CS6600: Computer Architecture
Assignment 1: Cache Profiling

Bala Dhinesh - EE19B011

Introduction:

Cachegrind simulator provides various cache statistics for different cache configurations. This assignment aims to perform cache statistics of various sorting algorithms like Bubble Sort, Selection Sort, Merge Sort, Quick Sort and Radix Sort for varying array lengths of 250_000, 500_000, 750_000, 1_000_000. Random *long int* values are chosen with a constant seed value so that we can able to reproduce the same array input across different sorting algorithms.

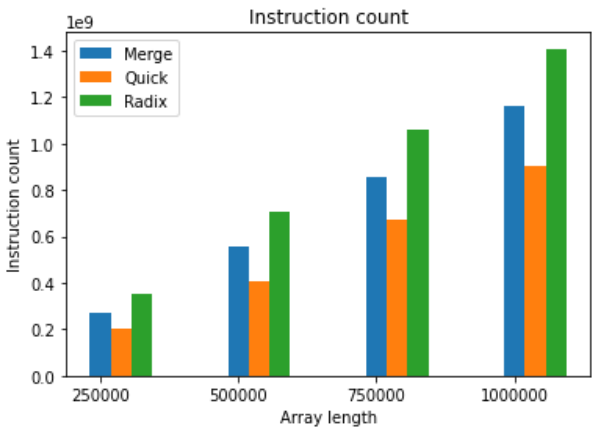
The algorithmic complexity of the Bubble and Selection sort is O(n^2), the Merge and Quick sort is O(N log N), and the Radix sort is O(N+K), where K is the number of digits.

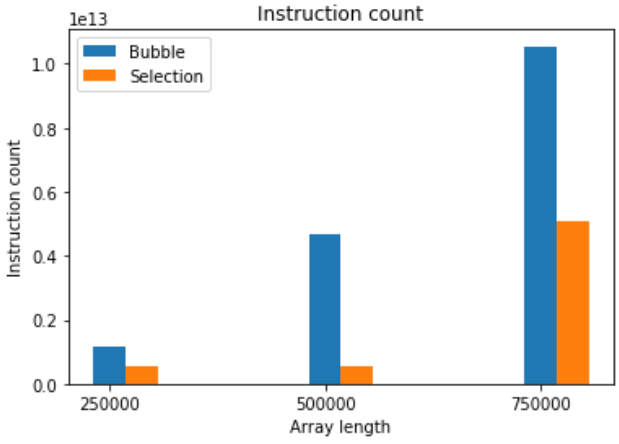
Default Cache configuration:

I1 cache: 32768 B, 64 B, 8-way associative
D1 cache: 32768 B, 64 B, 8-way associative
LL cache: 20971520 B, 64 B, 20-way associative

Analysis:

1. Instruction count:

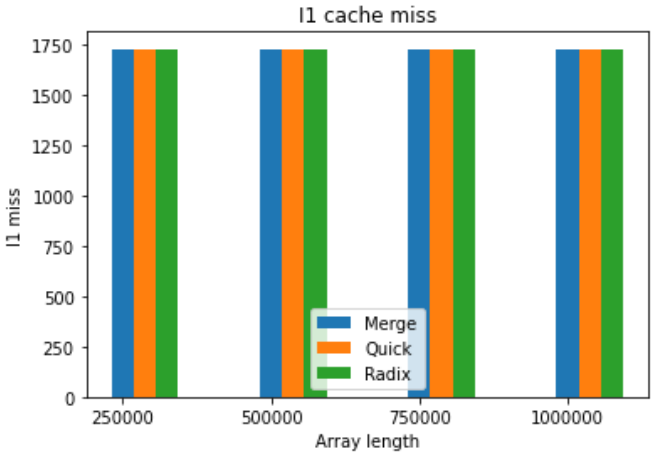


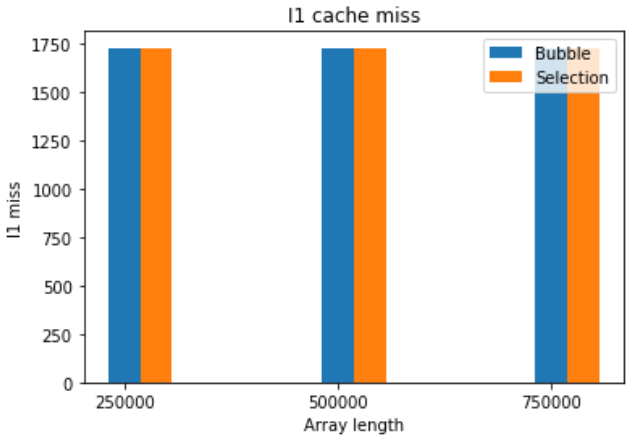


From the graph, we can infer that as the array length increases, the instruction count also increases. Instruction count has the minimum for Quick sort followed by Merge sort, Radix sort, Selection sort and Bubble sort. For selection and bubble sorting, an array length of 1,000,000 is not considered since it takes a very long time to compute.

2. First level I1 cache miss:

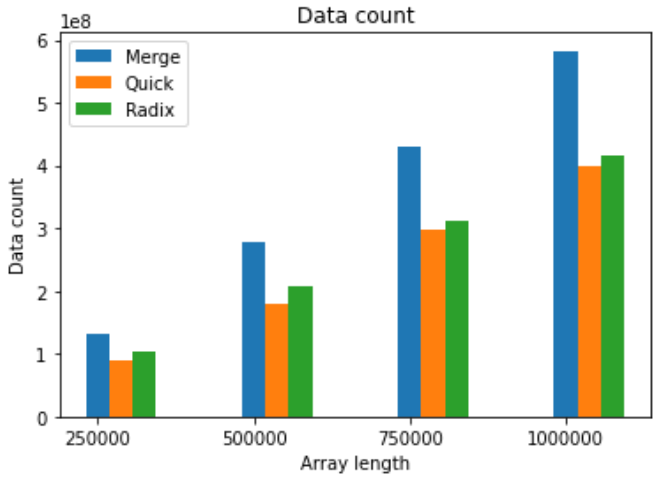
Here the cache misses are due to cold misses and hence all are almost constant for different array sizes. The same logic applies even for the Last level IL cache misses.

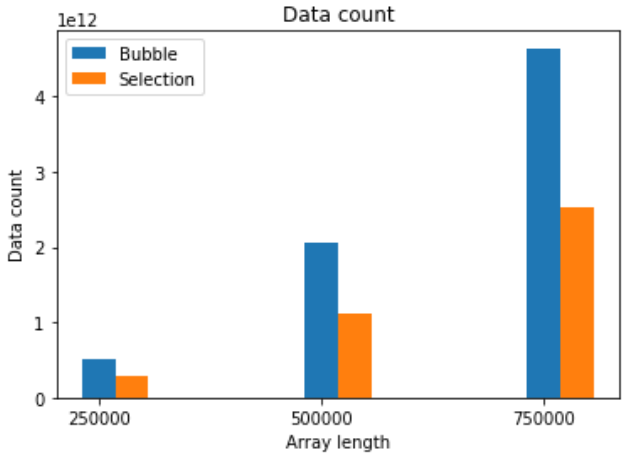




3. Data count:

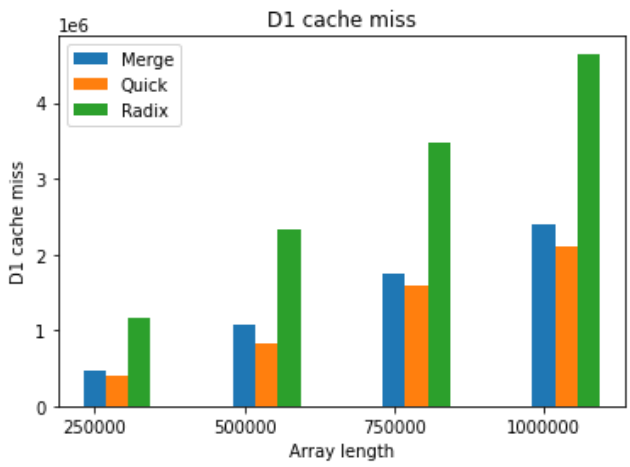
The image below shows the data read count for different array sizes. From the data, we can infer that as the array length increases, the data count also increases. Similar logic applies to data write count. Data read count has the minimum for Quick sort followed by Radix sort, Radix sort, Selection sort and Bubble sort.

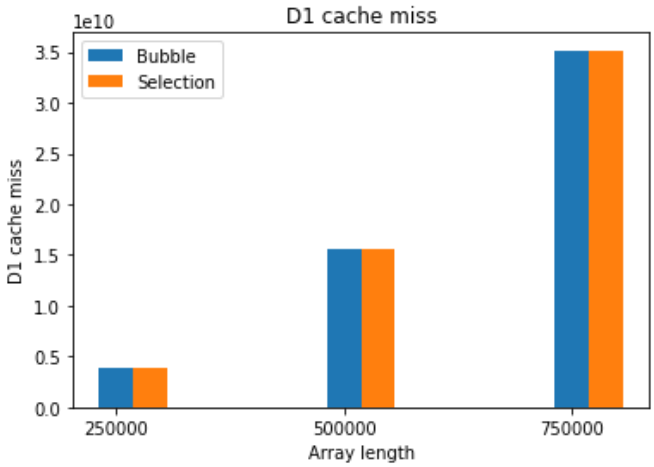




4. Data cache miss:

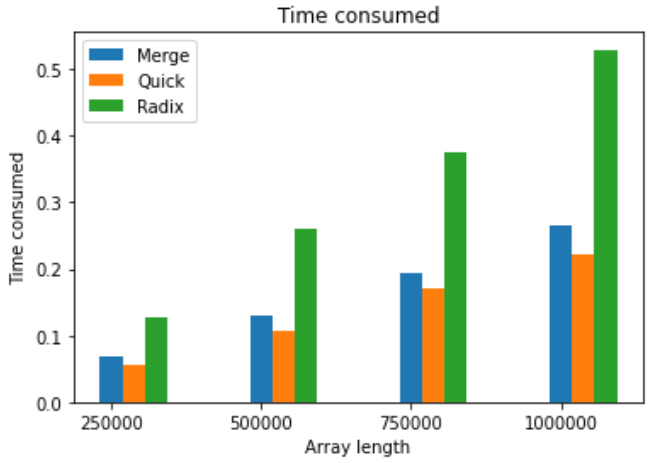
As array size increases, the cache miss rate increases since the L1 cache cannot fit the entire array(capacity miss) However, as we move to L3 last level data cache, the cache miss rate will be constant for all the different array size because, in L3 cache, there is no capacity miss and only cold misses, as the size of last level data cache is much higher compared to the first level data cache.

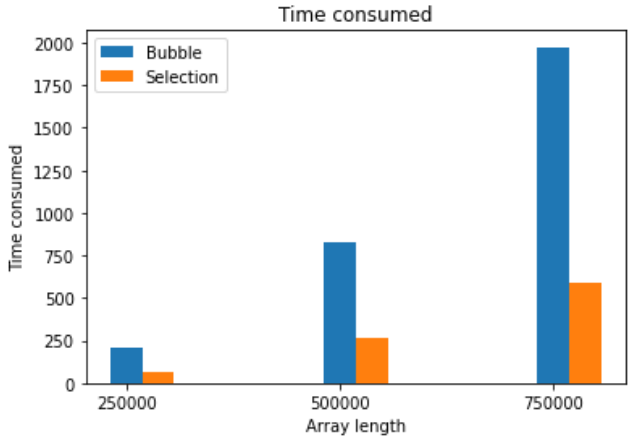




5. Time consumed:

From the graphs below, we can conclude that the time consumed is directly proportional to the time complexity of the algorithm followed by the array size.





6. Calculation of IPC:

Perf tool is used to find the instructions per cycle count(IPC). It provides a detailed analysis of many essential features such as cycle count, time utilisation, total instructions, IPC and so on. Below is the perf tool output for Radix Sort of 250000 elements.

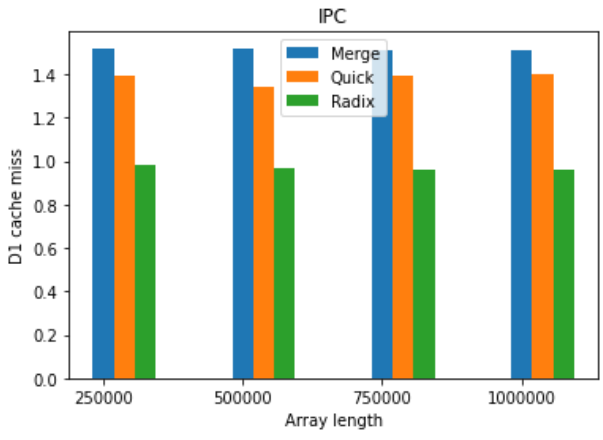
The syntax for running the perf command is:

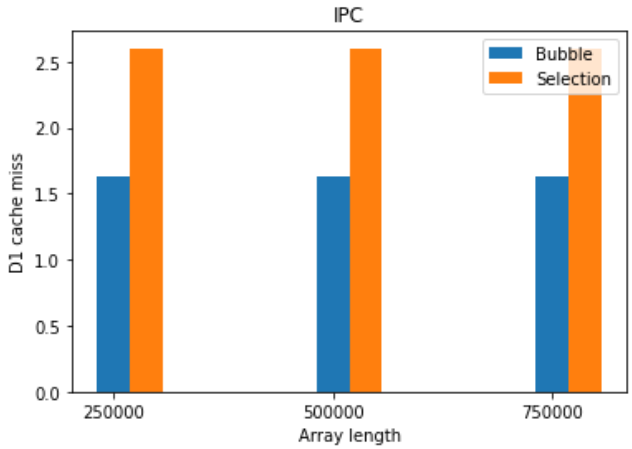
perf stat ./radix.out 250000

```
Performance counter stats for './radix.out 250000':

            132.19 msec task-clock:u              #    0.997 CPUs utilized
                 0      context-switches:u        #    0.000 /sec
                 0      cpu-migrations:u          #    0.000 /sec
             1,021      page-faults:u             #    7.723 K/sec
     36,14,88,200      cycles:u                  #    2.735 GHz
     35,22,09,715      instructions:u            #    0.97  insn per cycle
      1,37,86,450      branches:u                #  104.290 M/sec
            12,288      branch-misses:u          #    0.09% of all branches

      0.132593674 seconds time elapsed

      0.132591000 seconds user
      0.000000000 seconds sys
```

IPC count of 0.97 is inferred from the output.

7. IPC results:





We notice that the IPC value for different array sizes in a particular sorting algorithm is the same. This is because of the change in instruction count and time consumed to compensate for the effects.

System Configuration:

```
(base) ee19b011@pop-os:~/Files/CS6600/ass1$ lscpu
Architecture:                    x86_64
CPU op-mode(s):                  32-bit, 64-bit
Byte Order:                      Little Endian
Address sizes:                   39 bits physical, 48 bits virtual
CPU(s):                          20
On-line CPU(s) list:             0-19
Thread(s) per core:              2
Core(s) per socket:              10
Socket(s):                       1
NUMA node(s):                    1
Vendor ID:                       GenuineIntel
CPU family:                      6
Model:                           165
Model name:                      Intel(R) Core(TM) i9-10900 CPU @ 2.80GHz
Stepping:                        5
CPU MHz:                         2800.000
CPU max MHz:                     5200.0000
CPU min MHz:                     800.0000
BogoMIPS:                        5599.85
Virtualization:                  VT-x
L1d cache:                       320 KiB
L1i cache:                       320 KiB
L2 cache:                        2.5 MiB
L3 cache:                        20 MiB
NUMA node0 CPU(s):               0-19
```

Analysis for different cache configurations:

Default configuration:

I1 cache: 32768 B, 64 B, 8-way associative
D1 cache: 32768 B, 64 B, 8-way associative
LL cache: 20971520 B, 64 B, 20-way associative

Different I1 cache configurations are assigned and observed the trends are as follows:

1. 16384 B, 64 B, 8-way associative - decreased cache size by half
    a. There is a slight increase in the miss rate - due to capacity miss
2. 32768 B, 32 B, 16-way associative - increase associativity by twice and decrease block size by half
    a. Miss rate increases
3. 32768 B, 128 B, 4-way associative - increase the block size by twice and decrease associativity by half
    a. Since most of the miss rate is due to cold misses, increasing the block size helps reduce the miss rate.

If the D1 cache is configured with the following:

1. 16384 B, 64 B, 8-way associative - decreased cache size by half
    a. There is a slight increase in the write miss rate.