Logo, company name

Description automatically generated

DOCUMENTATION

CPU benchmark

Student: Paul Petruț-Betuel

Group: 30421

# CONTENTS

1. Assignment Objective ……………………………………………………….3
2. Problem analysis, modeling, scenarios, use case ……………….3
3. Design ………………………………………………………………………………………..6
4. Implementation ..……………………………………………………………………….7
5. Conclusion ………………………………………………………………………………….8
6. Bibliography ………………………………………………………………………………..8

# Assignment Objective

The main goal of this assignment is to develop a set of benchmark programs to assess the capabilities of a multicore processor.

# Problem Analysis, Modeling, Scenarios, Use Cases

The analysis focuses on understanding the challenges and opportunities presented by multicore processors. Scenarios and use cases involve exploring the potential benefits and performance implications of parallel processing.

In the analysis phase, we explore the intricate landscape of multicore processors, identifying key challenges and opportunities. The modeling process involves creating abstractions that capture the essence of parallel processing complexities. Scenarios and use cases further enrich our understanding by simulating real-world situations. For instance, we can gauge the potential benefits and performance implications of deploying parallel algorithms on multicore architectures.

Modeling:

For instance, creating mathematical models enables us to predict the scalability of algorithms across varying numbers of cores. These models help us grasp the theoretical limits and trade-offs inherent in multicore architectures, providing a foundation for informed decision-making during the design and implementation phases.

Scenarios:

Scenarios extend our analysis by simulating diverse operational conditions. These can include scenarios where specific cores may autonomously handle higher workloads or situations where tasks with varying computational intensities coexist. By exploring these scenarios, we gain a nuanced understanding of how multicore processors autonomously distribute and manage workloads, uncovering potential bottlenecks or areas for optimization.

Use Case:

A compelling application scenario involves harnessing the Bailey–Borwein–Plouffe (BBP) formula to meticulously calculate a predetermined number of digits of pi. This not only serves as a captivating showcase of the intricate technicalities of parallel processing but also underscores its profound practical relevance in scientific computation scenarios, where achieving both precision and speed is paramount.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Nr of threads\work | 1000 | 5000 | 50000 | 500000 | 1000000 | 10000000 |
| 1 | 1 | 1 | 7 | 62 | 122 | 1262 |
| 2 | 1 | 1 | 4 | 37 | 72 | 687 |
| 5 | 1 | 1 | 3 | 19 | 36 | 343 |
| 10 | 1 | 2 | 7 | 13 | 23 | 265 |
| 50 | 1 | 4 | 4 | 11 | 20 | 192 |
| 100 | 2 | 6 | 6 | 11 | 19 | 190 |

\*Time is measured in milliseconds.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Nr of threads\work | 1000 | 5000 | 50000 | 500000 | 1000000 | 10000000 |
| 1 | 100 | 500 | 714 | 806 | 819 | 792 |
| 2 | 50 | 250 | 500 | 675 | 694 | 727 |
| 5 | 20 | 100 | 333 | 500 | 555 | 583 |
| 10 | 5 | 25 | 250 | 384 | 434 | 425 |
| 50 | 1 | 2 | 25 | 90 | 100 | 104 |
| 100 | 0 | 0 | 8 | 45 | 52 | 52 |

\*This score is given by this formula: (iteration / number of threads) / (10 \* Time).

Performance Analysis:

Conducting a comprehensive performance analysis with varying numbers of threads and workload iterations provides valuable insights into the dynamics of parallelization. The tabulated results reveal a nuanced tradeoff between the number of threads employed and the time required to complete the computational tasks. Additionally, the efficiency of thread utilization exhibits a tradeoff concerning the workload. Intriguingly, the analysis exposes that, for the specific configuration on my computer, the optimal workload per thread lies within the range of 1,000,000 to 10,000,000 iterations, beyond which efficiency experiences a diminishing return.

Insights:

From these experiments, it becomes evident that the choice of thread count and workload significantly influences the overall performance, emphasizing the need for a balanced configuration. The findings highlight the intricate interplay between computational resources and workload distribution in a multicore environment.

Optimization Considerations:

These observations prompt consideration for optimization strategies, suggesting that a careful balance between thread count and workload is crucial to achieving optimal performance. Furthermore, recognizing the saturation point for a single thread's efficiency provides valuable guidelines for workload allocation in scenarios where computational resources are finite.

# Design

In the design phase, our focus extends to planning the intricate architecture of the benchmark programs. This involves translating insights from the modeling phase into practical strategies for effective parallelization. Strategies encompass load balancing mechanisms, synchronization techniques, and memory management schemes tailored to the multicore environment.

# Implementation

The implementation phase involves the actual coding of the benchmark programs. The student has chosen to compute a specific number of digits of pi using the Bailey–Borwein–Plouffe (BBP) formula. This formula is known for its efficient computation of individual digits of pi.

Code Overview

The provided C++ code implements a parallelized benchmark program using the Bailey–Borwein–Plouffe (BBP) formula to calculate a specific number of digits of pi. The program takes user input for the number of threads and precision (number of iterations) to perform parallel computations.

*void* ***calculateBBP****(int thread\_id, int num\_threads, int num\_iterations, double& partial\_sum) {*

*partial\_sum = 0.0;*

*for (int k = thread\_id; k < num\_iterations; k += num\_threads) {*

*double term = (1.0 / std::pow(16, k)) \* (4.0 / (8 \* k + 1) - 2.0 /*

*(8 \* k + 4) - 1.0 / (8 \* k + 5) - 1.0 / (8 \* k + 6));*

*partial\_sum += term;*

*}*

*}*

Function Purpose: Computes the partial sum for a given thread based on the BBP formula.

Parameters:

* thread\_id: The ID of the current thread.
* num\_threads: The total number of threads.
* num\_iterations: The total number of iterations for precision.
* partial\_sum: Reference to the variable storing the partial sum.

# Conclusions

In conclusion, our exploration into developing benchmark programs for multicore processors, using the BBP formula to compute pi digits, revealed intricate tradeoffs. The performance analysis highlighted the delicate balance between thread count, workload, and efficiency, showcasing optimal performance within a specific iteration range. These insights underscore the importance of thoughtful configuration in achieving peak multicore processor performance, providing valuable guidance for future optimizations and real-world workload allocation.

# Bibliography

The references that were consulted by the student during the implementation of the homework:

1. Lecture notes
2. Lab notes
3. https://mihai.utcluj.ro/computer-architecture/
4. <https://en.wikipedia.org/wiki/Bailey%E2%80%93Borwein%E2%80%93Plouffe_formula>
5. https://www.piday.org
6. https://mathworld.wolfram.com/BBPFormula.html
7. <https://www.geeksforgeeks.org/>
8. <https://stackoverflow.com/>