Project High-level Description

Aaron Li, April Zhang, Catherine Gai, Susannah Su, Yixuan Qiu

Overview

Our project aims to develop a high-performance stock trading system with two core components: 1) an order processor for processing a large number of live market data and maintaining an order book consists of pending buy and sell orders, as well as a comprehensive history of matched transaction in the recent past, and 2) an efficient (and potentially automatic) order matching engine that could match buyer and seller requests and update the corresponding information in the order book. The primary goal is to optimize the system for processing a large volume of live market data and executing transactions swiftly. To achieve this, we will employ various forms of parallelism to enhance performance and scalability.

Forms of Parallelism

- Data-Level Parallelism (DLP): We plan to exploit data-level parallelism by vectorizing computations and partitioning market data for parallel processing across threads. This approach is motivated by the observation that one significant computational bottleneck is the large amount of real-time market data and processing each individual piece of order information is independent.
- Task-Level Parallelism (TLP): Our system will utilize concurrent or lock-free data structures for the order book, enabling multiple threads to access and modify it concurrently. This will reduce latency and improve throughput by minimizing locking overhead.
- Instruction-Level Parallelism: This will be compatible with our multi-processor setup, handling incoming data, matching orders, and executing transactions that will be distributed across different processors in parallel. It is likely to have an MIMD architecture, which combines both shared and distributed memory strategies.

Exploiting Parallelism

- Shared Memory Parallelism: We plan to leverage the shared memory model through multithreading (e.g. OpenMP) for efficient data access and exchange among threads working on separate parts of the problem in parallel.
- Distributed Memory Parallelism: We will employ frameworks like MPI to facilitate task distribution and communication between nodes for scalability.

Problem Size and Memory Requirements

The data processing and computations within this trading system are not memory-expensive, and we plan to evaluate its performance with a few GBs of market data, so the academic cluster is expected to provide sufficient resources for this project. Since the primary focus of this project is not to incorporate additional functionalities (limited to two core components), the simulation is anticipated to be fast, with an expected completion time within minutes, based on the performance reported by similar applications.