

Homework #2

cpe 631

Kyle Ray

February 5, 2018

Contents

[Purpose 2](#_Toc505460612)

[Setup 2](#_Toc505460613)

[Problem 1 2](#_Toc505460614)

[Program Description 2](#_Toc505460615)

[Part A Results 2](#_Toc505460616)

[Part B Results 3](#_Toc505460617)

[Problem 2 5](#_Toc505460618)

[Program Description 5](#_Toc505460619)

[Results 5](#_Toc505460620)

[Problem 3 6](#_Toc505460621)

[Program Description 6](#_Toc505460622)

[Appendix 6](#_Toc505460623)

# Purpose

To demonstrate the differences in performance between gcc or g++ and the intel provided compiler icc. Also, to become familiar with timing analysis and other tools such as the Performance Application Programming Interface (PAPI).

# Setup

The programs for this homework assignment were created and executed on the Blackhawk machine located at the University of Alabama in Huntsville.

# Problem 1

In this problem I did make use of C++ and the g++ compiler to create the matrix multiplication problem; therefore, I will be comparing the g++ and icc compilers.

## Program Description

C++ application that multiplies two squared matrices A and B of size N (NxN) doubles. The program accepts the size of the matrices via a command line argument, then fills the matrices with random values and proceeds to calculate the resulting multiplication. The NxN resulting matrix is then written to a binary file.

## Part A Results

The tables below contain the results for running the matrix multiplication program with different sizes of N consisting of 512, 1024, 1536, and 2048. Table 1 houses the results for program execution using the g++ compiler with the -O3 optimization and contains the results for using the icc compiler with the -fast optimization.

**Table 1. Execution Times for g++ and icc Matrix Multiplication**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Matrix Size | 512 | 1024 | 1536 | 2048 |
| Execution Time (g++) in seconds | 0.251 | 2.343 | 19.37 | 143.494 |
| Execution Time (icc) in seconds | 0.216 | 0.779 | 2.602 | 6.691 |

**Figure 1. Execution Times for Each Compiler**

Comparing the two tables it is easy to see that with a square matrix of 512 x 512 both compilers have roughly the same execution time; however, the intel compiler completely out performs the g++ version for the remainder of the test scenarios. The results do make sense given that the Blackhawk machine is using an Intel(R) Xeon(R) CPU E5-2643 0 @ 3.30GHz processor and the compiler would be more optimized towards intel processors than would g++.

## Part B Results

The tables below contain the data collected when measuring the execution time of the critical loop, the part of the code that is performing the matrix multiplication.

**Table 2. Critical Loop Execution Times for g++ and icc Matrix Multiplication**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Matrix Size | 512 | 1024 | 1536 | 2048 |
| Execution Time (g++) in seconds | 0.23 | 2.28 | 18.77 | 142.87 |
| Execution Time (icc) in seconds | 0.09 | 0.72 | 2.49 | 5.87 |

**Figure 2. Critical Loop Execution Times for Each Compiler**

Again, the intel compiler has better performance over the g++ compiler. It is interesting that regarding the g++ compiler the overall execution time is not that much larger than the critical loop execution time, in the case of matrix size 512 0.251 vs 0.23; however, the icc compiled version overall execution times hold a greater difference from the critical loop measurements, in the case of matrix size 512 0.216 vs 0.09.

# Problem 2

## Program Description

Modified the existing matrix multiplication program from problem 1 to utilize OpenMP to parallelize the critical loop to increase overall performance.

## Results

The table below shows the results for critical loop execution time for both the g++ and icc compilers using OpenMP.

**Table 3. OpenMP Critical Loop Execution Times for g++ and icc Matrix Multiplication**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Matrix Size | 512 | 1024 | 1536 | 2048 |
| Execution Time (g++) in seconds | 0.144 | 0.473 | 1.405 | 35.625 |
| Execution Time (icc) in seconds | 0.163 | 0.346 | 1.055 | 35.333 |

**Figure 3. Critical Loop Execution Times for Each Compiler with OpenMP**

It is interesting that utilizing OpenMP to parallelize the critical loop greatly benefitted the performance of the g++ version of the program, which was expected, yet seemed to hinder the performance of the icc version. It seems that the overhead associated with introducing OpenMP to the icc compiled version decreased the performance significantly. In the end, the OpenMP implementation results in nearly identical execution times for both compilers. In regards execution time as the performance metric the serial icc version is still in the lead.

# Problem 3

## Program Description

Implement PAPI interface calls to measure number of instructions executed, total number of clock cycles, level 1 data cache misses, and the total number of L2 misses in the critical loop.

## Results

# Appendix

Add anything else that might be pertinent to the assignment.