

Assignment 1 – Report

Quick sort vs Merge sort

Quick sort and Merge sort uses Divide and Conquer approach. Initial partition causes more cache misses for both the algorithms. But once a problem is divided into small sub-problems, cache miss rate decreases for Quick sort as every comparison in the *partition* method takes place among adjacent elements i.e. *localized* memory locations. For merge sort we have to merge two sorted lists (not necessarily adjacent) into a third list.

We can observe the difference between both the sorts based on cache miss at D1mw - Data write miss at first level and LLd - Last level cache miss.

Quick sort performs better than Merge sort in terms of cache locality.

Refer excel 1.1

Radix sort is having more cache misses

Radix sorts the data digit by digit starting from LSB to MSB digit which leads to continuous shuffling of the data. For its buckets, radix sort must employ excessively huge arrays. It is doing the same process (shuffling) till **d** number of times where **d** is maximum no of digit of element of input array.

Last Level cache misses are more expensive than L1 cache misses because this data is being read from Main Memory which requires more cycles. Therefore, I am considering LL cache misses for program analysis.

$$LL = LL\ d + LL\ i$$

LL (misses) = LL d (LL data misses at last level cache) + LL i (LL instruction misses at last level cache)

From the graph (excel 1.2) we can conclude that radix sort is having poor spatial locality.

Refer excel 1.2

Bubble and Selection sort

Both bubble and Selection sort are comparison based sorting. In bubble sort each pair of adjacent elements is compared and the elements are swapped if they are not in order. Hence holds good spatial locality. For worst case, bubble sort takes more time i.e $O(n^2)$

For selection sort, It also performs worst when input size is large. It finds out the minimum element in the array in each pass and places it at its correct place.

Both Bubble and Selection hold good spatial locality as compared to other sorting techniques.

Refer excel 1.2

Compulsory Miss

When a block is initially accessed but is not yet in the cache, a compulsory miss occurs, and the block needs to be added to the cache. For all sorting algorithms I1 misses and LL i Misses are the same irrespective of input size hence it is “Compulsory misses”.

I1mr is I1 cache read misses and ILmr is LL cache instruction read misses.

Initially cache is empty. When the first request comes it searches the instruction into Level 1 cache . if it's missed in I1 then search in LL happens. If instruction is still missing in LL (ILmr) then need to bring the block from main memory to caches.

For cache configuration 1 following are the observations :

Sorting Algorithm	Compulsory Miss (I1mr)
Merge	12
Quick	6
Radix	13
Bubble	4
Selection	3

Refer excel 1.3

Different cache configurations

I have done the following changes to I1, D1 and LLi:

Cache Configuration of System (C1)
I1 cache: 32KB, 64 B, 8-way associative
D1 cache: 32768 B, 64 B, 8-way associative
LL cache: 8MB, 64 B, 16-way associative

Cache Configuration of System (C1)
I1 cache: 32768 B, 64 B, 16-way associative
D1 cache: 32768 B, 64 B, 16-way associative
LL cache: 524288 B, 64 B, 2-way associative

Cache Configuration of System (C3)
I1 cache: 32768 B, 32 B, 8-way associative
D1 cache: 32768 B, 32 B, 8-way associative
LL cache: 8MB, 64 B, 16-way associative

Cache Configuration of System (C4)
I1 cache: 32768 B, 128 B, 8-way associative
D1 cache: 32768 B, 128B, 8-way associative
LL cache: 8MB, 64 B, 16-way associative

C2 configuration : As associativity of first level cache is higher than last level cache, we will see performance decreased significantly. It happened because L1 cache's Associativity is more than Last Level cache hence L1 cache needed more time to search elements .

For input size $n=500000$ LL miss of merge sort is 827251 for configuration C2 whereas it is 97008 for Configuration C1.

In excel 1.4 I have compared cache configuration 1 and cache configuration based on all input sizes.

Refer Excel. 1.4

Compulsory Miss & Block size

Note that increasing block size resulted in reduced compulsory misses (due to spatial locality) i.e For each compulsory miss, more data blocks are brought into cache .

Let's consider an example of merge sort with $n = 250,000$.

L1 Instruction cache

Cache size	Associativity	Block Size	miss
32KB	8 way	32B	1,665
32KB	8 way	64B	873
32KB	8 way	128B	620

L1 data cache

Cache size	Associativity	Block Size	miss
32KB	8 way	32B	991
32KB	8 way	64B	868
32KB	8 way	128B	614

LL

Cache size	Associativity	Block Size	miss
8MB	16 way	64B	50268
8MB	16 way	64B	50133
32KB	8 way	128B	25547

Note that the Configuration C2- 8MB cache is a special case where Associativity of LL cache is greater than L1 cache.

Commands

Following are the list of commands :

Compile program

`gcc -g <Program Name.c> -o <Program Name>`

Run callgrind

```
valgrind --tool=callgrind --cache-sim=yes ./<Program Name> <input>
```

Annotation

```
callgrind_annotate --inclusive=yes - - auto=yes callgrind.out.<pid> > Program Name.txt
```

To change L1,D1 and Associativity

```
valgrind --tool=callgrind --cache-sim=yes --l1=32768,16,64 --D1=32768,16,64 --LL=524288,2,64  
./selection 75000
```

(with different cache configuration & sorting algorithms)

To get IPC

```
perf stat ./merge 250000
```

IPC

Machine Instructions typically consist of a number of elementary microoperations that vary in numbers & Complexity depending on CPU organization used. single machine instruction may take one or more CPU cycles to complete termed as the Cycles Per Instruction (CPI).

$\text{CPU Time (Execution Time)} = \text{CPU Clock Cycle Time for Program} * \text{cycle Time}$

$\text{CPI} = \text{CPU Clock Cycle Time for Program} / \text{Instruction Count}$

$\text{IPC} = 1 / \text{CPI}$

Program to get cycle count :

```
#include <stdint.h>
```

```

uint64_t rdtsc(){
unsigned int lo,hi;

__asm__ __volatile__ ("rdtsc" : "=a" (lo), "=d" (hi));

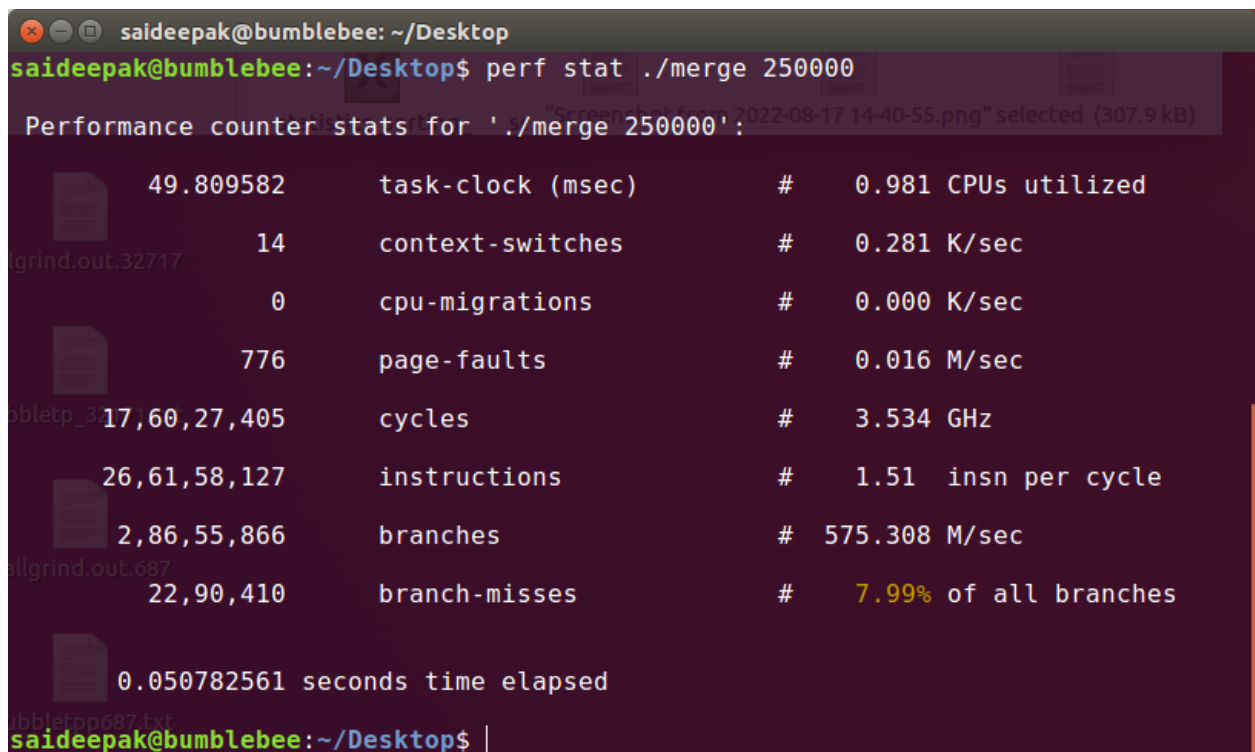
return ((uint64_t)hi << 32) | lo;

}

```

After getting the number of cpu cycles I divide it with Instruction count.the value which I got is above 2000 for different configurations. So I dont think having IPC > 2000 is a valid case.

Then I used perf tool to get IPC as follows :



```

saideepak@bumblebee: ~/Desktop
saideepak@bumblebee:~/Desktop$ perf stat ./merge 250000
Performance counter stats for './merge 250000':

   49.809582    task-clock (msec)         #    0.981 CPUs utilized
             14    context-switches          #    0.281 K/sec
              0    cpu-migrations            #    0.000 K/sec
             776    page-faults              #    0.016 M/sec
 17,60,27,405    cycles                    #    3.534 GHz
 26,61,58,127    instructions                #    1.51 insn per cycle
  2,86,55,866    branches                  # 575.308 M/sec
 22,90,410      branch-misses              #    7.99% of all branches

0.050782561 seconds time elapsed

saideepak@bumblebee:~/Desktop$

```

Refer Excel. 1.5

Configuration of my system

Memory	15.5 GiB
Processor	intel® Core™ i7-7700 CPU @ 3.60GHz × 8
Graphics	Intel® HD Graphics 630 (Kaby Lake GT2)
Os Type	64 bit
Disk	945.6 GB