Part 1

Fox Method and Divide-and-Conquer Strategy

The Fox Method is an optimized parallel algorithm for matrix multiplication, primarily designed for distributed memory systems. It applies the divide-and-conquer strategy by breaking down the task of matrix multiplication into smaller, manageable subtasks that can be solved independently on different processors. Here's how the Fox Method utilizes this strategy:

1. Partitioning the Matrices:

Given two matrices and , the Fox Method divides these matrices into blocks of submatrices, creating a grid of smaller matrices.   
For processors, the matrices are divided into blocks, each of size .

2. Distribution of Work:

Each processor is responsible for computing a part of the resulting matrix by working on specific submatrices.

The algorithm assigns each processor to a specific block in the grid. Processors will only handle matrix blocks they are responsible for, reducing redundant calculations and facilitating parallel computation.

3. Communication and Computation:

The Fox Method involves three main steps in each iteration: broadcasting, multiplication, and shift.

Broadcasting: Each row of processors shares its relevant blocks of matrix with the row processors.

Multiplication: Each processor computes the product of the broadcasted matrix block and its assigned block of .

Shift: The blocks of matrix are shifted to the neighboring processor in a circular pattern.

This divide-and-conquer approach ensures that each processor works with localized data, reducing communication overhead while utilizing the full processing power available.

4. Iteration:

The algorithm performs iterations to cover all block combinations, ensuring each block of and is involved in the calculation for the result matrix block assigned to each processor.

This strategy of dividing the matrix into submatrices allows the Fox Method to exploit data parallelism and task parallelism, making it efficient on distributed systems.

Work and Parallelism Analysis

In analyzing the work and parallelism of the Fox Method, we can estimate the total work and parallel work done by each processor and derive the parallelism (the ratio of work to critical path length).

1. Total Work (Sequential Work):

- For classical matrix multiplication, the work required to multiply two matrices is .

- The Fox Method, in total, performs the same amount of work as the classical algorithm (i.e., \( O(n^3) \)) since it divides the work across multiple processors without altering the total computations.

2. Parallel Work:

- With \( P \) processors (where and ), the work done by each processor involves computing the multiplication for a submatrix of size .

- Each processor’s work (submatrix multiplication) is , because each submatrix multiplication within a processor block requires operations.

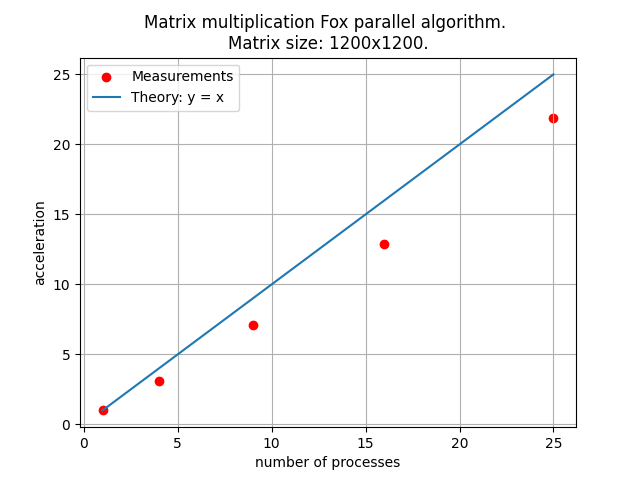
3. Parallelism:

- Parallelism is defined as the ratio of total work to the critical path length (or span).

- The critical path length for the Fox Method includes the computation of matrix multiplications and communication steps (broadcast and shift operations). Since there are communication rounds, each taking time , the span is .

- Therefore, the parallelism is:

- This shows that as increases, the Fox Method achieves greater parallelism, making it well-suited for systems with a large number of processors.

Part 2

The real speedup of the Fox algorithm in matrix multiplication is worse than the theoretical speedup due to a variety of practical limitations and overheads that are not accounted for in idealized theoretical models.

1. Communication Overhead

- Theoretical Assumption: The theoretical analysis often assumes that communication between processors is instantaneous or that its cost scales linearly and predictably with the number of processors.

- Reality: In distributed systems, communication can be a significant bottleneck, especially as the number of processors increases. Network latency, bandwidth limitations, and contention for communication channels can all increase the time taken for data exchanges. In the Fox algorithm, each step involves both broadcasting and shifting data blocks, so as \( P \) (number of processors) grows, communication costs grow faster than computation time decreases.

2. Latency and Synchronization Costs

- Theoretical Assumption: Synchronization costs are often overlooked in theoretical speedup models.

- Reality: The Fox algorithm requires each processor to synchronize with others at each step (for example, waiting for broadcasts to complete before starting calculations). This synchronization creates idle time, especially if there are variations in individual processor speeds or network latency. These delays accumulate and reduce the overall speedup, as processors spend more time waiting than computing.

3. Cache and Memory Hierarchy Effects

- Theoretical Assumption: Models often assume that data access is uniformly fast.

- Reality: Real-world memory hierarchies (caches, main memory, and possibly disk storage for very large matrices) introduce variations in data access times. When data is not in cache, memory access becomes slower. In the Fox algorithm, data blocks must frequently move between processors and between levels of the memory hierarchy. Cache misses and inefficient memory access patterns can degrade performance, especially as the matrix size grows relative to cache size.