

ENCE464 Assignment 2: Computer Architecture

Group 13: Jack Duignan, Isaac Cone, Daniel Hawes

The result of running the completed program over a range of cube sizes for 300 iterations using 20 threads can be found in Figure 1.

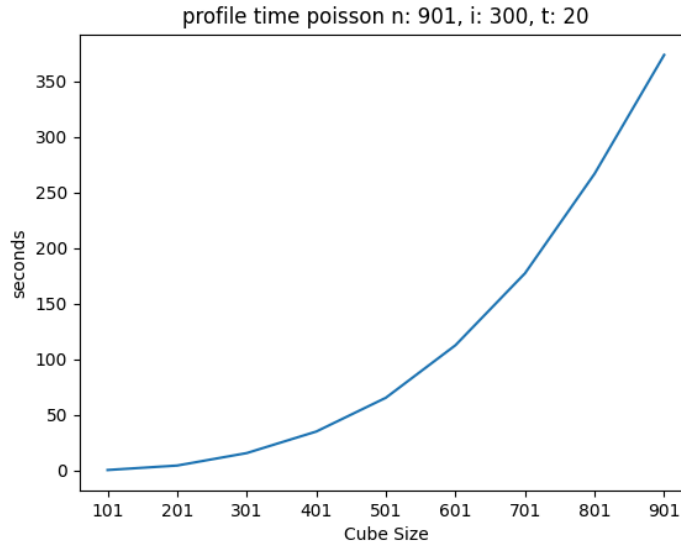


Figure 1: A complete run of the firmware across all cube sizes with 300 iterations and 20 threads.

Architecture Overview

The Central Processing Unit (CPU) described in this section is the AMD Ryzen 9 6900HX. Released in 2022, this CPU has 8 identical cores with 2 threads per core for a total of 16 logical cores. This architecture is shown in Figure 2 and will be described in further depth.

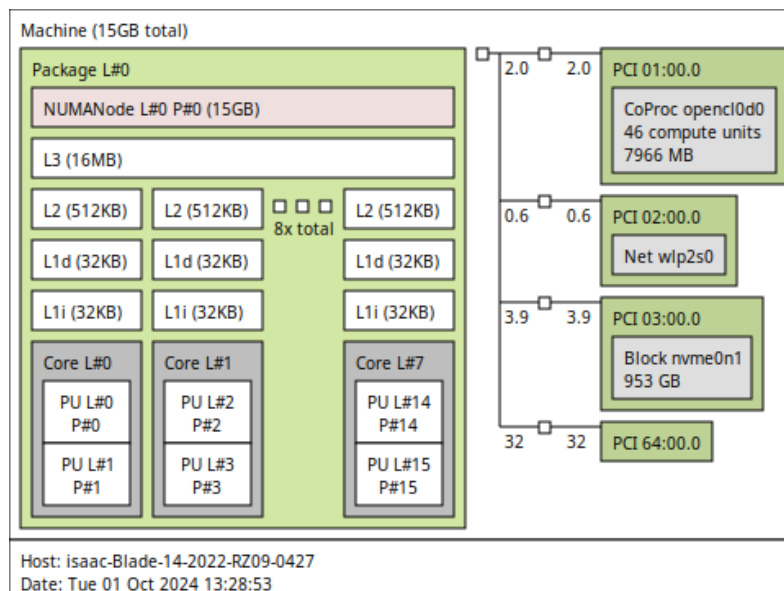


Figure 2: Central Processing Unit (CPU) architecture for the x86 AMD Ryzen 9 6900HX.

Cores

Memory Caching

Architecture Overview - hard - Isaac

big picture

ALU

FPL

Cache

Intruction decodeing etc.

Multithreading - easy - Daniel

row selection memcpy barrier

Cache - hard

Profiling - easy - Jack

Profiling was used throughout all stages of this projects development. This was done to identify which areas of the program where the slowest and how often these slow areas where called. From these results optimisations where made to the code to reduce execution time. When selecting areas of the code to optimise the sections called most often were prioritised as these give a larger performance benefit then optimising slower less frequent functions. To make profiling easier the various components of the code where compartmentalised into functions, while this does add some execution time (due to stack overheads) it allows the profiling tool gprof to provide more granular results.

Profiling was conducted on both optimised and non-optimised code to gain a wholistic understanding of the programs execution. The non-optimised program was profiled and used in the initial development stage and once the program was at an acceptable level profiling switched to using the optimised code as this was the more efficient source. A breakdown of the execution times and call counts for a non-optimised run of the program with a 201 node cube over 300 iterations using 20 threads can be seen in Table 1. The result of profiling using O3 optimised code on the same cube size as before can be found in Table 2.

Table 1: GProf results for a non-optimised run of the program with 201 nodes 300 iterations and 30 threads.

Function	Call Count	Time per call (ms)
poisson_iteration_inner_slice	5965	1.25
memcpy_3D	5977	0.61
apply_von_neuman_boundary_slice	5956	0.05
Barrier waits cumulative	11945	0
Setup	0	0

Table 2: GProf results for a O3 optimised run of the program with 201 nodes 300 iterations and 30 threads.

Function	Call Count	Time per call (us)
poisson_iteration_inner_slice	5958	676.40
memcpy_3D	5971	410.32
apply_von_neuman_boundary_slice	5926	37.12
Barrier waits cumulative	N/A	N/A
Setup	0	0

The results found in Table 1 and 2 show that in both runs the largest time cost is the iteration over the inner slice of the cube. This is expected as this the largest iteration over the nodes in the cube in the program. The next highest is the memcpy that occurs at the end of each iteration. Third is the application of the Von Neuman boundary. In earlier iterations of the program the Von Neuman boundary was called at every inner loop of the main poisson iteration. Based on profiling the team was able to identify this as a bottle neck and move the function to its own self contained iteration that only is called once.

One interesting finding of the profiling was how little time is spent at the synchronisation barriers in the code. The team was originally concered that these will cause large delays in the program as

different threads took longer to execute. By using profiling this was found to not be the case and thus didn't need to be optimised.

- Python script
- gprof outputs and how they were used

Compiler Optimisation - easy

Individual Topic 1 Jack Duignan - Branch Prediction

- What is branch Prediction
- Branch prediction errors
- Changing of program
- Results of changing

Individual Topic 2 Isaac Cone - GPU

Individual Topic 3 Daniel Hawes - SIMD

References