Case Study

Equity Trading Digital Transformation – Tier 1 Investment Bank

THE PROBLEM

Equity trading systems are complex systems characterized by the need to move, process, and analyze massive amounts of data and trades in real time. These systems operate on extreme volumes and velocities of market data and trades, are extremely sensitive to data analysis and trade execution latencies, mandate ordered and zero loss execution during both steady state operation and while recovering from any kind of system failure and demand a very high development velocity as they continually adapt to new rules and market conditions. These systems are continually pushed to their limits by this trifecta of performance, reliability and development agility.

The extreme nature of these systems renders traditional application architectures as inadequate. This mandates bespoke architectures and compute models that need to be developed on an ad-hoc basis resulting in the entire application stack, including infrastructure and business logic, to be owned by the development teams. Furthermore, given the stringency of the performance, reliability and agility requirements and the tight deadlines within which these development teams operate, these architectures are inexorably driven by how to achieve the right non-functional compromises rather than system resilience and robustness. This is very costly, resource intensive, hard to maintain, exhibit higher than mandated time to market characteristics, result in brittle systems and have no ramp to the cloud.

This reason for the high cost, low time to market and low developmental agility lies not in the development of the application layer, but rather, in the implementation and maintenance of the infrastructure that underpins this layer and frames these bespoke architectures. *Over 60% of development time is spent in the development and maintenance of this infrastructure and is done by the development teams*. Until now, trading alpha was generated primarily from optimized or differentiated infrastructure thus justifying such an investment. However, as financial institutions rapidly shift to generating alpha from data instead of infrastructure using modern and advanced data science techniques and the ability to differentiate themselves at the infrastructure level becomes prohibitively hard and costly, this low developmental agility and high time to market adds a significant opportunity cost to the already high monetary cost making such an investment untenable.

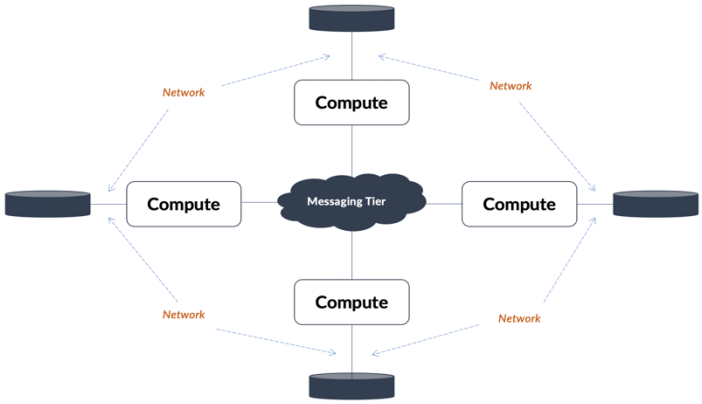
*It is due to this that a **one of the world's largest investment banks**, decided to embark on a digital transformation project to rebuild its equity trading system. The current system's infrastructure layer was homegrown and, for the reasons described above, had become extremely brittle, highly fragmented, hard to add features to, exhibited performance that was significantly lower than needed, was losing data due to outages and a sub-par reliability machinery and had a massive hardware footprint. The inability to add features, the high cost of hardware, operational costs and penalties due to outages and performance SLA breaches, the overall unreliability of the system and the problem being exacerbated by the acquisition of additional systems via mergers and acquisitions during the Great Recession ultimately resulted in the bank launching an initiative to revamp its equity trading system across all trading flows.*

Case Study

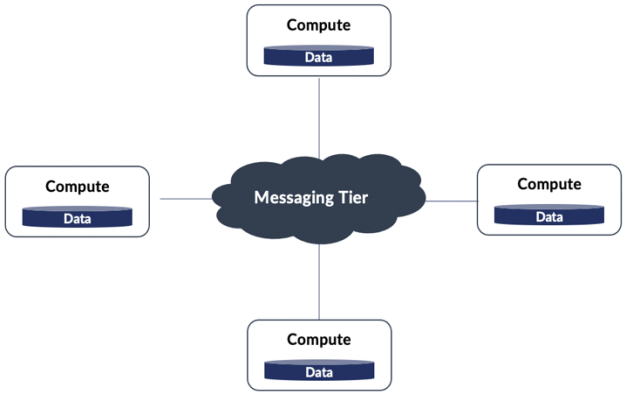
Equity Trading Digital Transformation – Tier 1 Investment Bank

THE CORE ISSUE

Traditional application architectures are multi-tiered in nature. They maintain data and compute in distinct architectural tiers. The compute and data tiers are separated by a network and compute nodes use a messaging tier to collaborate and orchestrate the execution of business function. In addition, there are tiers to manage and monitor these systems. Foundational technologies, such as databases, NoSQL data stores and data grids are used to implement the data tier; compute frameworks and containers used for the compute tier; messaging frameworks used to implement the messaging tier and specific database, charting and visualization technologies used to implement the management and monitoring tier. These technologies implement the infrastructural elements – such as data persistence, data consistency, encoding/decoding, horizontal scaling, message passing, message routing, guaranteed delivery, graphing and reporting – for each of their respective tiers and, in doing so, serve as the infrastructural substrate for the application specific portion of each of the tiers.



It is the network separation between the data and compute tiers that is the core issue for trading systems. This is because it is impossible to meet the combination of performance, reliability and scalability requirements of these systems due to the cost of shoveling data across the network on demand between the data and compute tier and the sheer volume and velocity of data being moved and processed by these systems. Over the years, these systems have evolved to adhere to a different architecture pattern – a microservices architecture pattern – comprised of several processing nodes, collaborating with each other using message passing and *each node capable of hosting the compute and the data needed by the node in the same memory space*. This is still a multi-tiered architecture but with the data and compute tiers logically separate but physically fused into a single tier. Thus far, there has been no foundational technology, such as a database in the data tier or a container in the compute tier, that can serve as the infrastructural substrate for the components of applications that adhere to this architecture pattern. That is why the development teams of trading systems needed to innovate and take ownership of the entire application stack.



Case Study

Equity Trading Digital Transformation – Tier 1 Investment Bank

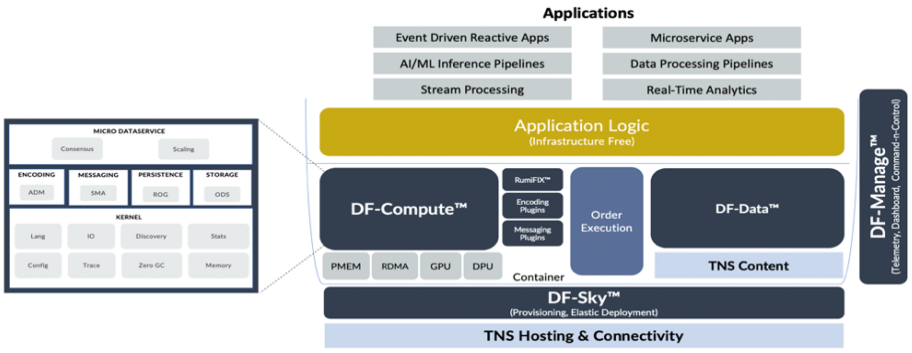
Implementing a layered infrastructural substrate that supports this architectural pattern and, at the same time, is easy to program to is a very complex task. At this bank, the trading system's development team spend several years developing and honing the infrastructural components of their trading system. Unfortunately, despite this investment, the infrastructural components could not keep pace with the latest technological developments and, due to developmental pressures, began to develop cracks that ultimately resulted in the issues the bank faced. This problem was significantly exacerbated by them inheriting systems, acquired through mergers and acquisitions, that had competing infrastructure components and were themselves built using bespoke architectures that were incompatible with the existing system's architectural assumptions.

THE SOLUTION

To solve this problem, one needs a foundational technology for such an architecture pattern, akin to a database in the data tier or a compute container in the compute tier in a traditional multi-tiered architecture. Such a foundational technology needs to handle all the infrastructural concerns of each of the processing nodes in the application leaving the application developer to focus solely on the domain and the business logic as is the case for applications built using the traditional architecture patterns.

INTRODUCING RUMI™

Enter Rumi™. Rumi's goal is exactly this – to serve as the foundational, infrastructure substrate for applications that adhere to this architecture pattern. Its mission is to make it simple to develop, manage and run data intensive, real-time applications, such as trading systems, using such an architecture pattern on an environment of choice, at scale, without compromise on performance or resilience. It achieves this by performing the heavy lifting of implementing all the infrastructure concerns of such applications leaving the developer to focus solely on the domain and business logic.



Rumi™ resides between the server/network/storage substrate and the application's business logic and implements all the software infrastructural needs of such applications. In doing so, Rumi™ serves as the software infrastructure substrate for these applications, handles all infrastructural concerns such as data sourcing, message and data encoding/decoding, persistence, storage, HA consensus, message passing, exactly once semantics, deployment, telemetry, scaling, charting, reporting and other such infrastructure needs while the developer is dedicated to the domain, focuses exclusively on the business and authors infrastructure free business code. This makes Rumi™ a perfect foundational layer for infrastructure intensive trading systems.

Equity Trading Digital Transformation – Tier 1 Investment Bank

THE RUMI™ SOLUTION

A traditional architecture, in which the data and compute tiers are separated by a network, is not capable of meeting the non-functional needs of a trading application. Rumi™ is purpose built to enable such applications to be built using the architecture pattern that eliminates the core issue with the traditional architectures and, therefore, serves as the perfect foundation for these applications.

REQUIREMENTS

As a result, this that the bank selected Rumi™ as the foundation for its next generation equity trading system. The following were core requirements for Rumi™ and the new system.

1. *The bank's trading **development team would only focus on Java based business logic** while Rumi™ handles all infrastructural needs.*
2. *The **business logic should be completely "plumbing free"***
3. *The **business logic should be testable for functional correctness using unit tests independent of Rumi™** and then enabled to run on Rumi™ in a distributed environment purely via configuration.*
4. *The existing system had 10+ OMS instances each supporting a different set of trade flows. The **new system should possess a single OMS that handles all flows**.*
5. *The existing system should be sunset, and the **new system handle all flows within 3 years of the start of the project. First flow should be live is less than 1 year**.*
6. *The new system needs to possess the following performance characteristics*
    a. *High touch trading flows need to **handle at least 10x better throughput than current system***
    b. *Zero touch (electronic and DMA) flows need to **reduce wire-to-wire latency from milliseconds to low double-digit microseconds***
7. *The **system needs to be totally reliable**, with zero data and message loss across process, network, machine and data center failures, with **MTTR (Mean Time To Recovery) from failures in double to low triple digit milliseconds** (except DC failures)*
8. *The **system needs to horizontally scale** with ability to change scale factor overnight in response to high load days such as a Russell Rebalance*
9. *The system needs to **significantly reduce the server footprint***
10. *The system should be able to **draw down and chart application, JVM and server level telemetry**. Application-level telemetry should be possible down to per message basis with negligible impact to application performance.*
11. *It should be possible to **implement the new system incrementally on a flow-by-flow basis** and **operate the new system and old system concurrently during cutover**.*
12. ***Rumi™ should be useable by the different class of services in the system** – low latency/high throughput core trade execution services and middle office services – that have different HA and performance requirements. In other words, Rumi™ should be usable as a common platform across all front and middle office services in the new system*

Case Study

Equity Trading Digital Transformation – Tier 1 Investment Bank

RESULTS

The following is a summary of the results of the implementation:

Category	Result
Time to Market	First flow was implemented in less than 9 months with the first 6 months spent on just implementing business logic independent of Rumi™. This ability of Rumi™ to enable "plumbing free" business logic to be developed and tested independent of Rumi™ with a unit testing toolset of choice is a key capability in enabling the developmental agility observed in this project. Subsequent flows were incrementally moved over to the new system and the existing system was fully sunset in less than 3 years.
Performance	High touch flows were certified to handle >10x throughput than the existing system with the system comfortably handling spikes up to 60x during the high volatility trading activity observed during Covid. Zero touch flows successfully demonstrated >10x reduction in wire-to-wire latency.
Reliability	The system has been running live for multiple years now with failures of a variety of components experienced along the way. The system has demonstrated full recoverability from these failures well within the required MTTR with no data or message loss.
Resilience	During Covid, the market experienced very high trading volatility causing many systems, such as Robinhood, to frequently fail caused by the high load peaks. This new system handled these high load peaks comfortably.
Cost	The bank realized a cost saving of $50M year-on-year due to a combination of reduction in hardware footprint, operational efficiency and elimination of system downtime and performance SLA breaches.
Footprint	The new system successfully consolidated all the flows onto a single OMS and reduced the OMS server footprint from 200+ servers to 6 servers.
Monitoring	Rumi™ demonstrated the ability to tap, deliver and chart application, JVM and server telemetry with minimal intrusion to application performance.
Foundational	Rumi™ was used as the underlying foundation for service with a wide variety of different non-functional requirements including high touch flows, zero touch flows, data analysis flows, risk management flows and middle office flows. This demonstrated the general-purpose nature of Rumi™ in being able to serve as a common foundation across the entire equity trading landscape.

CONCLUSION

Rumi™ is a foundational technology for data intensive, real-time systems in general and trading systems in specific. It supports an architectural model that is designed to satisfy the extreme non-functional demands of such systems, serves as an infrastructural substrate to such systems by implementing all non-functional needs of these systems leaving the developer to focus exclusively on the domain and business logic. The breadth and completeness of the Rumi™ feature set in satisfying the non-functional needs of such applications is clearly demonstrated by the extent of use of Rumi™ as the foundational layer of the next generation trading system of this bank and the clear improvement that it facilitated in this bank across the full gamut of non-functional metrics including performance, reliability, resilience, time to market and cost. In doing so, it clearly demonstrated its suitability in serving as the foundational layer for extreme systems, such as trading systems, that combine real-time performance with processing of large and fast data sets.