Individual Assignments: Matrix Multiplication

Big Data

Grado en Ciencia e Ingeniería de Datos
Universidad de Las Palmas de Gran Canaria

This document outlines the individual assignments for the Big Data course. The main topic of the assignments will be matrix multiplication, a fundamental operation in many computational applications, especially in the field of Big Data and high-performance computing. Students are required to conduct comparative studies (benchmarks) of different approaches to solve the matrix multiplication problem, focusing on various matrix sizes and at least three alternative methods (approaches) for each part of the assignment.

Benchmarking is the process of measuring and evaluating the performance of a system, algorithm, or process by comparing it. In the context of computing, benchmarking is commonly used to assess the efficiency, speed, and resource usage of algorithms or software applications under different conditions. It involves running a series of tests or tasks and collecting data, which is then analyzed to identify strengths, weaknesses, and potential areas for optimization. In the field of Big Data, benchmarking is crucial due to the scale and complexity of the datasets involved. Big Data applications often need to process massive amounts of data in real-time or near real-time, which places a significant burden on computational resources. Efficient performance is not only desirable but necessary to ensure timely results and optimal use of resources. By conducting benchmarks, students can:

- **Evaluate Performance**: Measure the speed and efficiency of different algorithms and approaches when applied to large datasets.
- **Optimize Resource Usage**: Identify bottlenecks and inefficiencies that may lead to excessive use of memory, CPU, or storage, and make informed decisions on how to optimize them.
- **Scalability Analysis**: Test how algorithms and systems perform as the size of the data increases, which is a critical factor in Big Data where datasets can scale to terabytes or even petabytes.

- **Informed Decision-Making**: Benchmark results provide valuable insights into which approaches are most suitable for a given problem, based on specific performance metrics such as execution time, memory usage, and resource efficiency.

In this course, benchmarking will be a key part of the assignments, as students will compare various matrix multiplication algorithms and approaches under different computational conditions and matrix sizes. This will help students develop a deep understanding of how different techniques perform in the context of Big Data, where performance and scalability are essential for practical applications.

Students are required to test their implementations on a range of square matrices, starting from small matrices (e.g., 10x10) to progressively larger ones (e.g., 100x100, 1000x1000, 10,000x10,000). The goal is to analyze how the performance of each approach scales as the matrix size increases, and to observe any bottlenecks or performance limitations that may arise with larger matrices.

Through these assignments, students will gain a deeper understanding of the performance implications of matrix multiplication in different computational environments, from basic algorithms to highly optimized, parallel, and distributed approaches. The goal is to enable students to apply these concepts to real-world Big Data problems, where performance and scalability are critical.

There are four main assignments, and each will be due at the end of specific weeks throughout the 15-week course.

1 Basic Matrix Multiplication in Different Languages

In this assignment, students will compare the performance of a basic matrix multiplication algorithm implemented in three different programming languages: Python, Java, and C. The goal is to analyze the efficiency of each language when handling matrix multiplication, considering factors such as execution time, memory usage, and computational overhead.

1.1 Requirements

- Implement the basic matrix multiplication algorithm ($O(n^3)$ complexity).
- Test the algorithms with increasingly larger matrix sizes.
- Use bench-marking tools.
- Measure and compare the execution time for each language.

- Consider any language-specific optimizations or features that may affect performance.

1.2 Performance Metrics

- Execution time (in seconds or milliseconds).
- Memory usage during execution.
- CPU usage (optional).

1.3 Due Date

October 6th, 2024

2 Optimized Matrix Multiplication Approaches and Sparse Matrices

For this part, students will investigate and compare different optimization techniques for matrix multiplication. Optimizations may include algorithmic improvements (e.g., Strassen's algorithm), loop unrolling, and cache optimization techniques. Additionally, students are required to explore and implement the use of sparse matrices, which can significantly reduce computational costs when most elements of the matrices are zero.

2.1 Requirements

- Implement at least two optimized versions of the matrix multiplication algorithm.
- Compare the performance of the basic approach with the optimized versions.
- Implement and evaluate the performance of matrix multiplication using sparse matrices. Analyze how the sparsity level (percentage of zero elements) affects performance.
- Test the algorithms with increasingly larger matrix sizes and report the maximum matrix size that each approach can handle efficiently.

2.2 Report

- Execution time.
- Memory usage.
- Maximum matrix size handled.
- Performance comparison between dense and sparse matrices (at different sparsity levels).
- Any observed bottlenecks or performance issues.

2.3 Due Date

November 3rd, 2024

3 Vectorized and Parallel Matrix Multiplication

In this assignment, students will compare vectorized approaches (e.g., using SIMD instructions) and parallel computing techniques (e.g., multi-threading or using libraries such as OpenMP) for matrix multiplication.

3.1 Requirements

- Implement a vectorized version of matrix multiplication.
- Implement a parallel version of matrix multiplication.
- Compare both approaches with the basic matrix multiplication algorithm.
- Test with large matrices and analyze the performance gain from vectorization and parallelization.

3.2 Metrics

- Speedup compared to the basic algorithm.
- Efficiency of parallel execution (using metrics such as speedup per thread).
- Resource usage (e.g., number of cores used, memory).

3.3 Due Date

December 1st, 2024

4 Distributed Matrix Multiplication

Students will explore distributed computing approaches to matrix multiplication. The focus here is on scalability and how distributed systems handle very large matrices that cannot fit into the memory of a single machine.

4.1 Requirements

- Implement matrix multiplication using a distributed computing framework.
- Compare the distributed approach with both the basic and parallel versions.
- Test with extremely large matrices.

4.2 Report

- Scalability (how performance changes as the size of the matrix increases).
- Network overhead and data transfer times.
- Resource utilization (e.g., number of nodes used, memory per node).

4.3 Due Date

January 12th, 2025