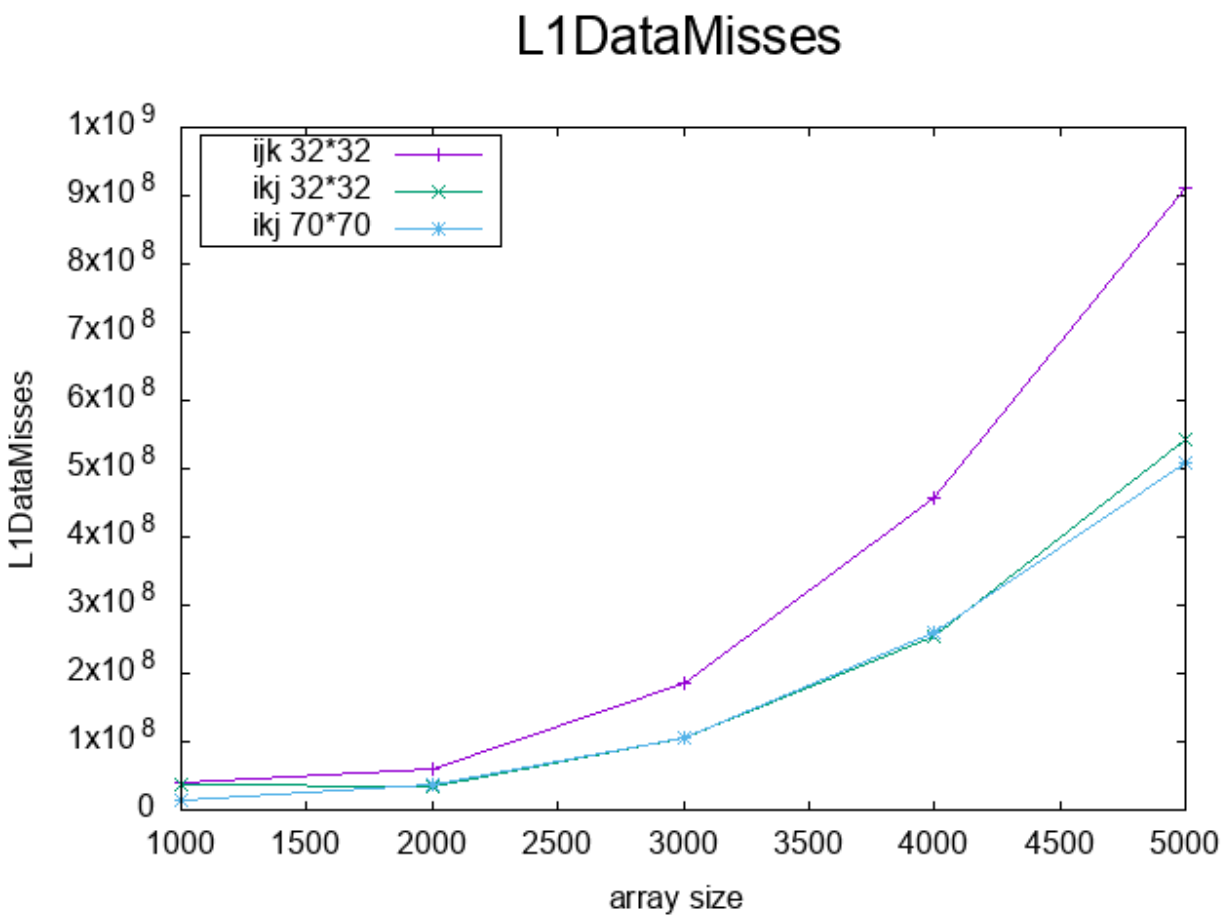
Результаты:

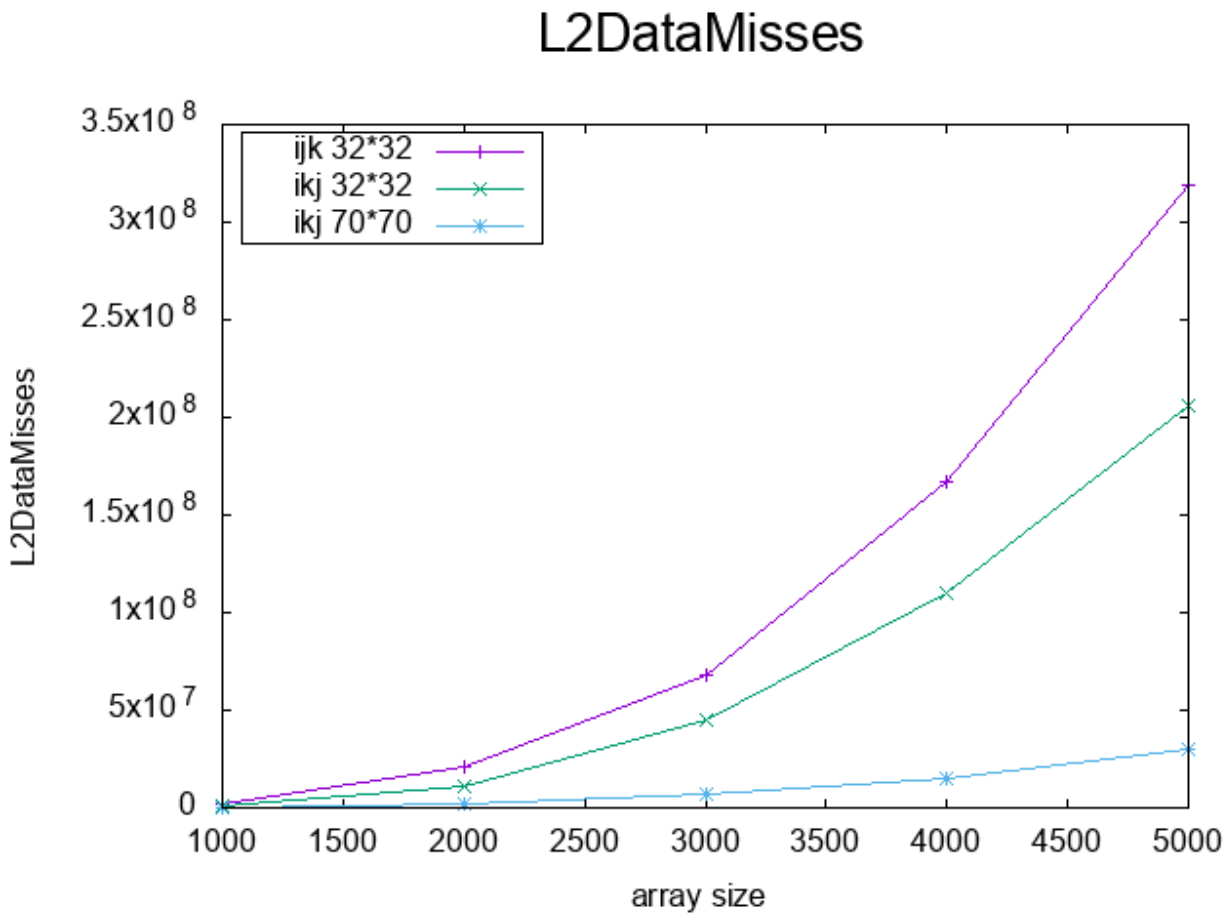
Проводилось перемножение матриц размерами 1000x1000, 2000x2000, 3000x3000, 4000x4000, 5000x5000. Зависимость счётчиков от соответствующих условий запуска представлена на графиках(в конце).

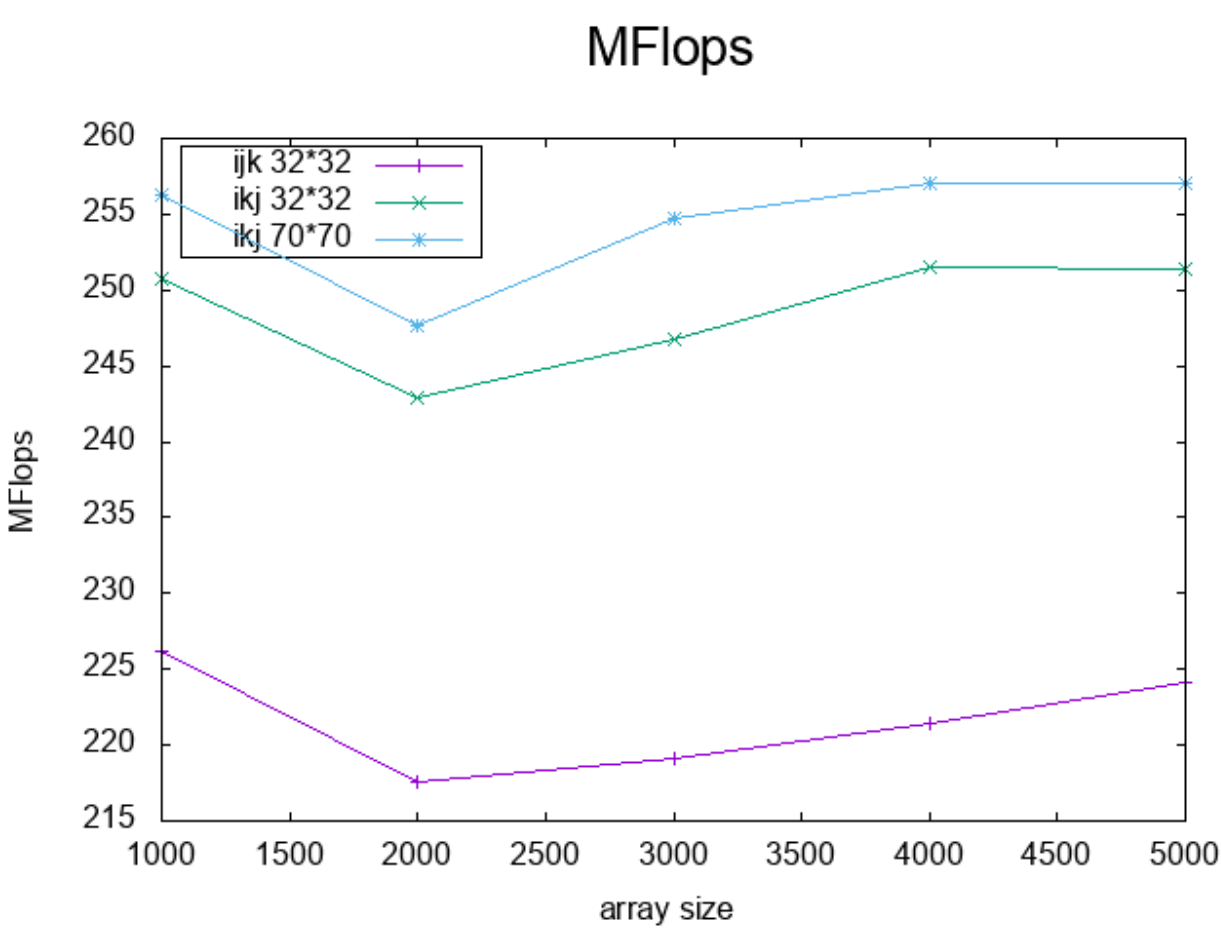
Основные выводы.

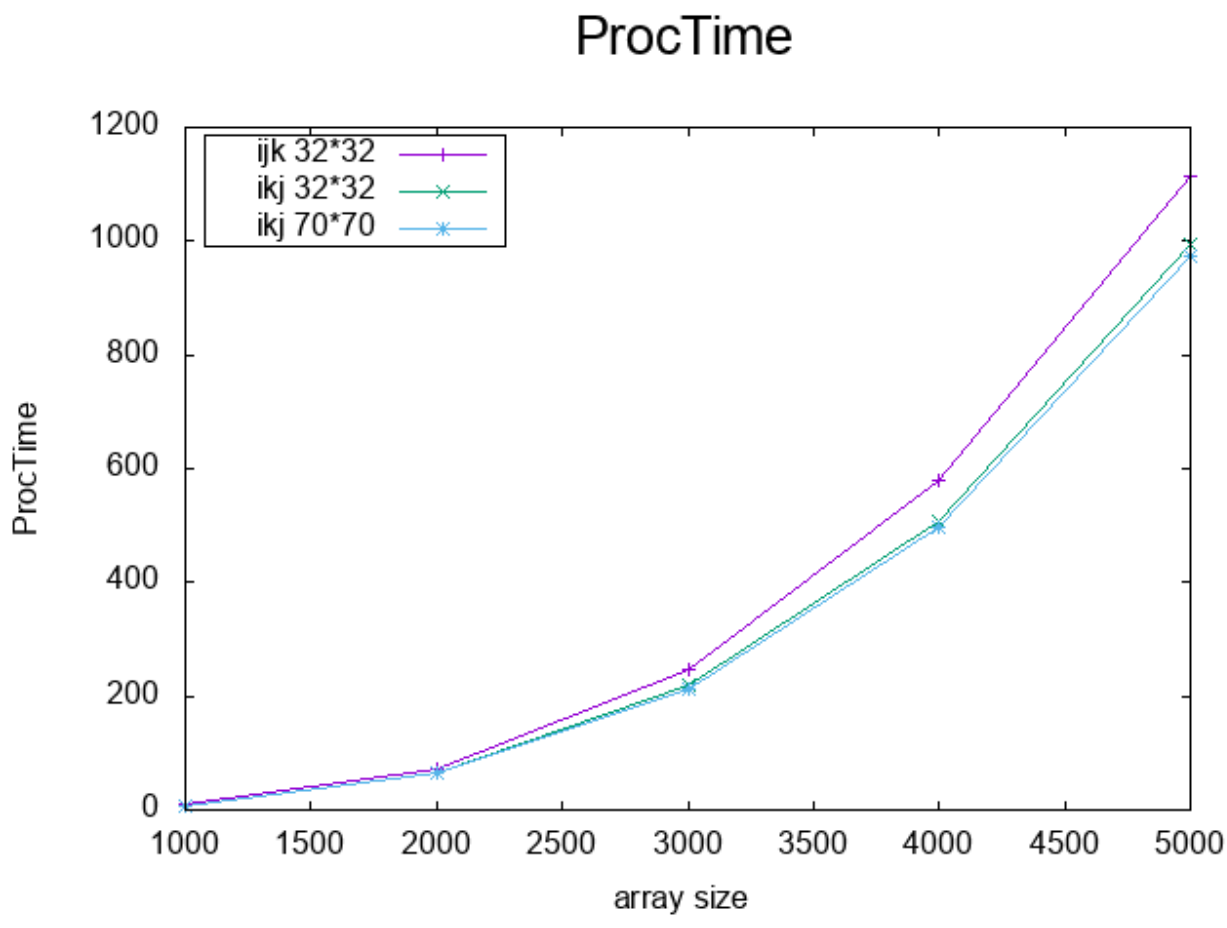
Исследования показывают, что не только изменение порядка индексов суммирования (ijk->ikj), но и использование оптимального размера блока оказывают положительное влияние на показания всех счётчиков. Блоки матриц, при использования оптимального размера, начинают помещаться в L1 кэш и мы видим прирост производительности для всех размеров матриц.




L1DataMisses
ijk 32*32
ikj 32*32
ikj 70*70
L1DataMisses
1x10 9
9x10 8
8x10 8
7x10 8
6x10 8
5x10 8
4x10 8
3x10 8
2x10 8
1x10 8
0
1000
1500
2000
2500
3000
3500
4000
4500
5000
array size


L2DataMisses
ijk 32*32
ikj 32*32
ikj 70*70
L2DataMisses
3.5x10 8
3x10 8
2.5x10 8
2x10 8
1.5x10 8
1x10 8
5x10 7
0
1000
1500
2000
2500
3000
3500
4000
4500
5000
array size


MFlops
ijk 32*32
ikj 32*32
ikj 70*70
MFlops
array size
260
255
250
245
240
235
230
225
220
215
1000
1500
2000
2500
3000
3500
4000
4500
5000


ProcTime
ijk 32*32
ikj 32*32
ikj 70*70
ProcTime
array size
1200
1000
800
600
400
200
0
1000
1500
2000
2500
3000
3500
4000
4500
5000