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 кеша в алгоритм умножения блочной матрицы и оценка эффекта

Работу выполнил
Тони Кастильо Мартин

Москва 2018

Постановка задачи и формат данных.

Снимать необходимо информацию о промахах кэша (1 и 2 уровней), числе процессорных тактов, числе FLOP-ов и TLB, в зависимости от размеров блока (фиксированный или по формуле из лекций) и двух порядков индексов, для 5 квадратных матриц.

Задача: Реализовать последовательный алгоритм блочного матричного умножения и оценить влияние кэша на время выполнения программы. Дополнить отчёт результатами сбора информации с аппаратных счётчиков, используя систему PAPI.

Формат командной строки: <имя файла матрицы A> <имя файла матрицы B> <имя файла матрицы C> <режим, порядок индексов> <размер блока>
Режимы: 7 – ijk, 8 – ikj
Размер: 1- формула.

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 матричного умножения ($A \times B = C$) можно представить в следующем алгоритме:

```
for (int i = 0; i < N; i+=b)  // b is the blocksize
    for (int j = 0; j < N; j+=b)
        for (int k = 0; k < N; k+=b)
            /* B x B mini matrix multiplications */
            for (i1 = i; i1 < i+b; i++)
                for (j1 = j; j1 < j+b; j++)
                    for (k1 = k; k1 < k+b; k++)
                        c[i1][j1] += a[i1][k1]*b[k1][j1];
```

Анализ времени выполнения: Для оценки времени выполнения программы использовалась функция:

1- PAPI_start_counters().
2- PAPI_flops()
3- PAPI_ipc()

и следующие события:

1- PAPI_L1_DCM
2- PAPI_L1_DCA
3- PAPI_L2_DCM

Для повышения надёжности экспериментов опыты проводились несколько раз (10).
Верификация: Для проверки корректности работы программы использовались тестовые данные.

Основные функции:

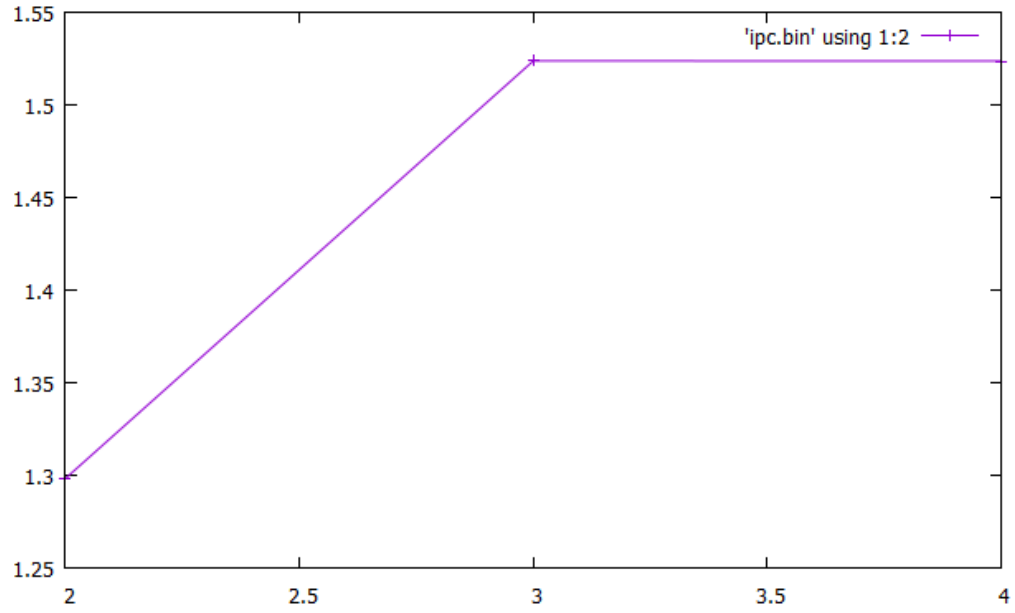
- **Разбор командной строки.** В рамках функции осуществляется анализ и разбор командной строки.
- **Чтение файлов матриц.** В рамках функции осуществляется анализ совместимости входных матриц и их чтение.
- **Перемножение матриц.** В рамках функции осуществляется перемножение матриц в соответствие с выбранным порядком индексов суммирования.

Результаты выполнения.

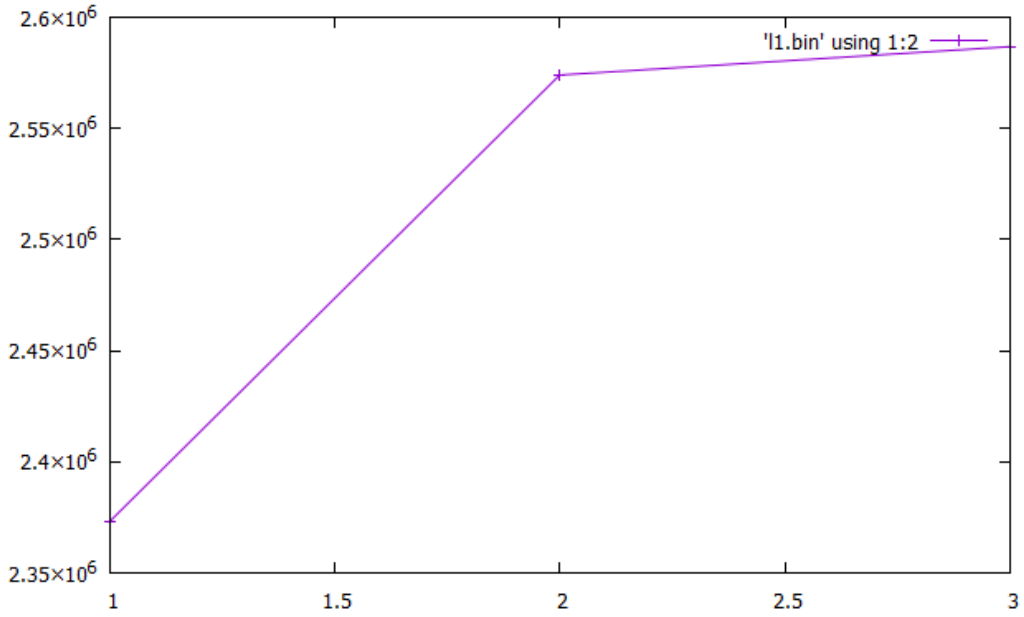
Результаты:

Проводилось перемножение двух матриц размерами $2000x2000$.

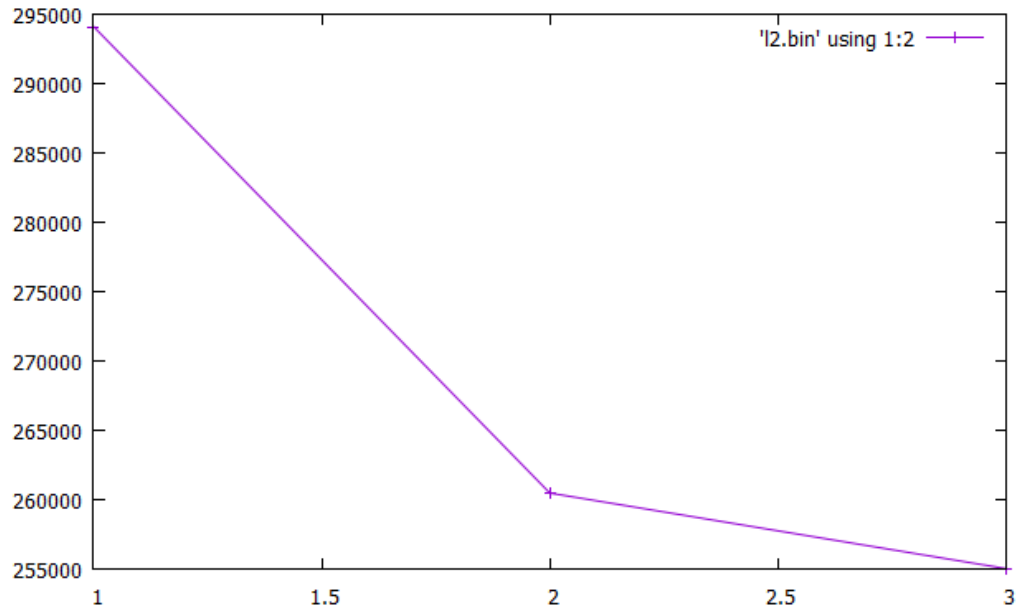
1- Зависимость времени выполнения рабочих циклов: для размера блока 32x32 и порядка индексов ijk; для размера блока 32x32 и порядка индексов ikj; для размера оптимального блока, определённого по формуле, и порядка индексов ikj. представлена на графике (время в секундах).



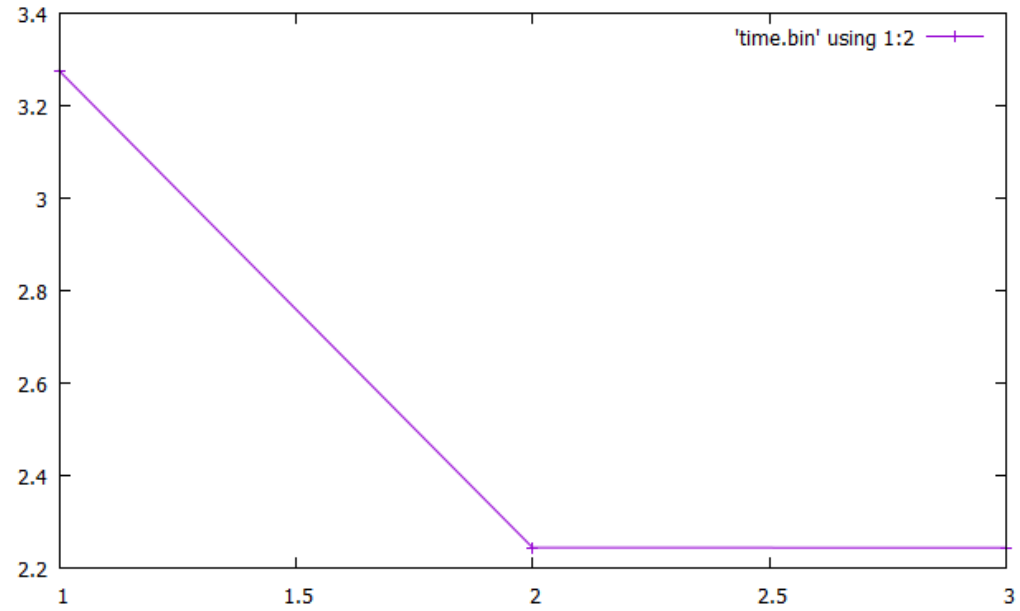
2- Зависимость промахов кэша L1: для размера блока 32x32 и порядка индексов ijk; для размера блока 32x32 и порядка индексов ikj; для размера оптимального блока, определённого по формуле, и порядка индексов ikj. представлена на графике (время в секундах).



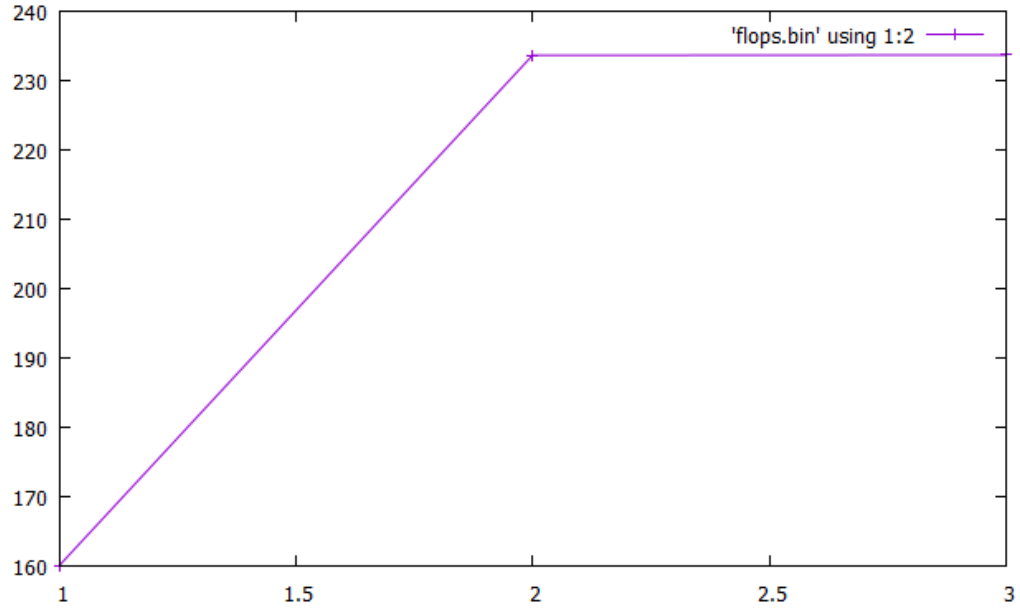
3- Зависимость промахов кэша L2: для размера блока 32x32 и порядка индексов ijk; для размера блока 32x32 и порядка индексов ikj; для размера оптимального блока, определённого по формуле, и порядка индексов ikj. представлена на графике (время в секундах).



4- Зависимость процессорных тактов: для размера блока 32x32 и порядка индексов ijk; для размера блока 32x32 и порядка индексов ikj; для размера оптимального блока, определённого по формуле, и порядка индексов ikj. представлена на графике (время в секундах).



5- Зависимость FLOP: для размера блока 32x32 и порядка индексов ijk; для размера блока 32x32 и порядка индексов ikj; для размера оптимального блока, определённого по формуле, и порядка индексов ikj. представлена на графике (время в секундах).



Основные выводы.

Исследования показывают, что изменения порядка индексов суммирование оказывает влияние на время выполнения программы. Наименьшее время выполнения рабочих циклов при следующем порядке индексов - ijk. Наименьшее время промахов кэша L1 при следующем порядке индексов - ijk. Наименьшее время промахов кэша L2 при следующем порядке индексов - ikj. Наименьшее время выполнения процессорных тактов при следующем порядке индексов - ikj. Наименьшее время выполнения FLOP при следующем порядке индексов - ijk.