STOCK PRICE PREDICTION SYSTEM ARCHITECTURE

Task 04

Intellihack_CypherZ_Task04

Dhanushi Dewmindi
Amanda Hansamali

Contents

Overview

The Stock Price Prediction Platform is a robust and scalable system designed to predict stock prices five trading days into the future using both real-time and historical market data. The system architecture integrates advanced data processing, machine learning, and real-time analytics to support informed decision-making for analysts and traders.

---

1. Data Collection & Ingestion

What It Is:

- The platform collects stock market data from external sources such as **Market APIs** (e.g., **Alpha Vantage**, **Yahoo Finance**).
- These APIs provide historical and real-time market data, including stock prices, volumes, and other financial metrics.

How It Works:

- The system periodically pulls batch data for historical analysis.
- Simultaneously, it listens to streaming data from real-time APIs to update predictions dynamically.

Data Ingestion Methodology:

- Data Sources: Market data APIs (e.g., Alpha Vantage, Yahoo Finance)
- Streaming Method: Apache Kafka for real-time data streaming
- Batch Method: Scheduled data pulls and storage in a data lake (e.g., AWS S3)

Component Justification:

- Kafka: Ideal for high-throughput, low-latency streaming of real-time market data. Ensures durability and scalability of data ingestion.

---

2. Data Processing Pipeline

What It Is:

- Data ingestion involves collecting data from the market APIs and preparing it for processing.
- Uses **Apache Kafka** for **real-time streaming** of data.
- **ETL tools** (e.g., **Apache NiFi**, **Airflow**) handle **batch data ingestion**.

Data Flow:

- **Real-Time Data:** Market APIs → Kafka → Data Processing
- **Batch Data:** Market APIs → ETL Tool → Data Storage → Data Processing

Data Ingestion Methodology:

- Preprocessing: Apache Spark for data cleaning and feature engineering
- Real-Time Processing: Spark Streaming for generating features on the fly
- Storage: InfluxDB for optimized time-series data storage

Component Justification:

- Spark: Provides distributed processing power to handle large datasets quickly. Ideal for ETL processes and real-time feature engineering.
- InfluxDB: Specialized for time-series data, allowing efficient querying and storage of stock market data.

---

3. Model Operations

What It Is:

- This phase involves model training, evaluation, and deployment.
- The platform uses **Flask** or **FastAPI** to create a **RESTful API** that serves the predictive model.

Data Ingestion Methodology:

- Model Training: Weekly retraining on AWS EC2 using frameworks like TensorFlow or PyTorch
- Model Evaluation: MLflow for tracking performance metrics

- Deployment: Flask/FastAPI REST API for serving the model
- Monitoring: Prometheus + Grafana for system health and performance

Component Justification:

- **AWS EC2**: Offers scalable computing resources for model training.
- **Flask/FastAPI:** Lightweight and fast web frameworks ideal for serving prediction models as REST APIs.
- **Prometheus + Grafana**: Allows real-time monitoring and visualization of system performance.

---

4. Insight Delivery

What It Is:

- Insights from the model are delivered to users via a **Streamlit** or **Dash dashboard**.
- The dashboard displays current predictions, historical trends, and analytics.

How It Works:

- The **API** feeds data into the dashboard, updating predictions and analytics in real time.
- Users can view **graphs**, **charts**, and **predicted stock prices**.
- The dashboard can also display alerts based on certain thresholds (e.g., if a stock price is predicted to rise significantly).

Data Ingestion Methodology:

- Presentation Layer: Dash/Streamlit dashboard for real-time insights
- Reporting: Automated generation of analytics and performance reports
- User Interaction: Financial analysts and brokers accessing insights through the dashboard

Component Justification:

- **Streamlit:** Provides an easy-to-build, interactive dashboard for displaying model predictions and analytics.

System Considerations

- Scalability: Kubernetes for scaling components
- Reliability: Redundant data storage with S3 backups
- Latency Optimization: Redis caching for faster model inference
- Cost Management: Serverless functions (AWS Lambda) for non-critical tasks

Component Justification:

- Kubernetes: Manages containerized applications, offering auto-scaling and high availability.
- Redis: In-memory data store that reduces latency during real-time model predictions.
- AWS Lambda: Reduces operational costs by running code only when needed.

Data Flow

Streaming Workflow:

1. Market APIs provide real-time stock price data.
2. Apache Kafka ingests streaming data with minimal latency.
3. Spark Streaming processes data in real-time, performing preprocessing and feature engineering.
4. Processed data is stored in InfluxDB.
5. The model API (Flask/FastAPI) serves predictions based on incoming data.
6. Predictions and analytics are displayed on the Streamlit dashboard for end-users.

Batch Workflow:

1. Scheduled batch jobs retrieve historical stock data and additional features.
2. Data is processed through Spark, including preprocessing and feature generation.
3. The processed data is used for model training and evaluation on AWS EC2.
4. Updated models are deployed via the Flask/FastAPI model API.
5. Historical predictions and analysis results are stored in a data lake (e.g., AWS S3).

Decision Making:

- Model predictions generate buy/sell signals that are tested using a simulated trading strategy.
- The Streamlit dashboard provides visualizations and performance metrics for analysts.

Challenges & Mitigations

1. Data Latency:

- Challenge: Delays in streaming data can impact real-time predictions.
- Mitigation Strategy: Utilize Redis caching to store frequently accessed data, reducing load times and enhancing responsiveness.

2. Model Drift:

- Challenge: Changes in market conditions over time may reduce model accuracy.
- Mitigation Strategy: Implement continuous model retraining and use MLflow to monitor model performance metrics.

3. Scalability:

- Challenge: Handling increasing data volume and processing demands.
- Mitigation Strategy: Use Kubernetes to dynamically scale processing resources and manage containerized applications effectively.

4. Cost Management:

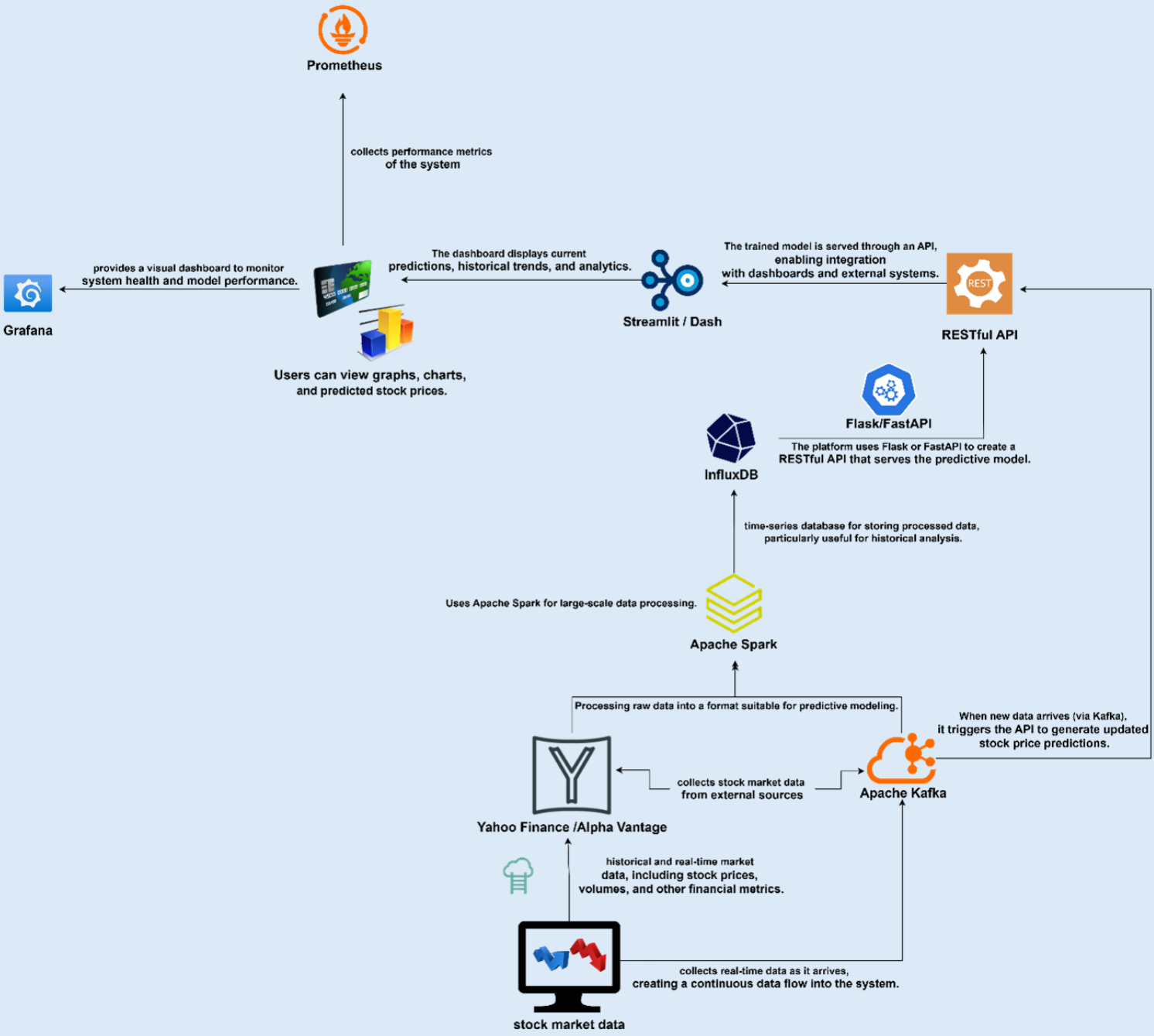
- Challenge: High operational costs for continuous data processing and model deployment.
- Mitigation Strategy: Deploy serverless functions (AWS Lambda) for on-demand processing, reducing idle infrastructure costs.

5. System Reliability:

- Challenge: Potential system failures that disrupt data flow or model predictions.
- Mitigation Strategy: Implement redundant data storage with S3 backups and use Prometheus + Grafana for proactive system monitoring and alerting.

Architecture Diagram

- The diagram will visually represent:
  - Data Flow: From ingestion to dashboard
  - Key Components: Data source, processing, model API, and frontend
  - Interactions: Between components like Kafka, Spark, Flask, and Streamlit



Conclusion

The **Stock Price Prediction Platform** presents a comprehensive solution for financial forecasting, blending real-time analytics with historical analysis to maximize prediction accuracy. The architecture is designed with **scalability**, **reliability**, and **performance** in mind, leveraging modern technologies like **Apache Kafka**, **Spark**, **Flask/FastAPI**, and **Grafana**.

By integrating advanced data processing and machine learning capabilities, the platform not only offers predictive insights but also provides a robust framework for continuous improvement through model retraining and performance monitoring. The clear data flow and modular design ensure that the system is adaptable to changing market conditions and can support future enhancements seamlessly.

This architecture supports a **data-driven decision-making process**, enabling analysts and traders to gain a competitive edge in the financial market through **real-time predictions** and **actionable insights**.

Appendix

Glossary of Key Terms:

- **Market API:** Application Programming Interface for accessing stock market data (e.g., Alpha Vantage, Yahoo Finance).
- **Apache Kafka:** A distributed streaming platform used for building real-time data pipelines and streaming applications.
- **Apache Spark:** An open-source unified analytics engine for large-scale data processing.
- **Flask/FastAPI:** Web frameworks used to build and deploy machine learning models as APIs.
- **InfluxDB:** A time-series database optimized for storing high-frequency data like stock prices.
- **Prometheus & Grafana**: Monitoring and visualization tools for system performance management.
- **ETL** (Extract, Transform, Load): The process of extracting data, transforming it into a suitable format, and loading it into a database.

Technology Stack:

- Data Collection: Alpha Vantage, Yahoo Finance
- Data Streaming: Apache Kafka
- Data Processing: Apache Spark
- Storage: InfluxDB, AWS S3 (for batch data)
- Model Operations: Flask/FastAPI for API development
- Visualization: Streamlit/Dash for dashboards
- Monitoring: Prometheus and Grafana
- Orchestration: Kubernetes (for scalability)

System Design Tools:

- Diagram Creation: Lucidchart, Draw.io
- Development Environment: Jupyter Notebook, PyCharm
- Programming Languages: Python (Pandas, Scikit-learn, TensorFlow)

Architecture Diagram