PARALLEL MATRIX MULTIPLICATION

CS3658
High Performance Computing

Jesswin J - 21011101050
M Arvind - 21011101067

DOCUMENTATION:

1. Problem Statement:

Given two matrices A (of size m x n) and B (of size n x p), the goal is to compute the product matrix C (of size m x p) where each element C*ij* is calculated as the dot product of the corresponding row of A and column of B: $C=A\times B$

The straightforward sequential algorithm for matrix multiplication has a time complexity of O(mnp). As the size of the matrices increases, the computation time grows significantly, making it impractical for large matrices.

2. Parallelization Approach:

Parallelizing matrix multiplication involves distributing the computation across multiple processing units to exploit parallelism and reduce the overall computation time. One common approach for parallelization is to divide the matrices into smaller submatrices and distribute the computation of each submatrix multiplication across multiple processors.

3. Algorithm Description:

- Matrix Partitioning.
- Local Computations.
- Aggregation.

4. Benifits

- Reduced Computation Time.
- Scalability.
- Resource Utilization.
- Improved Performance.

5. Example Implementations:

- Parallel matrix multiplication can be implemented using parallel programming paradigms such as MPI (Message Passing Interface), OpenMP, CUDA (for GPU acceleration), or distributed computing frameworks like Apache Spark.
- Example codes and implementations are available in various programming languages and libraries, facilitating easy integration into existing software systems.

APPROACH:

1.Understand the Problem:
- Familiarizing with matrix multiplication and its computational complexity.
- Understanding the requirements and constraints of the parallel matrix multiplication problem.
- Recognizing the potential benefits of parallel computing in speeding up matrix multiplication.

2.Choose Parallelization Technique:
- Evaluating thread-based parallelism, SIMD instructions.
- Considering factors like the nature of the problem, available hardware resources, and programming expertise.
- Selecting the most suitable parallelization technique based on your analysis.

3.Design Parallel Algorithm:
- Devising a parallel algorithm for matrix multiplication based on the chosen parallelization technique.
- Breaking down the matrix multiplication task into parallelizable units of work.

4.Implement the Algorithm:
- Choosing appropriate programming languages and frameworks for implementation.
- Writing code or pseudocode based on the parallel algorithm design.
- Utilizing parallel computing libraries.

5.Optimize Performance:
- Identify potential bottlenecks and areas for optimization in the implementation.

6.Test and Validate:
- Conduct performance testing using various input sizes and hardware configurations.

PERFORMANCE METRICS:

For matrix multiplication, we can define the following performance metrics:

1. Speedup (S): Speedup measures the improvement in performance achieved by parallelizing the matrix multiplication algorithm. It is calculated as the ratio of the execution time of the sequential algorithm to the execution time of the parallel algorithm:

S= T(sequential)/T(parallel)

Where:

- T(sequential) is the execution time of the sequential algorithm.
- T(parallel) is the execution time of the parallel algorithm.

2. Parallel Efficiency (E): Parallel efficiency measures the utilization of resources in the parallel system. It is the ratio of the speedup achieved by the parallel algorithm to the number of processors used:

E= S/P

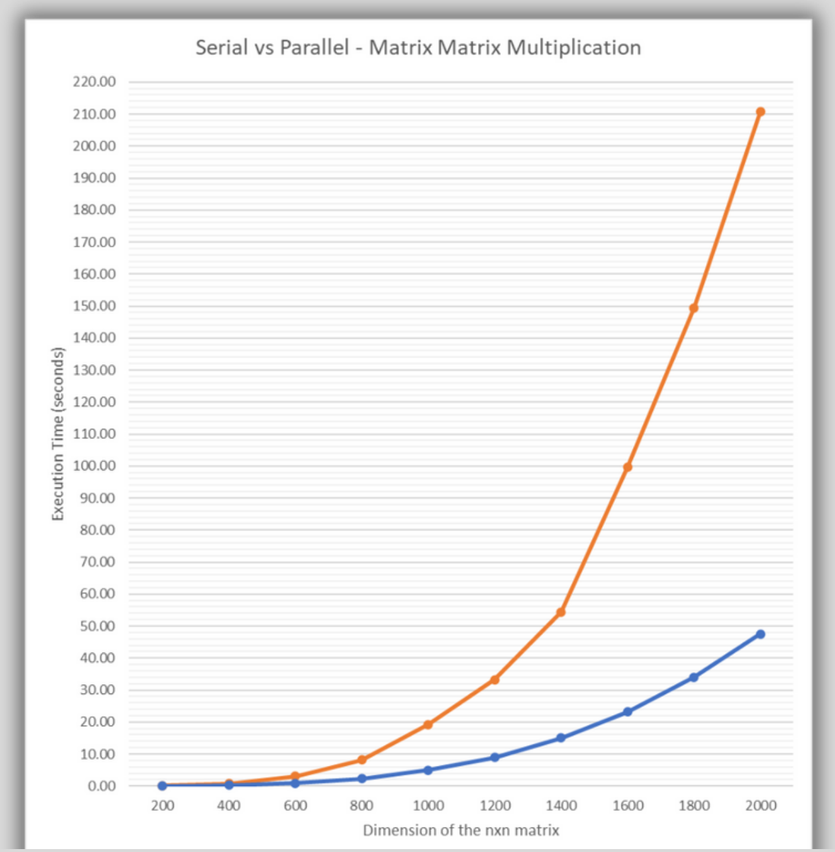
Where:

- S is the speedup achieved by the parallel algorithm.
- P is the number of processors used.

To analyze the performance of the parallel matrix multiplication algorithm, we can plot the following graphs:

1. **N vs Speedup**: This graph shows how the speedup varies with the size of the matrices (N). It helps in understanding the scalability of the parallel algorithm as the problem size increases.
2. **N vs Parallel Efficiency**: This graph shows how the parallel efficiency varies with the size of the matrices (N). It helps in understanding how effectively the parallel resources are utilized for different problem sizes.

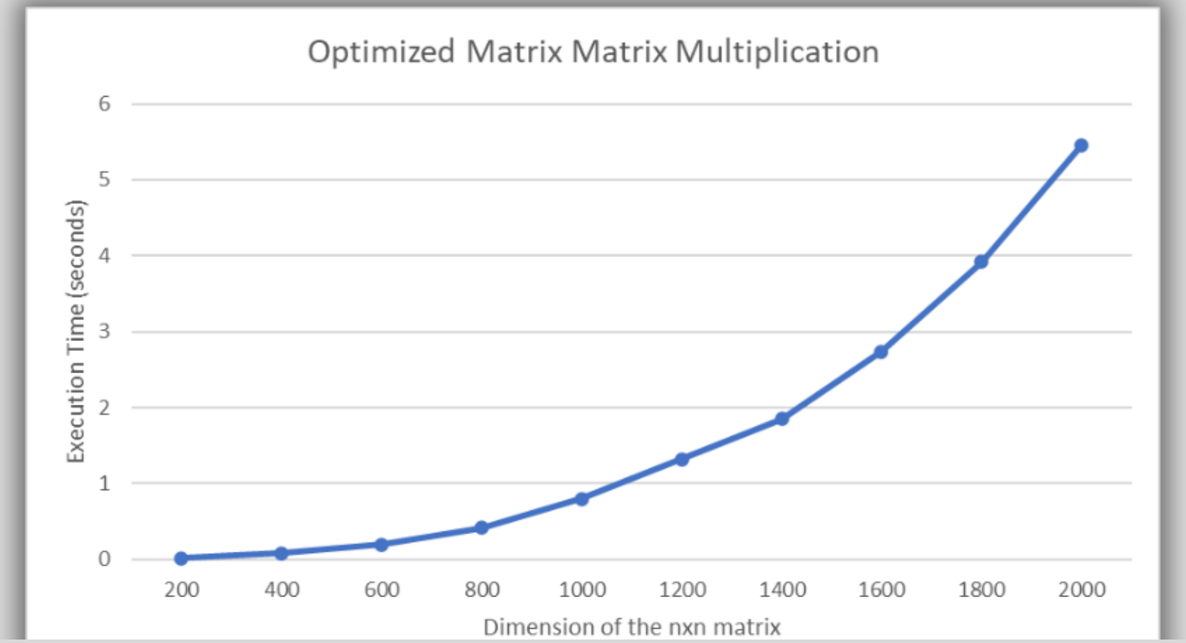
GRAPHS:



No. of threads*100

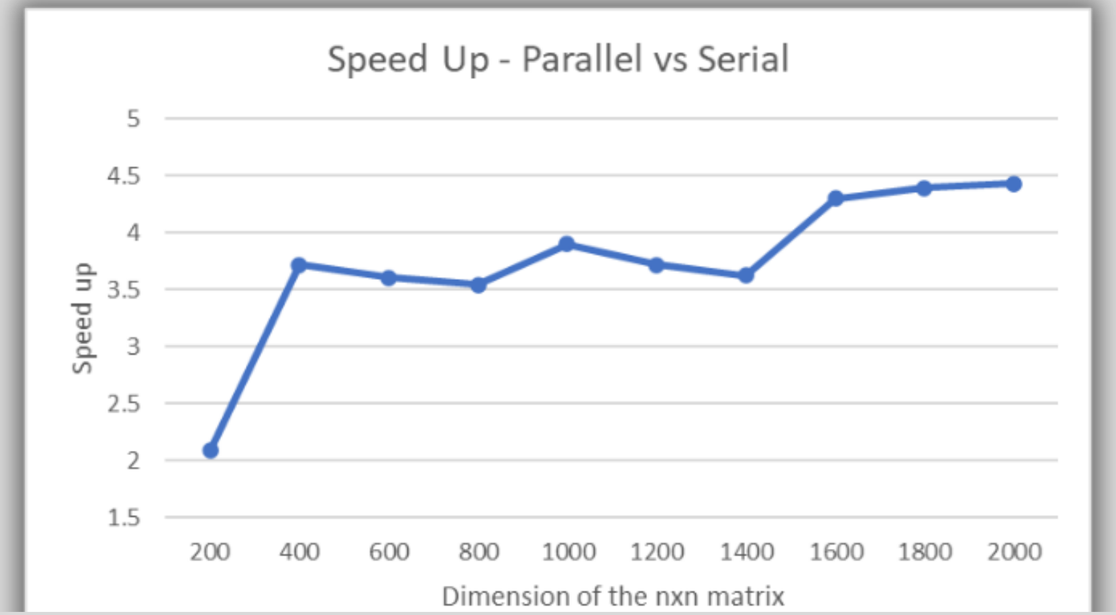
Serial Time

Parallel Time

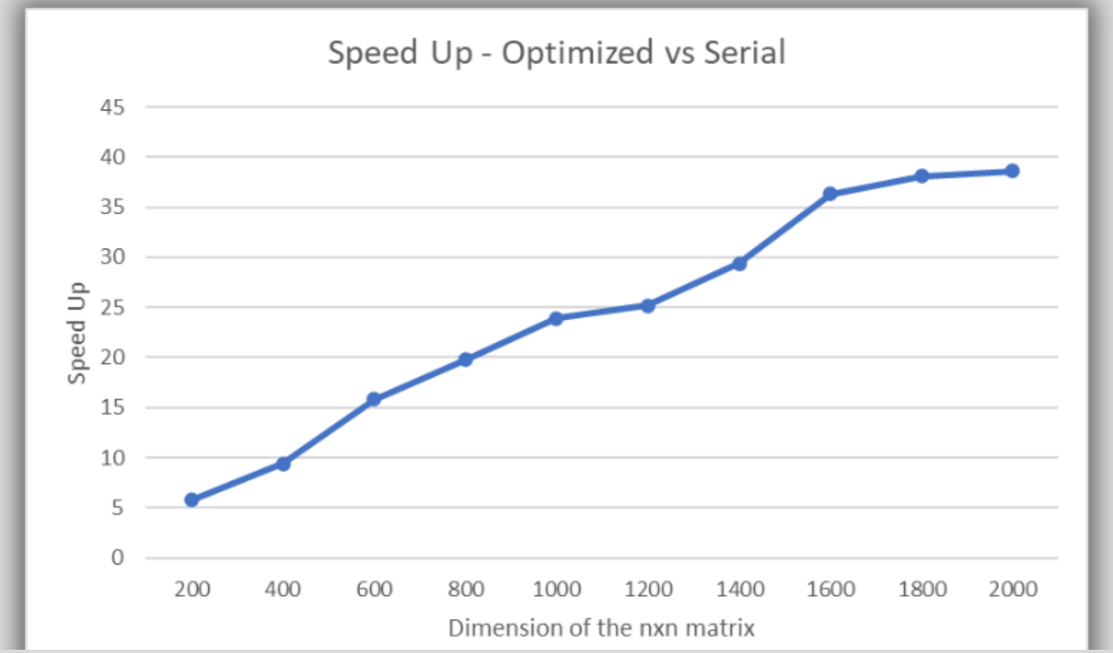


No. of threads*100

GRAPHS:(Speed up)



No. of threads*100



No. of threads*100

ADVANTAGES OF HPC

- **Speedup**: HPC platforms leverage parallel processing capabilities to distribute the workload across multiple computing units simultaneously. This results in significantly faster execution times compared to sequential.
- **Scalability**: HPC systems are designed to scale seamlessly with increasing computational demands.
- **Resource Utilization**: Parallel matrix multiplication algorithms can leverage resources effectively, minimizing idle time and maximizing throughput.
- **Large Memory Capacity**: HPC systems typically feature large memory capacities, allowing them to handle matrices of considerable size without running into memory constraints.
- **High Bandwidth Interconnects**: HPC clusters are equipped with high-speed interconnects that facilitate fast communication between computing nodes.
- **Advanced Parallelization Techniques**: HPC platforms offer access to advanced parallelization techniques and libraries optimized for parallel computing tasks.