High-Frequency Trading (HFT) System

Requirements Overview

1. Low Latency: The system must execute trades and update orders with **minimal delay**. Order Placement: **Rapidly** place buy and sell orders based on market conditions. Order Cancellation: **Quickly** cancel or modify existing orders.
2. Reliability: Ensure continuous operation and accurate order processing.
3. Scalability: Handle increasing trade volumes and market data efficiently.
4. Maintainability: Easy to monitor, manage, and update.

Kyrylo Cherneha

Functional Requirements

1. Order Management
    - Order Placement: Rapidly place buy and sell orders based on market conditions.
    - Order Cancellation: Quickly cancel or modify existing orders.
2. Market Data Handling
    - Real-time Data Processing: Consume and process real-time market data feeds with minimal delay.
    - Data Storage: Store historical market data for analysis and backtesting.
3. Risk Management
    - Position Limits: Enforce limits on positions to manage risk.
    - Exposure Monitoring: Continuously monitor market exposure and adjust positions.
4. Strategy Execution
    - Algorithmic Trading: Implement and execute trading algorithms based on predefined strategies.
    - Parameter Adjustments: Dynamically adjust strategy parameters based on market conditions.

Non-Functional Requirements

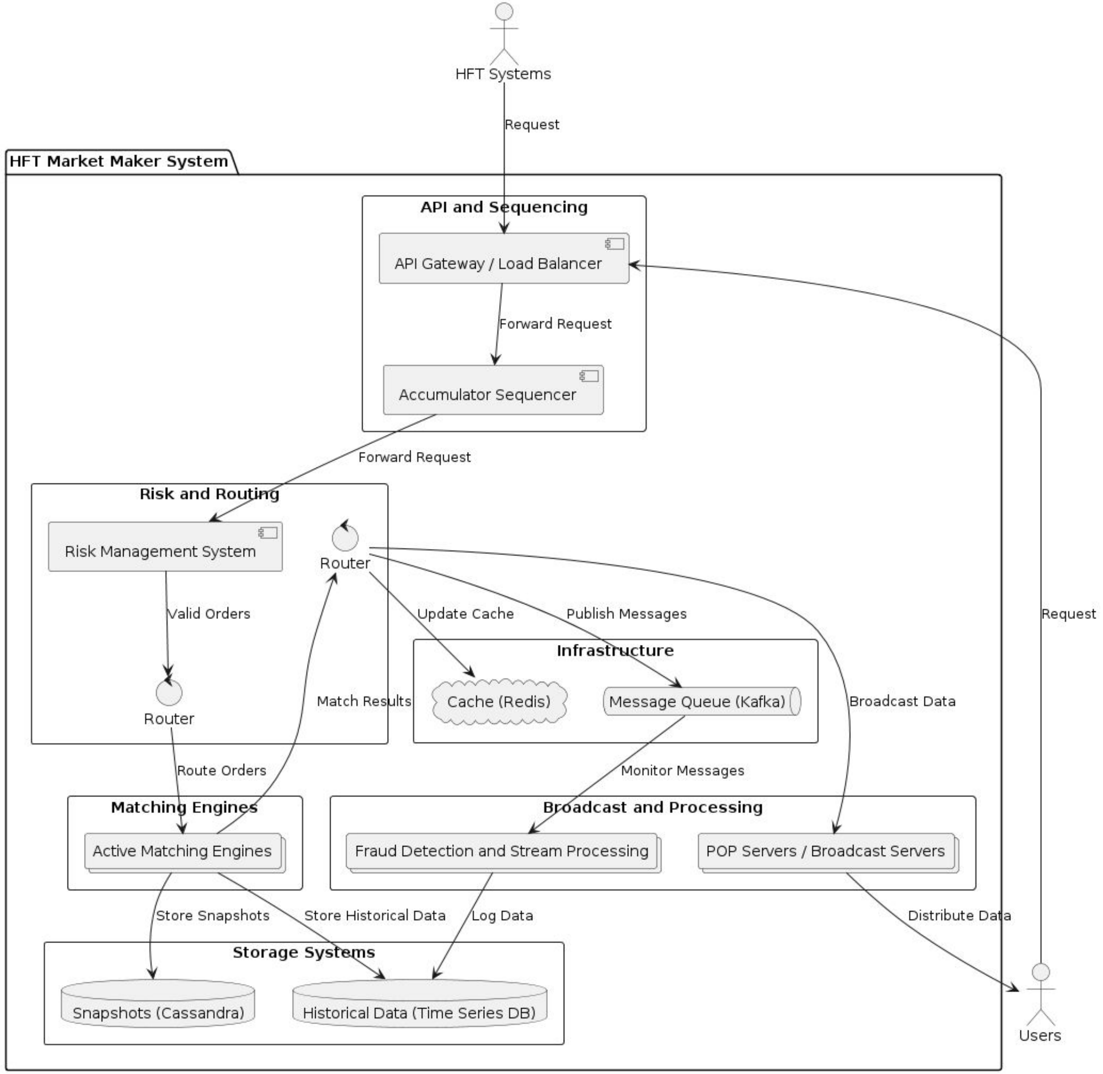
1. Performance
   ○ Sub-millisecond latency for order execution.
   ○ High throughput to handle large volumes of orders and market data.
2. Scalability
   ○ Horizontal scaling to handle more trades and data.
   ○ Vertical scaling for enhanced performance on powerful hardware.
3. Reliability
   ○ High availability with minimal downtime.
   ○ Data integrity and consistency in order processing.
4. Maintainability
   ○ Clear and concise logging for troubleshooting.
   ○ Automated deployment and configuration management.

Back-of-the-Envelope Calculations

1. Latency Estimation
    - Order Execution Latency: Assume an **execution latency of 200 microseconds** for placing or canceling an order.
2. Throughput Estimation
    - Assume a single instance can handle **10,000 orders per second.**
3. Market Data Processing
    - Data Feed Rate: Assume receiving 1,000 market updates per second per feed.
    - Processing Latency: Each update processed within 50 microseconds.
4. Memory Usage
    - Order Book Storage: **Each order entry is 100 bytes**. For 1 million active orders, memory usage is:
        - 1,000,000 orders * 100 bytes = 100 MB
5. Scalability
    - Horizontal Scaling: Adding more instances to handle additional orders and market data. If each instance can handle 10,000 orders/sec, a cluster of 10 instances can handle 100,000 orders/sec.
6. Risk Management Overhead
    - Position Monitoring: Assume monitoring 1,000 positions per second with 10 microseconds per position.
    - Total Monitoring Time: 1,000 positions * 10 microseconds = 10 milliseconds
7. Network Overhead
    - Bandwidth per Order: Assume each order involves 1 KB of data transfer.
    - Bandwidth per Node: For 10,000 orders/sec, the bandwidth requirement is:
        - 10,000 orders/sec * 1 KB = 10 MB/sec

Final Node Specifications

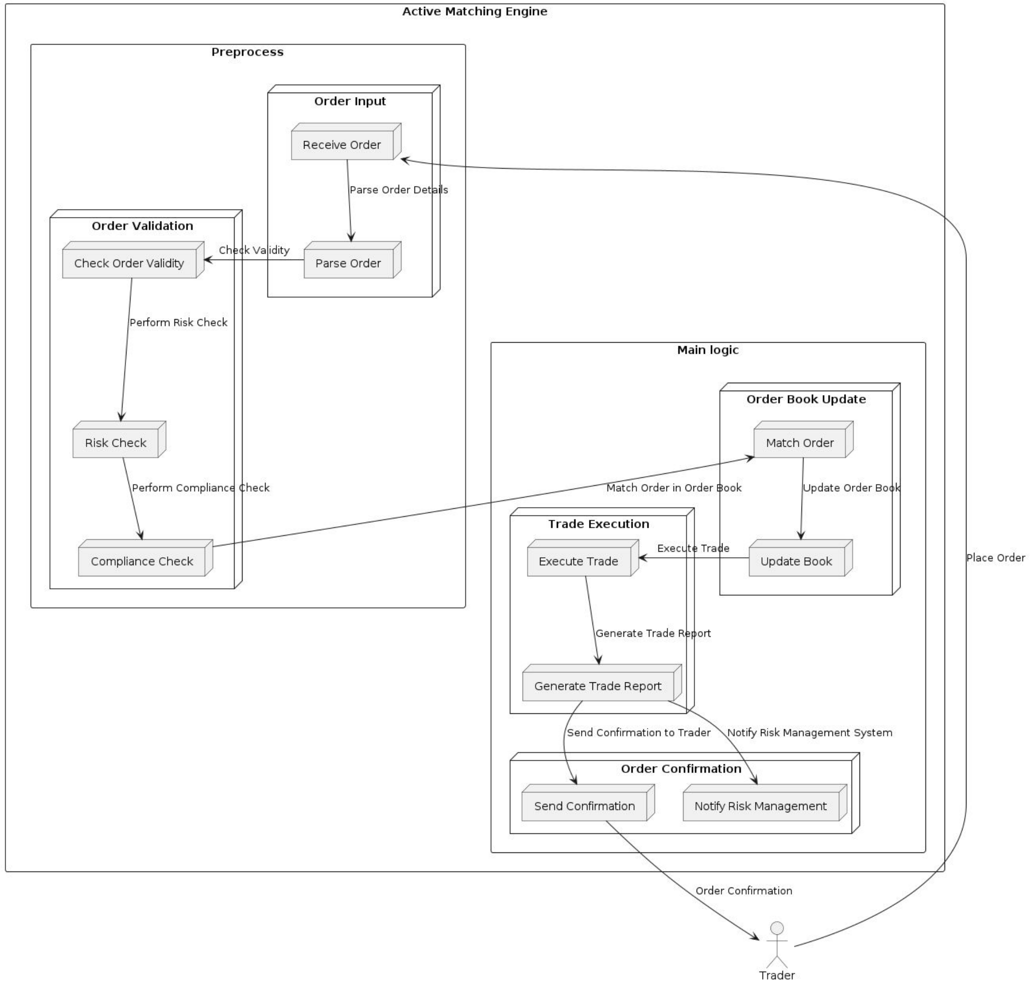
- CPU: 3 cores per node, 3+ GHz frequency.
- Number of Nodes: 10
- RAM per Node: 500 MB
- Network Bandwidth per Node: 10 MB/sec


HFT Systems
Request
HFT Market Maker System
API and Sequencing
API Gateway / Load Balancer
Forward Request
Accumulator Sequencer
Forward Request
Risk and Routing
Risk Management System
Router
Valid Orders
Router
Update Cache
Publish Messages
Request
Infrastructure
Match Results
Cache (Redis)
Message Queue (Kafka)
Broadcast Data
Route Orders
Monitor Messages
Matching Engines
Active Matching Engines
Broadcast and Processing
Fraud Detection and Stream Processing
POP Servers / Broadcast Servers
Store Snapshots
Store Historical Data
Log Data
Distribute Data
Storage Systems
Snapshots (Cassandra)
Historical Data (Time Series DB)
Users

Data Flow and Interactions

- User and HFT Requests: Users and HFT systems send requests through the API Gateway Load Balancer, which directs them to the Accumulator Sequencer.
- Sequencing and Risk Management: The sequenced data is processed by the Risk Management System to ensure compliance with risk parameters.
- Routing and Matching: Orders are routed to the appropriate matching engines (active), which execute trades based on market conditions.
- Data Storage and Processing: Market data, trade executions, and TICK calculations are handled by various data processing components and stored in databases like Cassandra and the Time Series DB.
- Real-Time and Historical Analysis: Stream processing components analyze data in real-time for fraud detection, while historical data analysis is supported by the RDBMS and Time Series DB.
- Broadcast and Delivery: POP and broadcast servers distribute data to users and HFT systems, with CDN ensuring efficient global delivery.

Detailed Active Matching Engine



Metrics for the Risk Management Subsystem in a High-Frequency Trading (HFT)

- Total Position Value:

- Description: The total value of positions held in a trader's portfolio.
- Metric Type: Business
- Usage: To ensure compliance with position limits and manage risk.
- Analysis: Can be compared week-over-week (WoW) or month-over-month (MoM), with average values for the last 10/30/60 minutes and corresponding alerts for significant changes.

- Maximum Allowable Position:

- Description: The maximum position size a trader can take for a specific asset or asset class.
- Metric Type: Business
- Usage: Ensuring adherence to set position limits.
- Analysis: Verification of compliance with established limits and alerting on breaches.

Metrics for the Risk Management Subsystem in a High-Frequency Trading (HFT)

- Market Exposure:

- Description: Current exposure to market risks, measured as a percentage of the total portfolio value or in absolute terms. This metric indicates how susceptible an investor's or trader's portfolio is to market changes.
- Metric Type: Risk Management
- Usage: To assess the level of risk associated with market movements.
- Analysis: Can be tracked in real-time using the 99th, 95th, and 50th percentiles to evaluate risk.

- VaR (Value at Risk):

- Description: The maximum potential loss at a given confidence level over a specific period.
- Metric Type: Risk Management
- Usage: For predicting potential losses under adverse market conditions.
- Analysis: Can be used for daily, weekly, or monthly analysis, comparing current values with historical data.

Metrics for the Risk Management Subsystem in a High-Frequency Trading (HFT)

P&L (Profit and Loss):

- Description: Current profit or loss across all positions.
- Metric Type: Business
- Usage: Evaluating the effectiveness of strategies and making decisions about closing positions.
- Analysis: Comparison with planned targets and previous results to assess performance.

- Delta:

- Description: The change in portfolio value with a one-unit change in the value of the underlying asset. This metric measures the sensitivity of the value of a financial instrument or portfolio to changes in the price of the underlying asset.
- Metric Type: Risk Management
- Usage: To assess the sensitivity of the portfolio to market price changes.
- Analysis: Can be analyzed hourly or in real-time to evaluate the impact of market fluctuations.

Technical Requirements for an Order Management System based on Cassandra

# Replication

Replication is necessary.

Cassandra uses asynchronous replication based on a masterless model, where each node is equal.

Replication ensures high availability and fault tolerance of the system. Thanks to replication, data can be recovered in case one or more nodes fail, which is critical for an order management system where data loss can have serious consequences.

# Partitions

Partitions are needed.

Partitions in Cassandra are implemented based on a **partition key**, which determines on which nodes a specific set of data will be stored.

Partitioning allows for data to be evenly distributed across all nodes in the cluster, ensuring efficient access and load balancing. This is important for ensuring the high performance of the order management system, where quick data access is critical.

Technical Requirements for an Order Management System based on Cassandra

# Transactions

A full ACID model is **not** required.

Cassandra supports lightweight transactions with serializable isolation for critical operations.

For most operations, "**Read Committed"** isolation will suffice. For important operations, lightweight transactions can be used.

Cassandra is not fully ACID-compliant because its architecture focuses on ensuring high availability and scalability, achieved through certain trade-offs in consistency. Lightweight transactions provide the necessary level of consistency for critical operations, such as creating or modifying orders.

# Data Integrity

Ensuring data integrity at the application level using quorum reads and writes.

This will balance consistency and availability. Using quorum reads and writes ensures that the majority of nodes agree on the data, significantly reducing the likelihood of reading outdated data.

Technical Requirements for an Order Management System based on Cassandra

# Consensus

Consensus mechanism is needed.

Cassandra uses a consensus mechanism based on quorum reads and writes, where consensus is achieved by the majority of nodes.

This ensures data consistency in a distributed system and minimizes the risk of reading outdated or incorrect data.

# PACELC

Storage with a priority on **EL** in the PACELC model (Consistency/Availability under Partition, Else Latency/Consistency).

For an order management system, high availability and low latency are important, but it is also necessary to ensure a sufficient level of consistency. Cassandra provides a good balance between these requirements due to its architecture and quorum read/write mechanisms.