THE PROBLEM

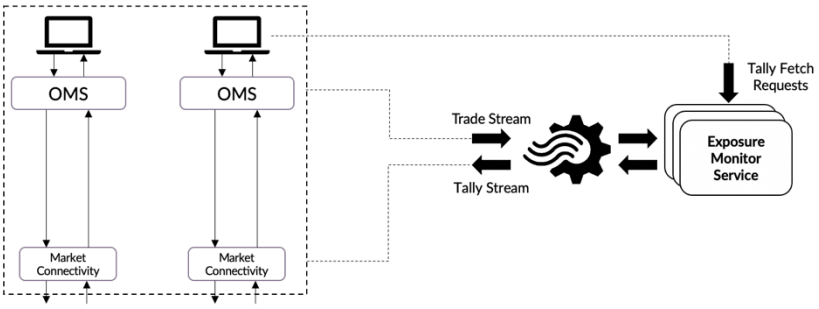
Hedge fund portfolio managers manage their portfolios independent of each other. In particular, each portfolio manager manages the risk profile of the portfolio independently. How orders are sliced to market venues for execution, how long and short positions are kept in balance and how margins are utilized are managed by a portfolio manager independent of other portfolios. However, with many portfolio managers operating concurrently, this can result in imbalanced risk across the entire fund. In order to ensure that the overall exposure of the fund is minimized, the fund needs to monitor all the trade flows across all portfolios and use that information, in conjunction with other information such as market data, to ensure trades are sliced for execution optimally, margins are kept within risk defined limits, the long-short mix of the fund is balanced and, in general, ensure the fund's overall operating risk is within tolerable limits.

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 move very large volumes of trades per second with very low execution latencies. Any decisioning engine that taps into and uses the trade flow to perform any real time, risk related decisioning needs to be able to absorb massive volumes of trades in real time, perform risk calculations and emit risk metrics and signals for downstream consumption in very short time windows. Given that this decisioning logic often depends on the causal relationship between orders, such an engine also needs to operate in a completely reliable and ordered manner ensuring zero loss across process, machine and network failures.

*It is due to this that, when **one of the world's largest hedge funds** embarked on implementing such a decisioning engine, the Fund Exposure Monitor, they found that implementing the infrastructure to handle the performance, reliability and agility demands of this service was much more complex than anticipated, taking too long and taking too much time away from the development of the core decisioning logic. Ultimately, they decided that the ROI of implementing the infrastructure for this service themselves was not high enough and looked to use an off the shelf product, such as Rumi, to serve as the infrastructural underpinning of this service.*

THE EXPOSURE MONITOR

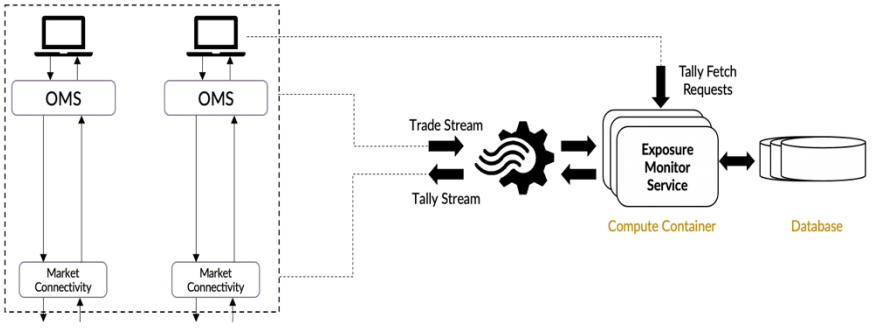
There are several metrics that are calculated to manage a funds exposure. For the purpose of this paper, we will focus on only one metric, called the Tally by this customer. The tally is principally used to determine how to slice and route orders being submitted by the PMs to appropriate trading venues in order to ensure minimal risk exposure of the fund to the trading venues and prime brokerages. The exposure monitor taps into the consolidated trade stream being emitted by the PM trade flows and updates the tallies on receipt of each order and trade. The tallies are used by (1) the PM desk and (2) by the market connectivity engines. For some of the orders, the slicing of the order and re-aggregation of the corresponding trades is done pre-submission of the order to the OMS while, for others, the pre-sliced order is



submitted into the OMS with the slicing and re-aggregation occurring at the edge by the market connectivity engines. For the former, the PM desk requests the exposure monitor for the tallies, uses it to determine how to slice the order, performs the slicing and submits each slice for execution to the OMS. For the latter, the tallies are streamed, in real time, downstream to the market connectivity engines to keep them updated with the freshest tallies so that the slicing logic is done by the market connectivity engines using the most current tallies.

THE CORE ISSUE

To perform the above, the exposure monitor needs to durably ***store*** the computed tallies, ***serve*** the tallies to interested parties, such as the PM desk, ***stream*** tallies to downstream engines, such as the market connectivity engines, and execute the tally computation ***business logic*** on inbound orders and trades tapped from the consolidated PM trade stream.



Services, such as this, are commonly implemented using a two-tiered architecture. This architecture is comprised of a compute tier and a data tier. The compute tier houses the business logic i.e., the tally computation and serving logic and joins the consolidated trade steam to absorb the trades to activate the tally computation. The data tier, that sits across the network from the compute tier, stores the computed tallies and the data needed by this logic and serves the tallies to upstream entities such as the PM desk.

This architecture poses several challenges regarding meeting the performance, scale and reliability requirements of such a service:

- **Performance**
  - For the exposure monitor to be viable, it needs to execute the tally computation in double-digit microseconds or lower. Anything higher than that will cause the tallies to be too outdated at the market connectivity engines and trade desks to meet the risk mitigation SLAs. Each time an order or trade is received, the compute engine needs to reach out across the network and fetch the data needed to perform the tally computation. This is a costly operation that makes meeting this performance requirement a complex and delicate task, if possible, at all. If possible, this requires, at the very least, very advanced infrastructure and delicate engineering that mandates a dedicated investment in infrastructure that, in turn, takes time and money away from developing and honing of the business logic which results in higher to cost with minimal ROI.
- **Scalability**
  - The interaction between the compute and data tier is inherently synchronous. Since the data needed by the compute is fetched across the network on each request, the tally calculation time is high enough (even if the infrastructure is tuned) that the system would need a multi-threaded, concurrent architecture, in conjunction with horizontal scaling of both the compute and data tiers, to scale the

system to handle the high throughput requirements of the exposure monitor. This results in more complexity and high server footprint that, in turn, results in higher time to market and cost.

- **Reliability**
  - Finally, in an architecture such as this, achieving application-level consensus on failures is a complex, non-trivial task that requires assistance from the application layer. Once again, this requires delicate engineering that, in turn, mandates a dedicated investment in infrastructure that, in turn, takes time and money away from developing and honing the business logic which adds to higher cost with minimal ROI.

The core issue here lies not with the sophistication of the data or compute tiers but rather in the fact that the compute and data tier are separated by a network and the interaction between these tiers is synchronous in nature. Procuring a faster database or faster underlying servers or a faster network improves the situation but not in a manner that moves the needle. The hedge fund's experience pushed them to look for a solution that was an orders of magnitude improvement which could not be achieved by independently beefing up the compute or data tiers.

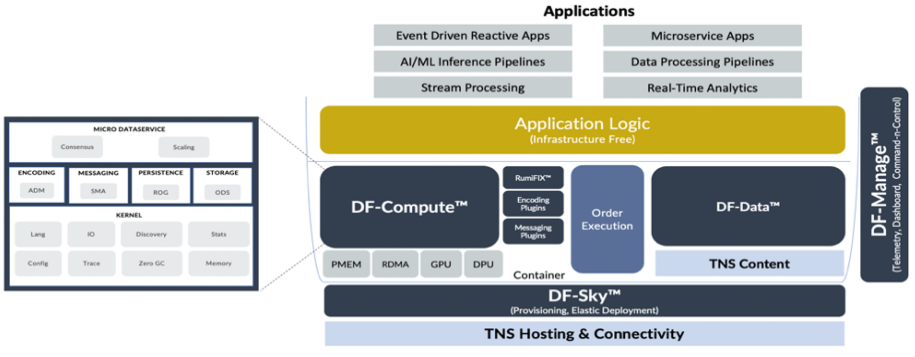
THE SOLUTION

To solve this problem, one needs a mechanism by which (1) the data fetch time is significantly reduced and (2) fault tolerance and zero loss could be handled by the infrastructure without *any* assistance from the application layer. This would enable one to meet the performance goals, reduce the scale factor needed to achieve the desired throughput and latency with minimal cost and tighten the development times.

Given the scale factor that the fund's internal solution needed to meet the desired throughput and latency targets, the fetch time would have to be reduced by multiple orders of magnitude.

INTRODUCING RUMI®

Enter Rumi®. Rumi's goal is exactly this – to serve as the foundational, infrastructure substrate for data intensive, real-time applications such as this exposure monitor. Its mission is to make it simple to develop, manage and run data intensive, high throughput real-time applications, such as this service, on an environment of choice, at scale, without compromise on performance or resilience. It is based on the core principle that the core bottleneck in systems that need to process large amounts of data in real-time, such as this exposure monitor service, lies not in the data or compute tiers of the application but rather in the fact that these tiers are separated by a network. It is impossible to meet the combination of performance, reliability, scalability and agility requirements of these systems due to the cost of shoveling data across the network on demand between the data and compute tiers and the sheer volume and velocity of data



being processed by these systems. While other vendors focus on accelerating either data or compute, Rumi solves the problem by focusing on both. It brings together data and compute into a single hyper-converged software node thus eliminating the network between these tiers. A Rumi node is both, a first-class data node and a compute node. By eliminating the network between the data and compute, Rumi supercharges the amount and speed at which data can be processed, analyzed and served.

THE RUMI® SOLUTION

A traditional architecture, in which the data and compute tiers are separated by a network, is not capable of meeting the combination of the performance, reliability and time to market requirements of this exposure monitor service. Rumi is purpose built to enable such applications to be built such that it eliminates the core issue with the traditional architectures and, therefore, serves as the perfect foundation for such applications.

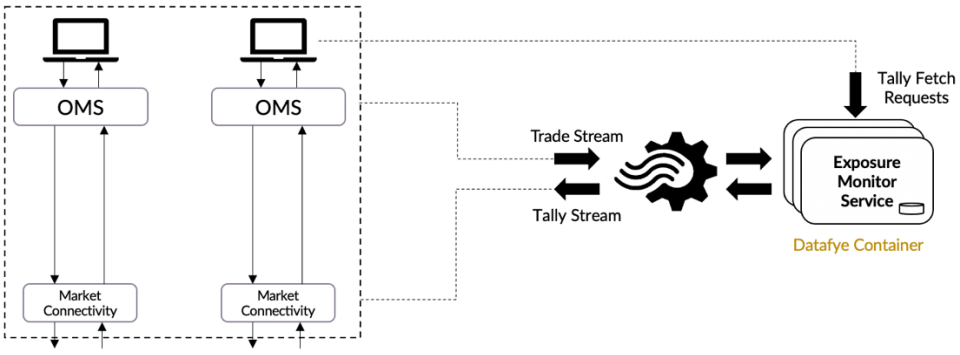
Requirements

As a result, this hedge fund selected Rumi to implement their exposure monitor service using the Rumi technology stack. The following were core requirements for Rumi

1. *The fund's **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 service should be in production within 3 months of start of development*
4. *The new system needs to possess the following performance characteristics*
    a. ***> 1M tally fetches per second** with affordable footprint/cost*
    b. ***Wire-to-wire tally compute processing latency of <50us***
5. *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)*
6. *The **system needs to be horizontally scalable***

THE SOLUTION

The following depicts the solution built using Rumi.



With Rumi, the exposure monitor service is a hyperconverged node that contains the tally computation logic *and* the data needed for the tally computation. This, in essence, is what enables the data fetch time and, thus, the scale factor for the Rumi based solution to be orders of magnitude lower than the existing implementation. This is because the data fetch time has essentially been brought down to zero. It is the unique ability of Rumi to enable data storage, serving, streaming and business logic processing to be converged onto a single node that enables this.

Results

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

Category	Result
Time to Market	The existing tally calculation logic was reused and ported to the new system. The system was in production within 3 months of start of development.

Performance	Latency
	• Existing implementation: > 1ms
	• Rumi implementation: **<50µs**
	  ○ ➔ **20x** of existing implementation
	Throughput
	  ○ Existing implementation: **<Not Measured>**
	  ○ Rumi implementation: **~6M** orders/sec

Reliability	The system demonstrated zero loss recovery within the stipulated MTTR from network, process, machine and DC failures.
Scalability	The system demonstrated the ability to scale linearly as more cluster partitions were added to the exposure monitor service
Footprint	Number of Servers • Existing implementation: **<Not Measured>** • Rumi implementation: **1 (Primary) + 1 (Backup)** Number of Threads per Serve • Existing implementation: **<Not Measured>** • Rumi implementation: **< 4 Threads**

CONCLUSION

Rumi is a foundational technology for data intensive, real-time systems. It supports an architectural model that eliminates the network between the compute and data tiers of such applications and handle all the infrastructural needs of such applications. As such, it 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 performance, footprint reduction, time to market reduction and cost improvement that Rumi demonstrated in this fund's exposure monitor service clearly demonstrates its suitability in serving as the foundational layer for extreme systems, such as this exposure monitor service, that combine real-time performance with processing of large and fast-moving data sets.