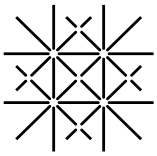

University of Basel

Proceedings of the Distributed Information Systems (DIS) Course

Workshop Reports

Spring Semester 2025

Natural Science Faculty of the University of Basel
Department of Mathematics and Computer Science

Preface

This documents contains all student workshop reports of the Distributed Information Systems (DIS) course held at the University of Basel in the spring semester 2025.

Organization

The Distributed Information Systems (DIS) course of the spring semester 2025 is organized by the Databases and Information Systems (DBIS) research group, Department of Mathematics and Computer Science, Univeristy of Basel.

Lecturers

Prof. Dr. Heiko Schuldt (heiko.schuldt@unibas.ch)
Dr. Marco Vogt (marco.vogt@unibas.ch)

Assistants

Florian Spiess, M. Sc. (florian.spiess@unibas.ch)
Yiming Wu, M. Sc. (yiming.wu@unibas.ch)

Table of Contents

Reports

TSLA Stock Anomaly Detection

Scott Barcomb and Soham Roy

University of Basel
Distributed Information Systems (DIS) course
Spring Semester 2025

Introduction

In recent years, the volume of data generated by financial markets has grown exponentially, increasing the importance of efficient and scalable data analysis solutions. This project leverages Apache Spark, a widely used open-source framework designed for big data processing, to perform real-time data analysis on simulated streaming data from Tesla's stock market history.

The objective of this project was to design, implement, and deploy a robust data streaming analysis pipeline capable of handling continuous input data. To simulate the streaming environment, a historical dataset containing Tesla's stock prices was used, with data being progressively introduced to mimic live market feeds. Initial development and testing was carried out locally using Python to ensure the correctness and functionality of the Spark application prior to deployment.

As such, to demonstrate scalability and real-world applicability, the Spark application was deployed on a distributed computing environment consisting of a three-node AWS EC2 cluster. This setup utilized Hadoop Distributed File System (HDFS) for data storage and Apache YARN as the resource management system, allowing the Spark application to efficiently distribute tasks and manage computational resources.

This report covers the methodology, implementation steps, and insights gained from processing the simulated real-time streaming data. With it, we hope to highlight the strengths of Apache Spark in handling high-throughput analysis tasks and showcase the potential scalability of our project.

Related Work

Real-time anomaly detection in stock market data is a critical area within distributed information systems that offers insight into market irregularities that can inform trading strategies and risk management decisions. This project specifically addresses anomaly detection in financial data streaming, aligning to the

scope of the workshop specification.

The dataset used for this project was sourced from Yahoo Finance, comprising historical stock data from Tesla Inc. (TSLA). The chosen dataset included the essential columns required for our financial analysis: open price, high price, low price, close price, adjusted close price, and trading volume. These metrics provide the comprehensive market behavior insights needed for anomaly detection tasks, as they reflect market trends, volatility, and liquidity.

To facilitate effective anomaly detection, several statistical techniques were implemented:

- **Exponentially Weighted Moving Average (EWMA):** Utilized to smooth out short-term fluctuations and highlight longer-term trends or deviations within the data.
- **Z-Score:** Calculated to measure the degree to which individual data points deviate from the dataset's mean, identifying potential anomalies.
- **Bollinger Bands:** Employed to quantify volatility, with anomalies identified by data points significantly deviating outside these dynamically adjusting bands.

These techniques were chosen due to their widespread use in financial analytics for detecting unusual market movements, providing a foundation for our real-time anomaly detection pipeline. This selection of methods aligns with some of the common practices and recent advances explored within the distributed systems community.

Positioning this project within the broader academic and practical context, it leverages Apache Spark's scalable streaming capabilities, reflecting the technology's relevance to current industry standards. Thus, this project demonstrates a practical implementation for how distributed systems can enhance financial data analysis, specifically targeting real-time anomaly detection scenarios.

Implementation

The implementation of this real-time anomaly detection pipeline was influenced by considerations related to distributed systems, cloud deployment, available development tools, and our programming ability. AWS was selected as the deployment platform due to its huge prominence in distributed computing and direct relevance to the course curriculum and workshop scope. Additionally, our existing AWS Academy accounts provided a convenient and cost-effective environment for the project's deployment.

Python was chosen as the primary programming language due to its integration with Apache Spark through PySpark, in addition to the large ecosystem of

libraries available, which greatly streamlined the process of data analysis and visualization. To utilize existing resources effectively, the project adapted the previously configured three-node EC2 instance cluster from Exercise 3 of the course that already featured Apache Hadoop (HDFS) for file storage and Apache YARN for resource management. Apache Spark was subsequently installed and configured to integrate seamlessly with this environment, significantly simplifying the deployment process.

Conceptually, the project architecture and code structure were organized into three distinct but interconnected components:

1. **Dataset Acquisition and Preprocessing:**
   - A Python script was developed to automate downloading Tesla's historical stock data from Yahoo Finance.
   - A separate preprocessing script split this data into smaller CSV files, emulating a streaming data feed by sequentially placing these files into an input directory (`stream_input/`) accessible to Apache Spark.
2. **Streaming Data Analysis with Apache Spark:**
   - Apache Spark continuously monitored the input directory, ingesting new data files in real-time.
   - Spark performed streaming analysis, computing essential financial indicators, including EWMA, z-scores, and Bollinger Bands.
   - Processed results, along with identified anomalies, were outputted to an output directory, providing a continuous feed of analyzed data.
3. **Real-time Visualization and Dashboard:**
   - A Python-based visualization app read continuously from the Spark output directory to update a real-time dashboard accessible via localhost in a browser.
   - This dashboard displayed a line graph of recent closing prices alongside computed EWMA, upper, and lower Bollinger Bands, marking any detected anomalies directly on the graph.
   - Below the visualization, a recommendation section provided suggestions to buy, hold, sell, short, or avoid the stock, derived from comparing the latest price to the EWMA.
   - The dashboard presented explicit warnings when the computed z-score exceeded a threshold of 2, indicating significant anomalies.

This structure allowed us to create a clear separation of concerns, promoting maintainability, scalability, and ease of debugging, aligning well with best practices in distributed system design. A detailed discussion of the exact setup, including code specifics and AWS cluster configuration, will be provided in the subsequent Setup section.

Setup

This section provides a detailed overview of the setup procedures, describing how to build, configure, and run the project both locally and on an AWS EC2 cluster. We split it into two parts for clarity: local development and cloud deployment.

Local Setup

The local environment allows easy development and testing on a single machine. The project structure is:

```
DIS_Workshop/
    data/
        raw/                        # Downloaded full TSLA CSV dataset
        stream_input/               # Minute-sliced simulated stream
        output/
            combined_anomalies/     # Final output
            checkpoint_combined/    # Spark state checkpoints
    scripts/
        download_stock_data.py      # Download OHLC (open-high-low-close) data
        split_to_stream.py          # Split OHLC CSV into minute-based rows
        simulate_live_feed.py       # Simulated streaming writer
        streaming_combined.py       # Unified detection pipeline
    dashboard/
        app.py                      # Interactive Dash dashboard
    notebooks/
        analysis.ipynb              # Jupyter-based visual analysis
    requirements.txt
    README.md
```

Setup Instructions

1. Unpack the project repository.
2. Create and activate a Python virtual environment:

```
python3 -m venv env
source env/bin/activate
```

3. Install the required Python libraries:

```
pip install -r requirements.txt
```

Project Execution Execution involves three separate terminals due to three concurrent processes:

- **Terminal 1: Data preprocessing and streaming simulation**

```
python scripts/download_stock_data.py
python scripts/split_to_stream.py
python scripts/simulate_live_feed.py
```

- **Terminal 2: Apache Spark streaming analysis**

```
python scripts/streaming_combined.py
```

- **Terminal 3: Visualization dashboard**

```
python dashboard/app.py
```

Access the dashboard at `http://localhost:8050` in your browser.

AWS Deployment

The AWS setup mirrors local development with these changes:

- The Jupyter notebook (`notebooks/analysis.ipynb`) is omitted.
- Data directories live in HDFS:
  - Input: `/Workshop/data/stream_input/`
  - Outputs:
    - `/Workshop/data/output/combined_anomalies/`
    - `/Workshop/data/output/checkpoint_combined/`

Cluster Setup and Execution

1. Ensure your EC2 cluster has HDFS and YARN configured and running.
2. SSH into the NameNode/master node.
3. Prepare your data by downloading and splitting as described in local setup instructions, creating the necessary HDFS directories:

```
hdfs dfs -mkdir -p /Workshop/data/stream_input/
hdfs dfs -mkdir -p /Workshop/data/output/combined_anomalies/
hdfs dfs -mkdir -p /Workshop/data/output/checkpoint_combined/
```

4. Launch the Spark streaming job under YARN:

```
spark-submit --master yarn scripts/streaming_combined.py
```

5. On the NameNode, start the dashboard:

```
python dashboard/app.py
```

6. From your local machine, set up SSH tunneling:

```
ssh -i yourkey.pem -L 8050:localhost:8050 username@namenode-public-dns
```

Now open `http://localhost:8050` in your browser to view the live dashboard.

Results

In this section, the outcomes generated by our Apache Spark-based streaming system will be presented. The primary results include EWMA drift detection, identification of Z-score anomalies, recognition of Bollinger Band breaches, and the provision of actionable trading suggestions. Our system, deployed on AWS, successfully processed and analyzed streaming TSLA data in real-time by utilizing Apache Spark Structured Streaming. The implementation included three distinct anomaly detection techniques—EWMA, Z-score, and Bollinger Bands—and visualized them through an interactive dashboard.

Streaming EWMA Trends

window	EWMA
{2025-05-13 17:19:00, 2025-05-13 17:24:00}	321.5294121490478
{2025-05-14 18:07:00, 2025-05-14 18:12:00}	347.0706969940185
{2025-05-14 19:34:00, 2025-05-14 19:39:00}	344.01974906921384
{2025-05-15 18:08:00, 2025-05-15 18:13:00}	341.3331680603027
{2025-05-15 21:50:00, 2025-05-15 21:55:00}	343.43339649658196
{2025-05-16 17:10:00, 2025-05-16 17:15:00}	346.4749437896728
{2025-05-15 18:04:00, 2025-05-15 18:09:00}	340.58765051879874
{2025-05-15 18:45:00, 2025-05-15 18:50:00}	344.495147668457
{2025-05-15 20:00:00, 2025-05-15 20:05:00}	344.0934059570312
{2025-05-19 20:00:00, 2025-05-19 20:05:00}	340.3965481628417
{2025-05-15 19:34:00, 2025-05-15 19:39:00}	344.71595355224605
{2025-05-13 20:07:00, 2025-05-13 20:12:00}	333.7278114898681
{2025-05-13 18:07:00, 2025-05-13 18:12:00}	321.0683486114501
{2025-05-13 20:13:00, 2025-05-13 20:18:00}	334.8764242462157
{2025-05-14 15:45:00, 2025-05-14 15:50:00}	338.4937077880859
{2025-05-14 16:10:00, 2025-05-14 16:15:00}	342.4016720123291
{2025-05-14 19:15:00, 2025-05-14 19:20:00}	346.10685648803707
{2025-05-16 17:27:00, 2025-05-16 17:32:00}	347.4667334655761
{2025-05-16 16:57:00, 2025-05-16 17:02:00}	345.8138170837402
{2025-05-16 20:48:00, 2025-05-16 20:53:00}	347.8858678344726

Fig. 1: The EWMA values for TSLA, computed over sliding windows. A consistent upward drift is observed through mid-May.

Z-score Based Anomaly Detection

window	latest_close	z_score	is_anomaly
{2025-05-13 17:19:00, 2025-05-13 17:24:00}	321.5799865722656	0.7742276354006632	false
{2025-05-14 18:07:00, 2025-05-14 18:12:00}	347.5	1.544711045906067	false
{2025-05-14 19:34:00, 2025-05-14 19:39:00}	343.5350036621094	-1.3052613214380142	false
{2025-05-15 18:08:00, 2025-05-15 18:13:00}	341.3900146484375	0.1567515625531571	false
{2025-05-15 21:50:00, 2025-05-15 21:55:00}	343.25	-0.5100789675177955	false
{2025-05-16 17:10:00, 2025-05-16 17:15:00}	347.0050048828125	1.0902244750270202	false
{2025-05-15 18:04:00, 2025-05-15 18:09:00}	340.4198913574219	-1.1810244365332812	false
{2025-05-15 18:45:00, 2025-05-15 18:50:00}	345.0400085449219	0.7045408023627372	false
{2025-05-15 20:00:00, 2025-05-15 20:05:00}	344.11590576171875	0.04846053445589935	false
{2025-05-19 20:00:00, 2025-05-19 20:05:00}	340.4800109863281	0.7645894918277928	false
{2025-05-15 19:34:00, 2025-05-15 19:39:00}	344.9750061035156	0.7556148256400858	false
{2025-05-13 20:07:00, 2025-05-13 20:12:00}	333.6050109863281	-0.5028577704864665	false
{2025-05-13 18:07:00, 2025-05-13 18:12:00}	321.0150146484375	-0.09912275095039934	false
{2025-05-13 20:13:00, 2025-05-13 20:18:00}	334.7513122558594	-0.6849442270595038	false
{2025-05-14 15:45:00, 2025-05-14 15:50:00}	338.4649963378906	-0.05379212260145959	false
{2025-05-14 16:10:00, 2025-05-14 16:15:00}	342.1400146484375	-0.2805843366852474	false
{2025-05-14 19:15:00, 2025-05-14 19:20:00}	346.07000732421875	-0.3373924472512321	false
{2025-05-16 17:27:00, 2025-05-16 17:32:00}	347.6600036621094	0.7606017249235754	false
{2025-05-16 16:57:00, 2025-05-16 17:02:00}	345.4700927734375	-1.5626259834200502	false
{2025-05-16 20:48:00, 2025-05-16 20:53:00}	347.83990478515625	-0.5683215996916106	false

Fig. 2: Z-score calculations for stock prices over five minute windows. No Z-score breaches beyond ±2 detected, indicating price stability.

Bollinger Band Evaluation

window	latest_close	upper_band	lower_band	is_outside_band
{2025-05-13 17:19:00, 2025-05-13 17:24:00}	321.5799865722656	321.6702215856512	321.3757623010676	false
{2025-05-14 18:07:00, 2025-05-14 18:12:00}	347.5	347.6490324985347	346.33968820459023	false
{2025-05-14 19:34:00, 2025-05-14 19:39:00}	343.5350036621094	344.8266974832488	343.2635002706575	false
{2025-05-15 18:08:00, 2025-05-15 18:13:00}	341.3900146484375	342.031005964331	340.6400023364503	false
{2025-05-15 21:50:00, 2025-05-15 21:55:00}	343.25	344.0584565492761	342.77012011088016	false
{2025-05-16 17:10:00, 2025-05-16 17:15:00}	347.0050048828125	347.44394968581344	345.51404591965536	false
{2025-05-15 18:04:00, 2025-05-15 18:09:00}	340.4198913574219	340.96936467131405	340.27842585602974	false
{2025-05-15 18:45:00, 2025-05-15 18:50:00}	345.0400085449219	345.9708423630537	343.09669914085254	false
{2025-05-15 20:00:00, 2025-05-15 20:05:00}	344.11590576171875	344.68147504994545	343.52224809458585	false
{2025-05-19 20:00:00, 2025-05-19 20:05:00}	340.4800109863281	340.63736834402357	340.1278782380077	false
{2025-05-15 19:34:00, 2025-05-15 19:39:00}	344.9750061035156	345.3817835741287	344.07422228524626	false
{2025-05-13 20:07:00, 2025-05-13 20:12:00}	333.6050109863281	334.41117403949244	333.1227859214451	false
{2025-05-13 18:07:00, 2025-05-13 18:12:00}	321.0150146484375	321.5396573083753	320.5399203283435	false
{2025-05-13 20:13:00, 2025-05-13 20:18:00}	334.7513122558594	335.57848684803776	334.3461713550872	false
{2025-05-14 15:45:00, 2025-05-14 15:50:00}	338.4649963378906	339.5341528169541	337.45184571820215	false
{2025-05-14 16:10:00, 2025-05-14 16:15:00}	342.1400146484375	344.37511038055504	340.4548945022575	false
{2025-05-14 19:15:00, 2025-05-14 19:20:00}	346.07000732421875	347.06325911666556	345.36349869583444	false
{2025-05-16 17:27:00, 2025-05-16 17:32:00}	347.6600036621094	347.9608499547555	346.989906881182	false
{2025-05-16 16:57:00, 2025-05-16 17:02:00}	345.4700927734375	346.33436346752376	345.36398858325754	false
{2025-05-16 20:48:00, 2025-05-16 20:53:00}	347.83990478515625	348.1308185352437	347.6777385936626	false

Fig. 3: TSLA price vs. its Bollinger Bands. All values remained within bounds, suggesting low short-term volatility.

Dashboard Visualization

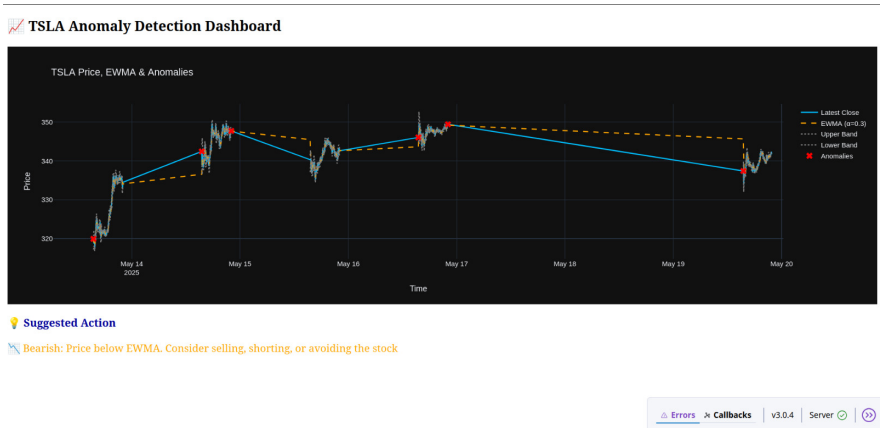

Fig. 4: The interactive dashboard displaying the price stream, EWMA, Bollinger Bands, and detected anomalies.

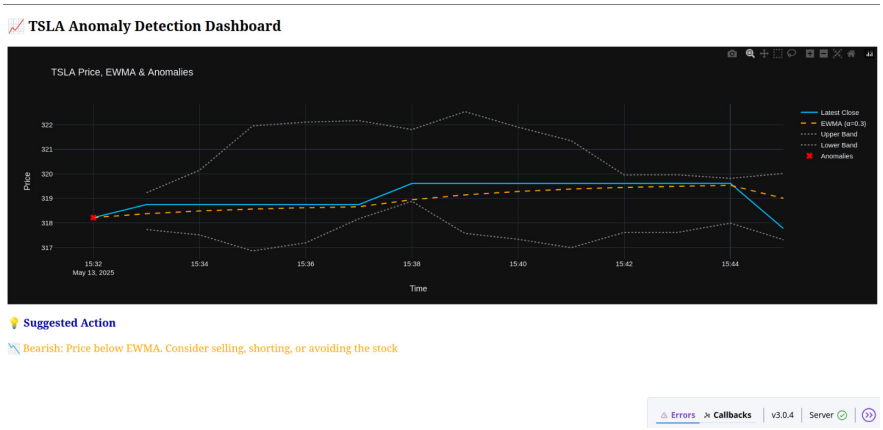

Fig. 5: A zoomed-in dashboard view showing a localized anomaly spike. The suggested action advises bearish strategy.

Interpretation

The EWMA score effectively tracked smoothed price trends using an alpha value of 0.3. On May 14, a noticeable spike in the actual stock price above the EWMA

value indicated upward momentum. The EWMA curve adjusted gradually, providing clear trend confirmation by filtering out the noise of short-term price fluctuation.

Overall, the EWMA provided a reliable baseline for identifying sustained market sentiment. When stock prices remained persistently above or below the EWMA, it indicated bullish or bearish sentiment, respectively. Across the analyzed periods:

- EWMA trends showed an upward drift between May 13–15, followed by stabilization.
- Z-scores generally fluctuated near 0, with no statistical anomalies ($|z| > 2$) observed.
- Stock prices consistently remained within the upper and lower Bollinger Bands.
- Dashboard alerts correlated with early price drops and subsequent returns to mean behavior.

The system performed reliably during periods of stable price movement. In live anomaly scenarios, alerts marked by red "X" symbols on the dashboard would recommend actionable trading strategies (buy, sell, or hold). Additionally, if the z-score exceeded the threshold value of 2, the system would alert users to a significant anomaly. However, throughout our testing period, most z-scores remained within ± 1.5, indicating stable market conditions. Notably, a price jump on May 12 triggered a z-score alert, marking one of the few instances where the threshold was breached.

Our third anomaly detection approach utilized Bollinger Bands to identify rapid price deviations. For the duration of our testing phase, prices consistently stayed within the computed bands. Prices occasionally approached the upper band, however, hinting at elevated volatility during that time. The Bollinger Bands thus served well as volatility thresholds. Although no breaches occurred, instances where prices neared the bands often aligned with z-score alerts, demonstrating the convergence of these detection signals.

Conclusion

This project successfully demonstrated the feasibility and effectiveness of using Apache Spark for real-time anomaly detection in streaming financial data. Using historical Tesla stock data to simulate a real-time streaming environment, the ability of distributed systems to provide insights into stock market behavior was effectively demonstrated.

The implementation utilized advanced statistical techniques such as EWMA, z-score, and Bollinger bands, which proved effective in detecting anomalies and

providing indicators for potential market actions. The deployment on AWS using a Hadoop and YARN-based three-node EC2 cluster validated the project's scalability and relevance to real-world big data processing scenarios.

Throughout this project, several insights were gained regarding the configuration and optimization of distributed streaming applications, especially in terms of deployment strategies, data pipeline design, and visualization techniques. Additionally, the modular design facilitated ease of debugging and maintenance, which was critical once the transition to cloud deployment began, requiring us to make fast, iterative changes to our streaming application and visual dashboard script.

Future work could explore further scaling the architecture to handle much larger datasets and faster streaming rates, improving dashboard interactivity for more advanced analytics, and improving anomaly detection accuracy with machine learning approaches. Overall, this project contributed valuable practical experience to us and provided a strong foundation for further research and application in distributed systems and financial data analytics.

References

1. H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.