EEL5764: Computer Architecture

Project Proposal

Names of team members: Harshit Raj

Nitesh Bakhati

Dev Hitesh Parmar

Kanak Sharma

Project Title: Performance Analysis of X86 Multi-Core Architectures

Project Description: This project aims to conduct a depth analysis of the scalability of parallel algorithm performance on multi-core processor architectures. The core of the project involves using the gem5 simulator in order to simulate an X86 system with a variable number of CPU cores and a detailed cache hierarchy.

We will utilize a computational intensive workload with a parallel merge sort algorithm implemented in C++ with OpenMP to study the impact of critical architectural parameters on performance.

Our primary research questions that we are attempting to find the answers for are:

- What is the effect of increasing the number of CPU cores in the execution time and total simulation ticks for a parallelizable workload?
- What impact does the modification of L1 and L2 cache parameters (size and associativity) have on overall performance and memory access patterns?

Proposed Approach

Our methodology will follow the following steps:

Environment Setup: We will begin by setting up the gem5 simulation environment which includes building the simulator for the X86 instruction set architecture on a Linux-based platform (we will be using ubuntu).

Workload Development: We will execute and execute a parallel merge sort algorithm in C++ and the OpenMP API. The program will be compiled as a static binary to ensure compatibility with gem5's Syscall Emulation (SE) mode.

Simulation and Experimentation: We will use gem5's Python based configuration scripts to define our simulated machine. We will run two major sets of experiments:

- Core Scaling Study: Fixed cache parameters and simulation with variable number of CPU cores. (2, 4, 8, and 16 cores)
- Cache Sensitivity Study: Fixed number of cores and variable L1 data and instruction cache associativity to measure the effect on the performance.

Data Collection and Analysis: For each simulation run, we will collect the crucial performance metrics that includes total execution time, total simulation clock cycles and overall hit/miss rates for the L1 and L2 caches.

Visualization: We will use Python with the Matplotlib and other libraries to process the collected data and generate graphs that visualize the relationships between core count, execution time, and simulation ticks.

Team Members Expertise and Work Distribution

The project tasks distribution among the team members for the project is as follows:

Harshit: Development of the C++ workload, focusing on the implementation and optimization of the parallel merge sort algorithm using OpenMP.

Nitesh: Environmental setup of gem5, configuration of simulation scripts, and developing the automation script for running experiments.

Dev: Responsible for running the cache experiments, collecting all simulation data.

Kanak: Final data analysis, developing script for all plots and graphs for visualization.

Demonstration Plan and Proposed Schedule

We will demonstrate our project by executing our automation script to launch a gem5 simulation. Showing the generated output files and explaining the key metrics. Generating the final graphs that illustrate the performance scaling trends of the performance that we discovered. Submitting a comprehensive final report and presentation describing our methodology, results, and conclusions.

Week	Task
Week 1	Set up gem5 environment, compile and test C++ workload.
Week 2	Develop and test the simulation automation script.
Week 3	Develop and run all core scaling experiments and collect data.
Week 4	Develop and run all cache sensitivity experiments and collect data.
Week 5	Complete data analysis and generate all final plots.
Week 6	Prepare the final report and record the presentation.