

Homework #5

cpe 631

Kyle Ray

February 26, 2018

Contents

[Answers 1](#_Toc491808573)

[Output 1](#_Toc491808574)

[Source Code 1](#_Toc491808575)

[Appendix 1](#_Toc491808576)

# Purpose

To become familiar with the Intel PIN and learn more about measuring the performance of modern computer systems with specific benchmarks.

# Problem 1

The inscount0.cpp pin tool will count the total number of instructions executed by the application it is monitoring and write the result to an output file. In this situation the application being monitored is running a SPEC benchmark. The inscount0 PIN tool was ran against the 621\_wrf\_s and 623\_xalanchbmk\_s benchmarks both with the train input set. The data for the experiment can be found in the table below.

**Table 1. Instruction Count for SPEC Benchmarks via Intel PIN Tool**

|  |  |
| --- | --- |
| Benchmark | Total Instructions Executed |
| 621\_wrf\_s train | 583459817340.00 |
| 623\_xalanchbmk\_s train | 257156226730.00 |

# Problem 2

Modify the inscount0.cpp to count the number of basic blocks, number of memory reads, number of memory writes, and the total number of executed instructions. Refer to Figure 1 for the coding modifications to achieve the goal of this problem statement.



**Figure 1. Code Modifications for Problem 2**

It should be noted that the accuracy of this implementation might not be as expected because of the way that the PIN application interprets certain instructions. The total number of basic blocks can be skewed because of this interpretation, another solution that could be implemented would be to check if the current instruction is a branch and count that as the end of the current basic block.

This PIN tool works with single-threaded applications, in this case there were two simple applications and two SPEC benchmarks tested. The benchmarks consist of the following:

1. Accumulating Array Elements with (256x256) input
2. Serial Matrix Multiplication with (1x256) input
3. SPEC Benchmark 621.wrf\_s with train input
4. SPEC Benchmark 623.xalanchbmk\_s with train input

The results from the PIN tool executions can be found in the table below

**Table 2. Modified PIN Tool Results for SPEC and Simple Benchmarks**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Benchmark | Basic Blocks | Memory Reads | Memory Writes | Total Instructions |
| mm\_mult\_serial 256x256 | 3104.00 | 3816.00 | 2559.00 | 482639296.00 |
| accumulate array 1x256 | 4619.00 | 5634.00 | 4632.00 | 1421192.00 |
| 621\_wrf\_s train | 104862.00 | 260269.00 | 251753.00 | 583459817334.00 |
| 623\_xalanchbmk\_s train | 37987.00 | 53881.00 | 39434.00 | 257156226729.00 |

It is clear to see that the SPEC benchmarks overall have the greatest amount of each category, which makes sense because they are created to test a system’s performance thoroughly.

# Bonus

Count and print the statistics for each thread in a separate column for a multithreaded program. This will require modifying the pin tool from problem 2 and making it work with multi-threaded applications. Note: that the pin tool will not generally work with pthreads or other multithreaded paradigms, there is a pin tool thread api that is provided and the code written for this problem should make use of that.

# Problem 3

Design and implement a PIN instrumentation tool for profiling dynamic basic blocks (or streams) in a program. A dynamic basic block is defined as a sequential run of instructions that starts with an instruction that is a target of a taken branch and ends with the first taken branch in a sequence.

# Appendix

Command to build and then run inscount0

Make: make obj-intel64/inscount0.so

Instruction API reference for PIN can be found here:

# <https://software.intel.com/sites/landingpage/pintool/docs/81205/Pin/html/group__INS__BASIC__API.html>