CS 301
High-Performance Computing

Lab 2 - Q1A

Problem A-1: PI_TRAPEZOIDAL

Aditya Nawal (202001402)
Divya Patel (202001420)

February 8, 2023

Contents

1 Introduction

This report evaluates the performance of integration using the Trapezoidal rule, with a focus on its run-time. This report describes the hardware specifications, compiler, and optimization flags used in the experiments, as well as the input parameters, output, and accuracy checks. Additionally, we present the results of our performance evaluation, including plots that compare the run-time of the algorithm against the problem size.

2 Hardware Details

2.1 Lab 207 PC

- Architecture: x86_64
- CPU op-mode(s): 32-bit, 64-bit
- Byte Order: Little Endian
- CPU(s): 4
- On-line CPU(s) list: 0-3
- Thread(s) per core: 1
- Core(s) per socket: 4
- Socket(s): 1
- NUMA node(s): 1
- Vendor ID: GenuineIntel
- CPU family: 6
- Model: 60
- Model name: Intel(R) Core(TM) i5-4590 CPU @ 3.30GHz
- Stepping: 3
- CPU MHz: 3300.000
- CPU max MHz: 3700.0000
- CPU min MHz: 800.0000
- BogoMIPS: 6585.38
- Virtualization: VT-x
- L1d cache: 32K
- L1i cache: 32K

- L2 cache: 256K
- L3 cache: 6144K
- NUMA node0 CPU(s): 0-3
- Flags: fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs bts rep_good nopl xtopology nonstop_tsc aperfmperf eagerfpu pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 fma cx16 xtpr pdcm pcid sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm abm epb invpcid_single tpr_shadow vnmi flexpriority ept vpid fsgsbase tsc_adjust bmi1 avx2 smep bmi2 erms invpcid xsaveopt dtherm ida arat pln pts

2.2 HPC Cluster

- Architecture: x86_64
- CPU op-mode(s): 32-bit, 64-bit
- Byte Order: Little Endian
- CPU(s): 16
- On-line CPU(s) list: 0-15
- Thread(s) per core: 1
- Core(s) per socket: 8
- Socket(s): 2
- NUMA node(s): 2
- Vendor ID: GenuineIntel
- CPU family: 6
- Model: 63
- Model name: Intel(R) Xeon(R) CPU E5-2640 v3 @ 2.60GHz
- Stepping: 2
- CPU MHz: 1976.914
- BogoMIPS: 5205.04
- Virtualization: VT-x
- L1d cache: 32K
- L1i cache: 32K

- L2 cache: 256K
- L3 cache: 20480K
- NUMA node0 CPU(s): 0-7
- NUMA node1 CPU(s): 8-15

3 Problem A1

3.1 Description of the problem

Problem A1 is about integrating a function using the Trapezoidal rule. The goal is to implement an algorithm that accurately calculates the integration of a given function using the Trapezoidal rule, and to verify the accuracy of the code by using it to calculate the value of pi. This problem involves the application of mathematical concepts and numerical methods to solve a real-world problem.

3.2 Serial Complexity

The serial complexity of Problem A1, which involves integrating a function using the Trapezoidal rule, is O(n), where n is the number of subintervals used in the calculation. This means that as the number of subintervals increases, the time it takes for the algorithm to complete will increase linearly with n. This complexity is considered linear because the algorithm requires a single pass through the data to calculate the integration. The serial complexity of an algorithm provides an estimation of the time it takes for the algorithm to complete as the size of the input increases.

3.3 Profiling Information

The algorithm uses the Trapezoidal rule for numerical integration. The integration is performed using a function integrate, which takes as input the interval $[a, b]$ and the number of subintervals n. The function function is used to calculate the value of the integrand at a given point x.

```
Flat profile:

Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls  ms/call  ms/call  name
100.79      0.03     0.03 10000001     0.00     0.00  function.4396
  0.00      0.03     0.00        2     0.00     0.00  diff
  0.00      0.03     0.00        1     0.00    30.24  integrate.4401

 %         the percentage of the total running time of the
time       program used by this function.

cumulative a running sum of the number of seconds accounted
 seconds   for by this function and those listed above it.

 self      the number of seconds accounted for by this
seconds    function alone.  This is the major sort for this
           listing.

calls      the number of times this function was invoked, if
           this function is profiled, else blank.
```

Figure 1: Profiling on Lab 207 PC using gprof

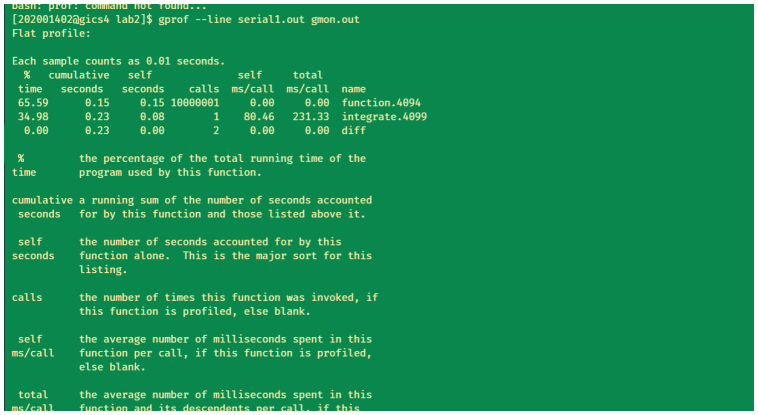

Figure 2: Profiling on HPC cluster using gprof

3.4 Optimization Strategy

Optimization strategy that we used was to use compiler optimization flags, such as the -O3 flag in GCC compiler. This flag enables the compiler to perform aggressive optimization on the code, such as inlining functions, removing unused variables, and reordering instructions for better performance. Using this flag can result in improved performance and faster execution time for the algorithm.

3.5 Graph of Problem Size vs Runtime

3.5.1 Graph of Problem Size vs Runtime for LAB207 PCs

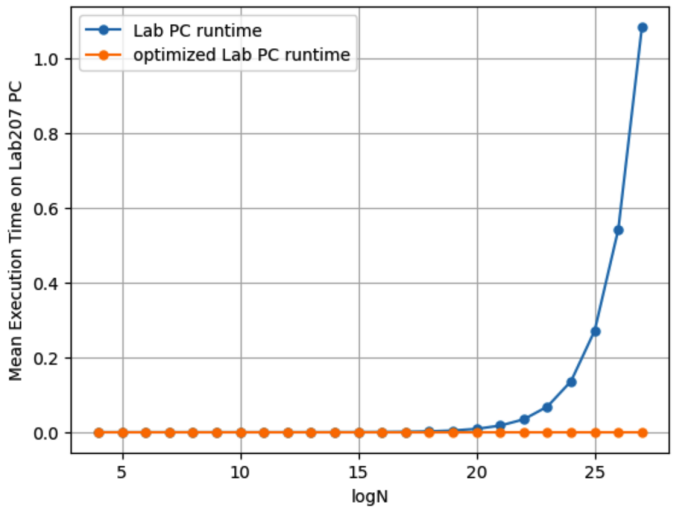

Figure 3: Total mean execution time vs Problem size plot for optimised and non-optimised code (Hardware: LAB207 PC). Normal code takes much longer time to run than pragma-optimized code.

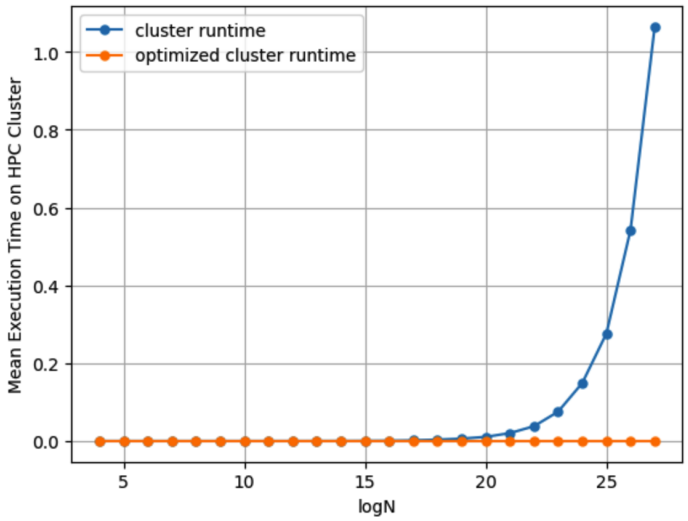

Figure 4: Total mean execution time vs Problem size plot for optimised and non-optimised code (Hardware: HPC Cluster). Normal code takes much longer time to run than pragma-optimized code.

3.5.2 Graph of Problem Size vs Runtime for HPC Cluster

3.6 Discussion

The graph shows that the run-time of the algorithm increases linearly with the problem size (exponentially with respect to log of problem size). This is expected because the complexity of the algorithm is O(n), where "n" is the size of the arrays. The run-time of the algorithm on the HPC cluster is almost equal to the run-time on the lab PC, which is expected because though HPC cluster has more cores, our algorithm is not parallelized and thus the performance does not increase with the number of cores and single core performance is almost the same on both machines.