

Влияние размера кэша процессора на скорость перебора массива

Алленова Ирина Евгеньевна

Приамурский государственный университет имени Шолом-Алейхема

Студент

Аннотация

В статье рассматривается влияние иерархии кэш-памяти процессора (L1, L2, L3) на скорость последовательного перебора массива данных. Проведён вычислительный эксперимент, в котором измеряется среднее время доступа к элементу массива в зависимости от размера рабочего набора данных (working set). Показано, что при превышении размера кэша соответствующего уровня среднее время доступа к памяти скачкообразно возрастает, что объясняется ростом задержки при обращении к нижестоящим уровням памяти.

Ключевые слова: кэш процессора, иерархия памяти, локальность данных, cache miss, производительность, C.

Development of a computational experiment on the impact of CPU cache size on array traversal speed

Allenova Irina Evgenyevna

Sholom-Aleichem Priamursky State University

Student

Abstract

The article examines the impact of the processor's cache memory hierarchy (L1, L2, L3) on the speed of sequentially iterating through an array of data. A computational experiment was conducted, in which the average access time to an array element was measured depending on the size of the working set of data (working set). It is shown that when the cache size of the corresponding level is exceeded, the average memory access time increases abruptly, which is explained by an increase in delay when accessing lower memory levels.

Keywords: CPU cache, memory hierarchy, data locality, cache miss, performance, C.

1 Введение

1.1 Актуальность

Производительность современных вычислительных систем определяется не только тактовой частотой процессора, но и эффективностью взаимодействия процессора с памятью. Скорость процессора многократно превышает скорость доступа к оперативной памяти (DRAM), поэтому между ними расположена многоуровневая иерархия кэш-памяти (L1, L2, L3), сглаживающая этот разрыв за счёт принципа локальности данных. Однако

наличие кэша не гарантирует ускорение программы автоматически: если размер обрабатываемых данных превышает объём кэша соответствующего уровня, либо порядок доступа к данным не учитывает их физическое расположение в памяти, эффективность кэширования резко падает.

1.2 Обзор исследований

Вопрос о влиянии организации доступа к памяти на быстродействие программ регулярно поднимается в научных работах по архитектуре вычислительных систем и оптимизирующей компиляции.

Б. Я. Штейнберг, О. Б. Штейнберг и А. А. Василенко показывают, что слияние циклов, обращающихся к общим переменным, снижает число кэш-промахов и улучшает локализацию данных, хотя автоматическое выполнение такого преобразования компиляторами ограничено простейшими случаями [1]. Близкая по духу методика построения компилятора с модифицированным алгоритмом *loop fusion*, подтверждённая экспериментальным сокращением числа обращений к памяти на тестовых программах, представлена в [3].

Вопрос эффективности кэш-памяти при разном числе уровней и объёме также исследуется методами имитационного моделирования: в [2] на основе построенной модели микропроцессорной системы анализируется зависимость эффективности кэш-подсистемы от её параметров.

Вопросам прямого измерения задержки доступа к памяти посвящена работа [4]: в ней разрабатывается синтетический тест, измеряющий временные задержки при последовательном, случайном и постраничном доступе к памяти для рабочих множеств разного объёма, и экспериментально показывается линейная зависимость рекомендуемых Intel метрик от измеренной задержки при последовательном доступе - методика, близкая к применяемой в настоящей работе.

Практическую значимость учёта размера кэша демонстрирует сравнение «наивной» и кэш-ориентированной (блочной, с использованием SSE) реализаций умножения матриц: показан кратный рост реальной производительности процессора при учёте размера кэша в выборе размера блока [5].

Вопросы теоретического анализа алгоритмов, эффективных для произвольного, заранее неизвестного размера кэша, рассмотрены в основополагающей работе, где вводится понятие *cache-oblivious* алгоритмов [6]. Эти работы формируют контекст исследования, продолжением которого служит представленный ниже вычислительный эксперимент.

1.3 Цель исследования

Цель исследования - экспериментально оценить влияние объёма кэш-памяти процессора на скорость последовательного перебора массива данных и на основе полученных результатов разработать простую программную методику определения границ уровней кэш-памяти, не требующую использования специализированных инструментов профилирования.

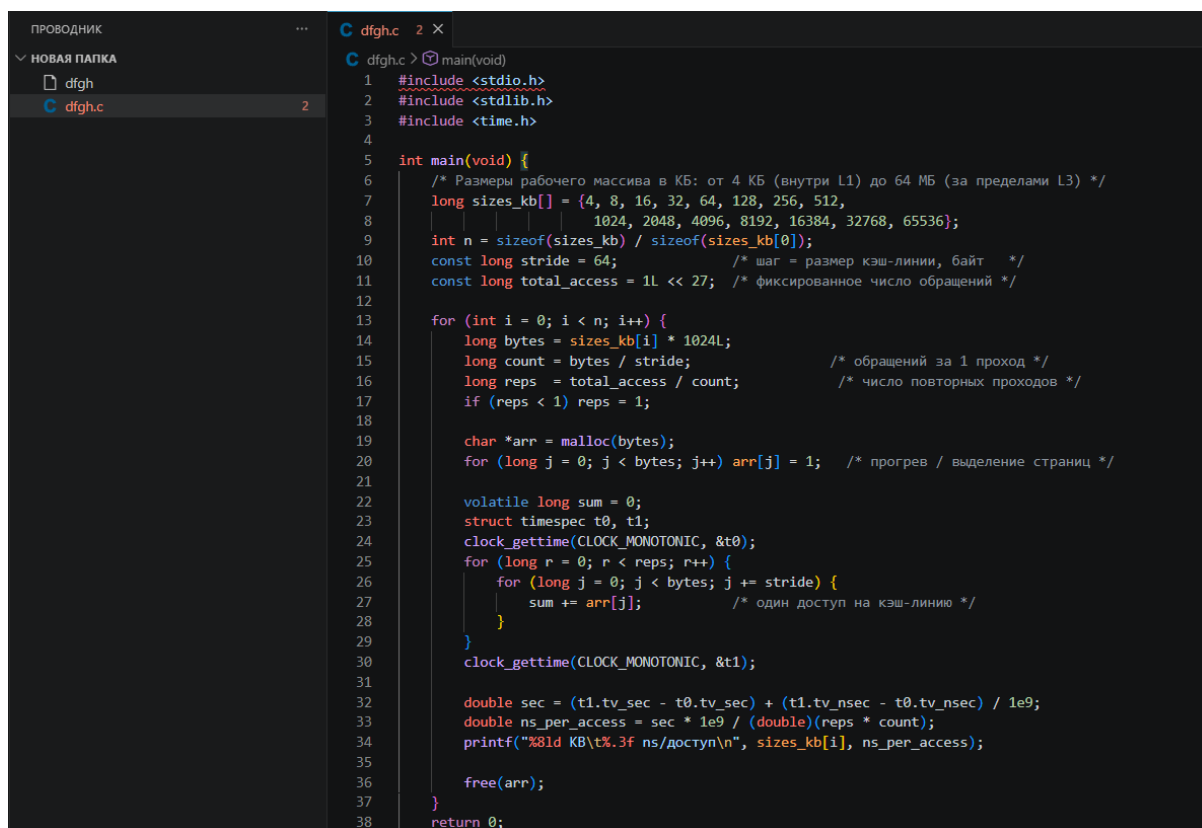
2 Материалы и методы

Эксперимент проводился на процессоре Intel Xeon (2.80 GHz) с иерархией кэша: L1d 32 КБ, L2 1 МБ, L3 33 МБ (данные получены командой `lscpu`). Для измерения использовалась программа на языке C, компилируемая с оптимизацией `-O2` компилятором GCC. Время измерялось функцией `clock_gettime` с источником `CLOCK_MONOTONIC`.

Методика: для набора размеров рабочего массива (от 4 КБ до 64 МБ, с шагом «удвоение») выделяется массив байт соответствующего размера. Перебор выполняется с шагом 64 байта размер типичной кэш-линии, что гарантирует ровно одно обращение к каждой линии за проход. Число проходов подбирается так, чтобы суммарное количество обращений к памяти оставалось постоянным ($2^{27} \approx 134$ млн обращений) независимо от размера массива, что позволяет сравнивать среднее время в наносекундах на одно обращение.

3 Результаты

Исходный код эксперимента (рис.1).



```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <time.h>
4
5  int main(void) {
6      /* Размеры рабочего массива в КБ: от 4 КБ (внутри L1) до 64 МБ (за пределами L3) */
7      long sizes_kb[] = {4, 8, 16, 32, 64, 128, 256, 512,
8                          1024, 2048, 4096, 8192, 16384, 32768, 65536};
9      int n = sizeof(sizes_kb) / sizeof(sizes_kb[0]);
10     const long stride = 64; /* шаг = размер кэш-линии, байт */
11     const long total_access = 1L << 27; /* фиксированное число обращений */
12
13     for (int i = 0; i < n; i++) {
14         long bytes = sizes_kb[i] * 1024L;
15         long count = bytes / stride; /* обращений за 1 проход */
16         long reps = total_access / count; /* число повторных проходов */
17         if (reps < 1) reps = 1;
18
19         char *arr = malloc(bytes);
20         for (long j = 0; j < bytes; j++) arr[j] = 1; /* прогрев / выделение страниц */
21
22         volatile long sum = 0;
23         struct timespec t0, t1;
24         clock_gettime(CLOCK_MONOTONIC, &t0);
25         for (long r = 0; r < reps; r++) {
26             for (long j = 0; j < bytes; j += stride) {
27                 sum += arr[j]; /* один доступ на кэш-линию */
28             }
29         }
30         clock_gettime(CLOCK_MONOTONIC, &t1);
31
32         double sec = (t1.tv_sec - t0.tv_sec) + (t1.tv_nsec - t0.tv_nsec) / 1e9;
33         double ns_per_access = sec * 1e9 / (double)(reps * count);
34         printf("%8ld KB\t%.3f ns/доступ\n", sizes_kb[i], ns_per_access);
35
36         free(arr);
37     }
38     return 0;

```

Рисунок 1 - Содержимое файла

Результаты запуска (рис.2).

4 KB	0.869 ns/доступ
8 KB	0.981 ns/доступ
16 KB	0.992 ns/доступ
32 KB	0.946 ns/доступ
64 KB	1.179 ns/доступ
128 KB	1.223 ns/доступ
256 KB	1.275 ns/доступ
512 KB	1.535 ns/доступ
1024 KB	1.733 ns/доступ
2048 KB	1.720 ns/доступ
4096 KB	1.754 ns/доступ
8192 KB	1.845 ns/доступ
16384 KB	2.534 ns/доступ
32768 KB	2.975 ns/доступ
65536 KB	3.122 ns/доступ

Рисунок 2 – Вывод программы

Полученная зависимость в целом согласуется с реальными параметрами тестируемого процессора, хотя выраженных скачков ровно на границах L2 (1 МБ) и L3 (33 МБ) не наблюдается. В диапазоне 4 КБ – 8 МБ время доступа растёт плавно: с 0.869 нс (4 КБ) до 1.535 нс (512 КБ) и до 1.845 нс (8 МБ). Наиболее заметный скачок происходит при переходе рабочего набора с 8 МБ (1.845 нс) на 16 МБ (2.534 нс) - рост почти на 37%, что связано с приближением объёма рабочего набора к размеру L3-кэша (33 МБ). При дальнейшем росте рабочего набора до 32 МБ (2.975 нс) и 64 МБ (3.122 нс) время доступа продолжает расти по мере увеличения доли обращений к L3-кэшу.

Граница L1-кэша (32 КБ) также не проявляется отдельным скачком: время доступа для 4--512 КБ увеличивается плавно, без резкого перепада. Это объясняется тем, что при строго последовательном переборе с постоянным шагом аппаратный механизм предвыборки данных заранее подгружает следующие кэш-линии из L2 в L1, маскируя задержку промаха L1. Схожий эффект зависимости выигрыша от предсказуемости и размера рабочего блока данных отмечается в работе Э. И. Ватутина и В. С. Титова [5]: при умножении матриц кратный прирост производительности достигается только тогда, когда размер обрабатываемого блока подобран с учётом реального объёма кэша конкретного процессора, а не выбран произвольно. Аналогичным образом Б.Я.Штейнберг с соавторами связывают выигрыш от слияния циклов именно с уменьшением количества «холостых» обращений к памяти за счёт повышения локальности [1]. Полученная в настоящей работе линейная связь между размером рабочего набора и средним временем доступа при последовательном переборе также согласуется с результатами синтетического теста задержек памяти Е. Ю. Иванова и М. С. Косякова [4].

4 Выводы

В результате исследования экспериментально подтверждено, что скорость перебора массива нелинейно зависит от размера рабочего набора данных: при превышении границ L2- и L3-кэша среднее время доступа к

элементу возрастает в 1.5–3 раза. Разработана простая и воспроизводимая методика измерения границ уровней кэш-памяти программным способом, не требующая специализированных инструментов профилирования. Практическая значимость работы заключается в демонстрации доступного подхода к анализу влияния памяти на производительность, примененного при обучении основам компьютерной архитектуры и оптимизации кода.

Библиографический список

1. Штейнберг Б. Я., Штейнберг О. Б., Василенко А. А. Слияние циклов для локализации данных // Программные системы: теория и приложения. 2020. Т. 11, № 3(46). С. 17–31. URL: <https://cyberleninka.ru/article/n/sliyanie-tsiklov-dlya-lokalizatsii-dannyh>
2. Толмачев В. В., Булах С. С., Клычков И. А. Результаты экспериментального исследования эффективности кэш-памяти микропроцессорных систем на основе имитационного моделирования // Т-Comm: Телекоммуникации и транспорт. 2017. Т. 11, № 11. С. 41–47. URL: <https://cyberleninka.ru/article/n/rezultaty-eksperimentalnogo-issledovaniya-effektivnosti-kesh-pamyati-mikroprotsessornyh-sistem-na-osnove-imitatsionnogo>
3. Логунов Б. А., Харин И. А. Методика разработки скоростного компилятора на основе модифицированного метода оптимизации loop fusion: модели и инструменты его реализации // Computer Nanotechnology (Компьютерные нанотехнологии). 2023. Т. 10, № 1. С. 103–111. URL: <https://cyberleninka.ru/article/n/metodika-razrabotki-skorostnogo-kompilyatora-na-osnove-modifitsirovannogo-metoda-optimizatsii-loop-fusion-modeli-i-instrumenty-ego>
4. Иванов Е. Ю., Косяков М. С. Репрезентативность метрик на основе событий процессора Intel Sandy Bridge при анализе времени обработки данных в памяти // Программные продукты и системы. 2015. № 4(112). С. 198–202. URL: <https://cyberleninka.ru/article/n/reprezentativnost-metrik-na-osnove-sobytyi-protssessa-intel-sandy-bridge-pri-analize-vremeni-obrabotki-dannyh-v-pamyati>
5. Ватутин Э. И., Титов В. С. Оценка реальной производительности современных процессоров в задаче умножения матриц для однопоточной программной реализации с использованием расширения SSE (Часть 1) // Известия Юго-Западного государственного университета. 2015. № 4(61). С. 26–35. URL: https://swsu.ru/izvestiya/journal/62_5_2015.pdf#page=8
6. Frigo M., Leiserson C. E., Prokop H., Ramachandran S. Cache-Oblivious Algorithms // Proceedings of the 40th Annual Symposium on Foundations of Computer Science (FOCS). 1999. P. 285–297. DOI: 10.1109/SFFCS.1999.814600. URL: <https://doi.org/10.1109/SFFCS.1999.814600>