Project Proposal

Simple Performance Analysis Tool

CPE 631

KYLE RAY

March 12, 2018

Contents

Executive Summary 2
Introduction 2
Project Layout 2
Testing and Verification 3
Environment 3
Goals 4

Executive Summary

In this project, I would like to utilize the open source project known as LIKWID (**L**ike **I** **K**new **W**hat **I** **W**as **D**oing) as a lightweight analysis tool backend to power a simple and easy to user Graphical User Interface (GUI) that would allow a software developer to easily gather performance information on their application in order to help with performance optimization.

Introduction

In this day and age, we have witnessed the end of Moore's Law. It is no longer the heavenly dream of the software developer to wait on the next great hardware update to provide their applications with out of the box speed ups. It is now up to the software developer to harness/exploit the power of current hardware via software optimization techniques. While there exists a plethora of software profiling and analysis tools to provide the developer with useful information on how to optimize their code, most of these tools have a steep learning curve.

I would like to create a simple user interface that would provide a software developer with some initial analysis information on their application to kickstart the optimization process. If, after this initial step, they require more optimization changes then it might be wise to invest time and resources into a more strenuous profiling and analysis tool. The tool would use the lightweight opensource tool known as LIKWID to gather information about the platform architecture as well information on the application under test.

Project Layout

The project will comprise of a simple GUI that the user can load their application into and choose from a selection of options. Refer to the figure below for a prototype of what the application might look like.

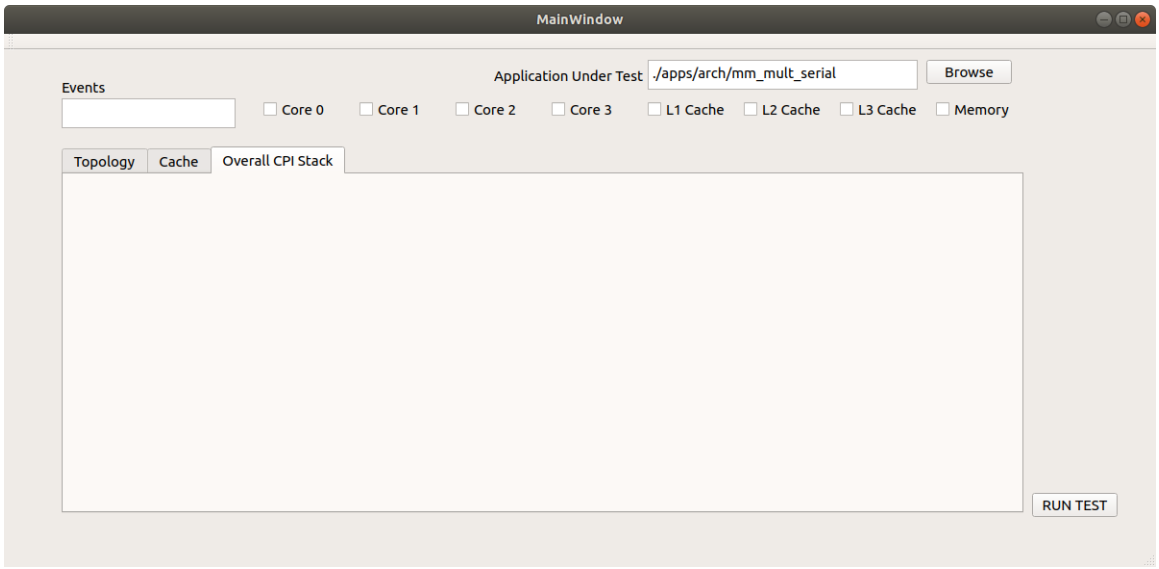

Figure 1. Prototype of Graphical User Interface for the Tool

The application would offer the user choices on what information that they would like to see. Events would include likwid-perfctr events that could possibly be measured, allowing the user to choose from a list of options. Once the choices were made the application run the corresponding LIKWID tools, parse the output with another tool (python, perl, sed/awk, etc.) and provide the information to the user. Currently there are three tabs in the graphical area, which could show the user the current architectural topology, cache information which could include cache misses, and if possible display the overall CPI stack. I will also incorporate a configuration saving mechanism that will allow the user to save their current selected options to later be loaded and ran again. Time permitting, I would like to extend this to work with threaded applications such as MPI, OpenMP, and pthreads.

NOTE: This is the first prototype and is subject to many changes throughout the development process.

Testing and Verification

The application will have a small subset of applications that will be used for unit tests. It will also be tested on available SPEC 2017 benchmarks.

It should also be noted that there are some security restrictions that might impede testing on systems where hardware counters and other necessary items will not be able to be accessed by non-root users.

Environment

The tests would be carried out on the machines that are readily available.

1.) Home Desktop – Intel i7-4790k Haswell with Hyper Threading
    a. Ubuntu 17.10.1 or 16.04.4 LTS depending on issues.
2.) Laptop – Intel i7-7500U Kaby Lake (LIKWID may not support all features)
    a. Ubuntu 17.10.1 or 16.04.4 LTS depending on issues.
3.) UAH Lab Computers and/or LaCASA if available

Goals

The overall goals of this project are as follows:

1.) To become more familiar with analysis tools such as LIKWID and performance analysis of computer architectures.
2.) Design a tool that would be readily accessible by a software developer and provide them with initial information and a visual way to see performance metrics of their applications, and possible insight into increasing overall performance.
3.) Increase overall knowledge of computer architectures and the synergy between hardware and software.