Improving Post-Trade Risk Analysis at Jump Trading

A Technical Report submitted to the Department of Computer Science

Presented to the Faculty of the School of Engineering and Applied Science

University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree

Bachelor of Science, School of Engineering

Arjun Kumar

Spring, 2022

Technical Project Team Members

On my honor as a University Student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments

Briana Morrison, Department of Computer Science

Improving Post-Trade Risk Analysis at Jump Trading

CS4991 Capstone Report, 2023

Arjun Kumar
Computer Science
The University of Virginia
School of Engineering and Applied Sciences
Charlottesville, Virginia USA
Ak7cw@virginia.edu

Abstract
Jump Trading, a high frequency trading firm, needed a way to reduce the risk of having unaccounted-for trades. I created a web application with specific visualization to solve this problem. I collaborated with the risk team to gather requirements and used React, Python, and SQL to build the application. This project enabled the risk team to account for millions of individual trades. In the future the UI needs to be improved to match the styling of the other Jump Trading internal applications.

1. Introduction

High frequency trading firms such as Jump Trading can make billions of trades a day. These trades go through multiple systems that sometimes are inconsistent with each other. For example, a trading algorithm may believe that a specific trade went through, whereas the exchange is not able to account for the trade.

It is important to fully understand what is happening with a specific trade in order to accurately log trades in the database and accurately understand current positions within the market. If trades are entered into the database without fully being acknowledged by all systems, then the risk team needs a way to view these unacknowledged trades and roll them back from the database.

2. Related Works

Kumar (2019) suggested using GraphQL subscriptions to achieve real-time dashboard functionality with React. GraphQL subscriptions allow for real-time updates from the backend to the frontend.

I initially followed Kumar's architecture recommendation, but eventually switched to using an API polling technique as Niedringhaus (2020) discussed. With the polling technique, the backend is called in a set interval every x seconds in order to keep the frontend data synced to the backend data.

3. Process Design

The firm already had an existing platform in place for viewing unacknowledged trades, organized by specific groups such as exchange or trading team. However, my job was to enhance the platform's capabilities by making it more detailed, allowing users to view individual trades within specific groups and acknowledge these trades on a granular level. The existing architecture of the platform consisted of a React frontend, a Python API webserver, and a SQL database from which the data was retrieved.

Fortunately, the requirement for the updated website did not involve having to alter this architecture or modify the SQL, as all of the relevant data was already being sent to the Python server. However, I needed to change the way data was being sent from the server

to the client in order to include the new information that would be displayed on the platform.

To implement this new feature, I opted to take a step-by-step approach, focusing on making frontend changes before tackling backend modifications. I chose this strategy because I wanted to fully comprehend how the frontend needed to receive and process the new data in order to successfully render and display it to users. While working on the frontend, I utilized fake data as a placeholder since the actual data had not yet been incorporated through the backend changes.

Upon completing the frontend alterations and gaining a deeper understanding of the data requirements, I proceeded to make the necessary backend adjustments. This ensured that the platform was able to efficiently receive, process, and display the new data as intended, ultimately providing users with a more detailed view of individual trades within specific groups and the ability to acknowledge these trades with ease.

4. Results

The implementation of a new feature within the risk management system has brought about a significant transformation in the way trades are handled. With this new feature, the risk team can now identify and unacknowledged specific trades individually, rather than having to group them together.

This marks a major breakthrough for the risk management team, as it enables them to exercise a more granular approach to risk control. By identifying individual trades, the team is able to analyze each trade on its own merits, taking into account its specific risk profile and associated factors.

The potential benefits of this new feature are immense. By being able to more accurately assess the risk of each trade, the risk team is better positioned to mitigate risk and potentially save millions of dollars.

5. Conclusion

The implementation of the web application with specific visualizations for Jump Trading has been successful in solving the problem of unacknowledged trades. The collaboration with the risk team, utilization of React, Python, and SQL, and the step-by-step approach have all contributed to the success of the project. The new feature has enabled the risk team to identify individual trades, exercise a more granular approach to risk control, and potentially save millions of dollars. Overall, this project demonstrates the importance of technology in the world of finance and the potential benefits that can be gained by utilizing innovative solutions.
Niedringhaus

6. Future Work

Future improvements include the improvement of the UI to match the styling of other Jump Trading internal applications. Memory improvements could also be made on the client in order to reduce memory usage.

References

Kumar, N. (2019, March 17). Building a real-time dashboard using React, GraphQL subscriptions, and Redis PubSub. Medium. https://medium.com/@nowke/building-a-real-time-dashboard-using-react-graphql-subscriptions-and-redis-pubsub-49f5e391a4f9

Niedringhaus, P. (2020, March 11). Polling in React Using the useInterval Custom Hook. Bits and Pieces. https://blog.bitsrc.io/polling-in-react-using-the-useinterval-custom-hook-e2bcefda4197