Reverse Engineering Sorting Algorithm Using Performance Counters

By
Nikhil Kumar singh (17CS60R63)

Under the supervision of
Debdeep Mukhopadhyay

Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Outline

- Introduction
  - Hardware Performance Counters (HPC)
  - Perf Tool
- Motivation
- Objective
- Related work
- Methodology
  - Sorting
  - Implementation
  - Implementation Result
- Conclusion
- Future Works
- References

Introduction

- Performance counters are special hardware registers available on most modern CPUs.

- *Performance Counters for Linux* (PCL) is a new kernel-based subsystem that provides a framework for collecting and analyzing performance data

- These registers count the number of certain types of hw events: such as
  - instructions executed, cache-misses suffered, or branches mispredicted
  - without slowing down the kernel or applications

Hardware performance counter

- Hardware Performance Counters (HPCs) are a set of special purpose registers, which are present in most of the modern microprocessor's Performance Monitoring Unit (PMU).

- These registers can be programmed to store the number of occurrences of different types of hardware and software events related to the execution of a program,
    - such as cache misses, retired instructions, retired branch instructions, and so on

Hardware performance counter(cont…)

- HPCs were primarily designed to debug the performance of complex software systems.

- currently, they are widely used for collecting the run-time behavioral information of software execution

- HPCs work along with the event selectors, which specify the hardware events to be monitored and a digital logic which increments a counter based on the occurrence of the specified hardware events.

Hardware performance counter(cont…)

- These performance counters can be accessed very fast without affecting or slowing down any software execution.

- Hardware Performance counters has been used in many recent literatures [18–21] for dynamic profiling of a system

Perf tool

- Perf is a profiler tool for Linux 2.6+ based systems that abstracts away CPU hardware differences in Linux performance measurements and presents a simple command line interface.

- Originally Performance Counters for Linux (PCL), focusing on CPU performance counters (programmable registers)

- Now a collection of profiling and tracing tools, with numerous subcommands,

Some Uses of Performance Counters

- Traditional analysis and optimization
- Finding architectural reasons for slowdown
- Validating Simulators
- Auto-tuning
- Operating System optimization
- Estimating power/energy in software

Motivation

Why do performance analysis?

- Reduce IT spend – find and eliminate waste, find areas to tune, and do more with less
- Build scalable architectures – understand system limits and develop around them
- Solve issues – locate bottlenecks and latency outliers

Objective

1. To Compare and analyze performance of Different Sorting algorithm Based On their HPCs values.

2. Reverse Engineering Sorting Algorithm Using Performance Counters and

Related Work

- Reverse Engineering Intel Last-Level Cache Complex Addressing Using Performance Counters[3]

- Branch Prediction based attacks using Hardware performance Counters.[5]

- Reverse Engineering Hardware Page Table Caches Using Side-Channel Attacks on the MMU[4]

Implementation

- Sorting algorithm
  - Insertion , merge , selection, Quick, Heap.

- Performance Counters Measured are:
  - Branches, Branche Misses, cache Misses, Cache reference, cpu cycle , NO. of Instruction.

LInk to code

https://drive.google.com/drive/folders/1k-1ar-4RQ61eX3Q2PW9rWQaHwkr_Ine_

Implementation

- Fore each hardware Counters we collected 500 values for each sorting algorithm. For example for selection sort we run the following command.

    g++ selection.cpp;

    for i in {1..500} ;

    do sudo perf stat -o s.txt --append -e branches ./a.out

    Done

- It runs selection sort for five hundred time and collects the Branches value in a text file named s.txt.

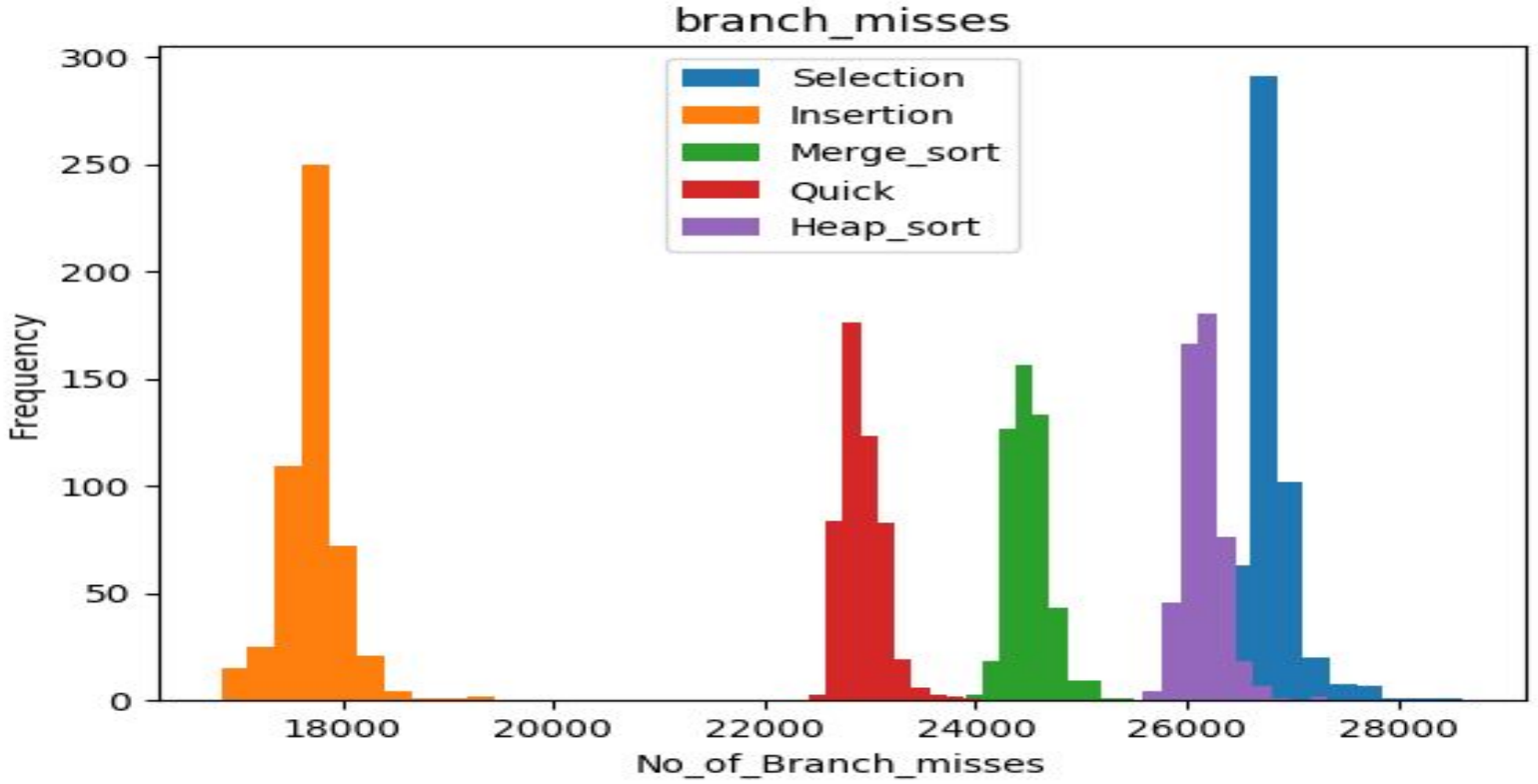
Implementation

- Similarly for other four sorting algorithm we collects the branches value.

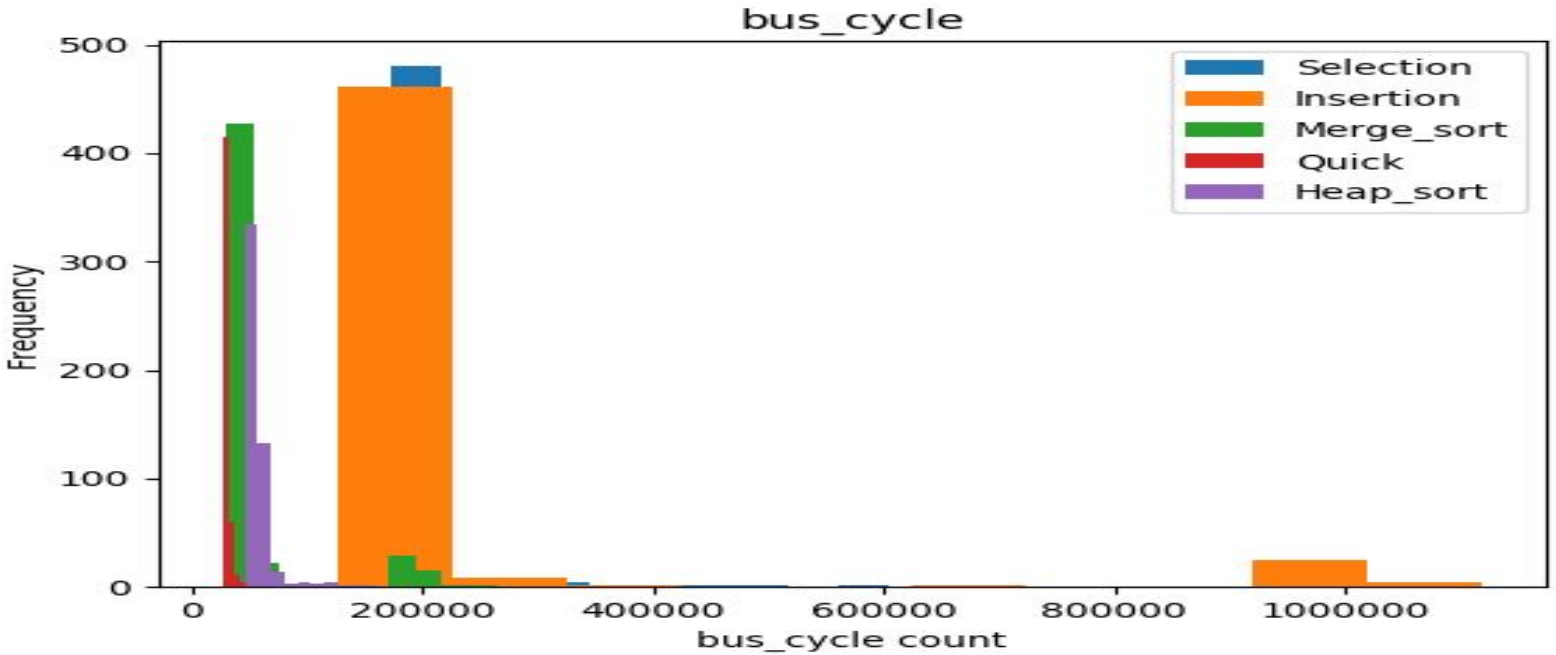
- We repeat the above steps for other HPCs. then we plotted the histogram for analysing the result.

Results for Branches

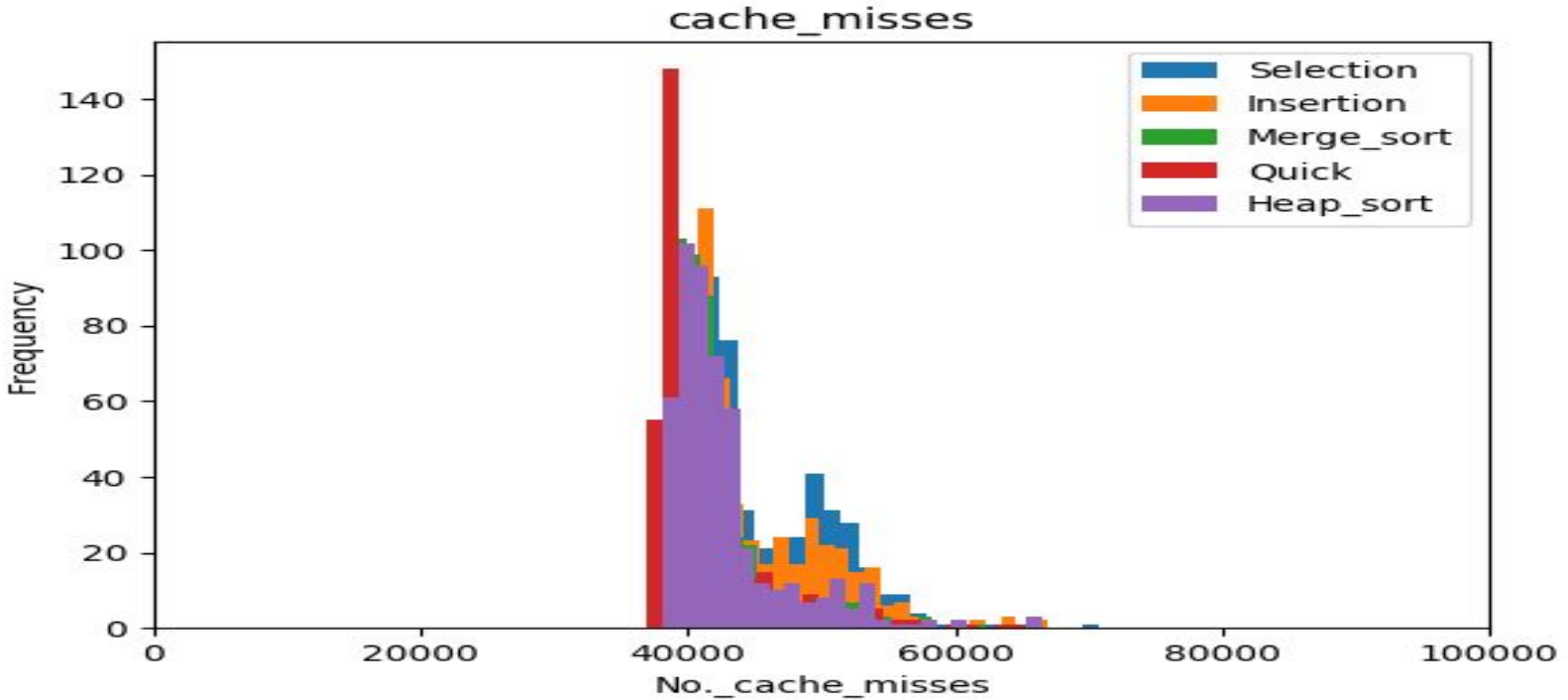
Result of Branch Misses



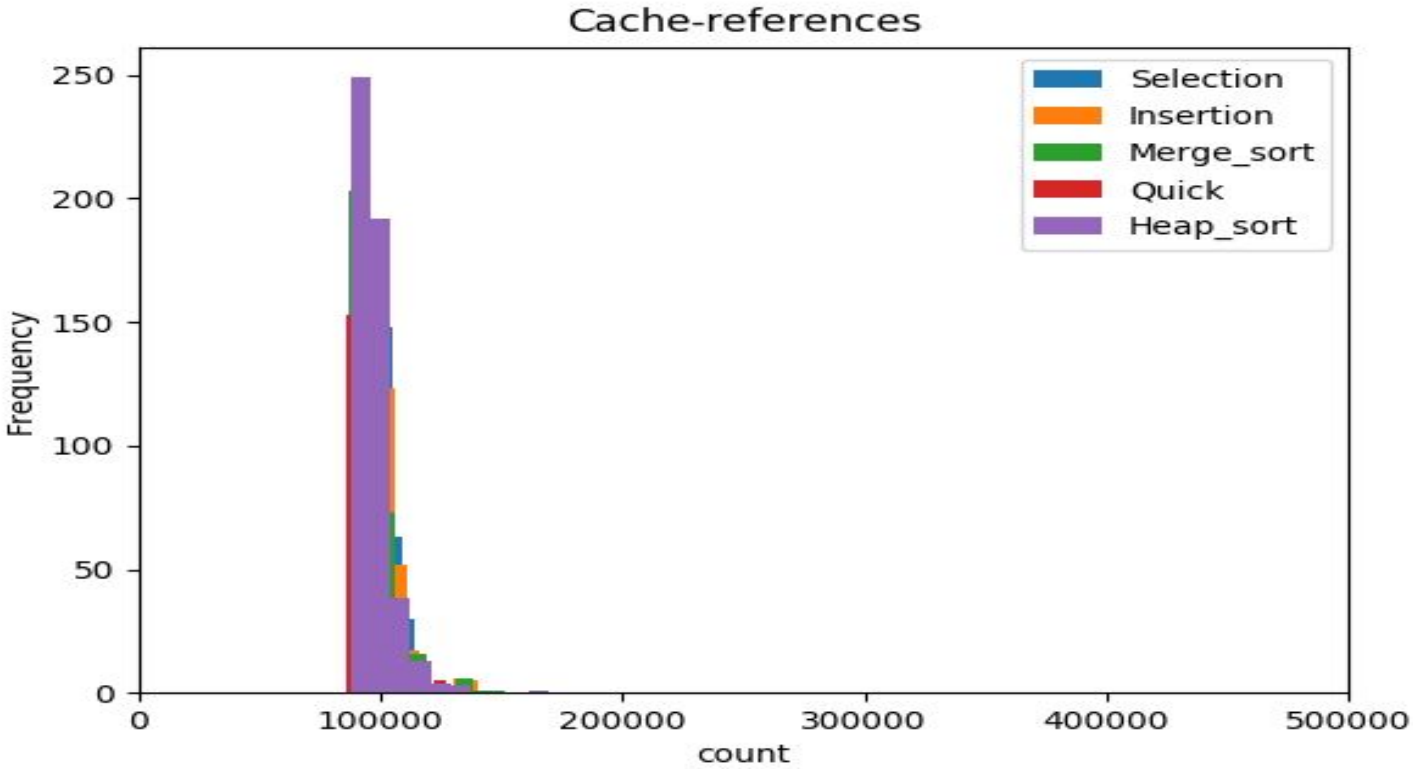
Result for Bus Cycles



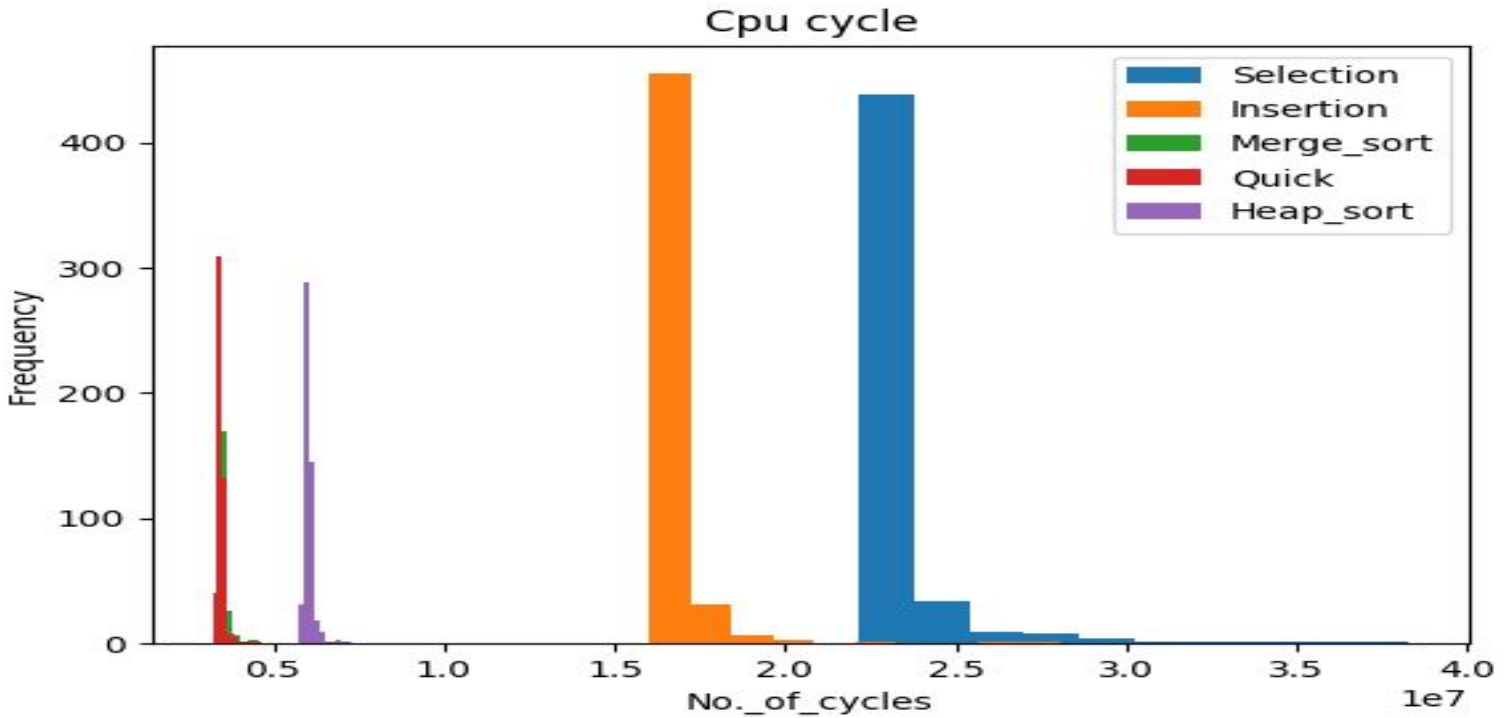
Result for cache-misses.



Result for Cache-References



Result for Cpu Cycle



Conclusion

- When tuning an application, either manually or automatically, there comes a point when further performance gains can only be achieved by truly understanding the minute details of the microarchitecture

- In this work we captured the hardware performance counter of different standard sorting algorithm and compared them based on their hardware performance counter value.

conclusion

- We found that except for cache references and cache misses all other HPCs are distinguable.

- Also we observe that quicksort is most optimal sorting algorithm

Futures Works

- we will introduced a method to reverse engineer Sorting Algorithms, using hardware performance counters.

- We will try to Implement sorting in different way, so that it is optimal and reverse engineering cannot be applied.

References.

1https://pdfs.semanticscholar.org/c419/3d67010a54c386bc9a2c75417cd6b0ad67c9.pdf

2 http://web.cse.ohio-state.edu/hpcs/WWW/HTML/publications/papers/TR-00-6.pdf

3https://pdfs.semanticscholar.org/0204/b810b62ced1b96d337d215572f5a56e05440.pdf?_ga=2.226506200.419825127.1542051089-45807953.1542051089

4 https://www.cs.vu.nl/~herbertb/download/papers/revanc_ir-cs-77.pdf

5 http://cse.iitkgp.ac.in/~debdeep/courses_iitkgp/HardSec/resources/ppt.pdf