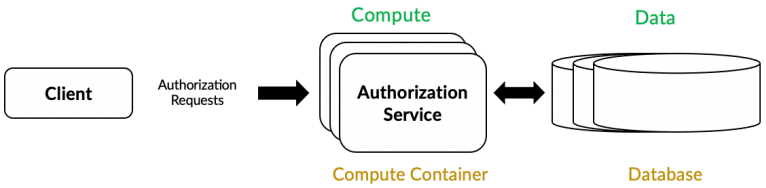
Case Study

Funds Authorization in Retail Banking

THE PROBLEM

Funds authorization and payment processing systems, in general, are mission critical systems that need to sustain high request throughputs, have data intensive request processing logic, have low request processing latency budgets and need to exhibit low MTTR and zero loss on failures. Such systems are commonly implemented using a client-server architecture in which the server side is implemented as a two-tiered architecture comprised of a compute tier that houses the authorization request processing logic, and a data tier that stores the data needed by this logic. Clients issue authorization requests to the authorization service that, then, fetches the data relevant to the request from the data tier, executes the request processing business logic using the fetched data and responds accordingly to the client. The synchronous nature of these systems combined with the data intensive nature of the request processing logic and the system mandating low latency budgets makes these systems very resource intensive and, consequently, expensive to scale.



*It is due to this that, when **one of the world's largest retail banking ISVs** embarked on implementing the cloud version of their funds authorization service, they found that scaling the service to meet the throughput and latency demands of the cloud version of the service was prohibitively expensive. They engaged N5 to implement and benchmark their funds authorization service using Rumi as an alternative and modern foundation for their funds authorization service.*

THE CORE ISSUE

In such systems, there are many data elements, each of varying sizes, that are used by the request processing logic. Since the interaction between the compute and data tier on the server side is inherently synchronous, one needs to employ a multi-threaded architecture in conjunction with horizontal scaling of both the compute and data tiers to scale the system to handle high throughputs. Since the amount of data fetched per request is varied and sizeable and the fetch time needs to be in milliseconds to meet the request processing latency budget, each fetch is very resource intensive on the data tier. This results in the system exhibiting a relatively low Transactions Per CPU Core (TPC) and, therefore, scaling such an architecture is very resource intensive and, consequently, quite expensive.

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 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. This cost of scaling the bank's current implementation of their funds authorization service was high enough that they were looking for a solution that was orders of magnitude lower in cost which could not be achieved by beefing up the compute or data tiers.

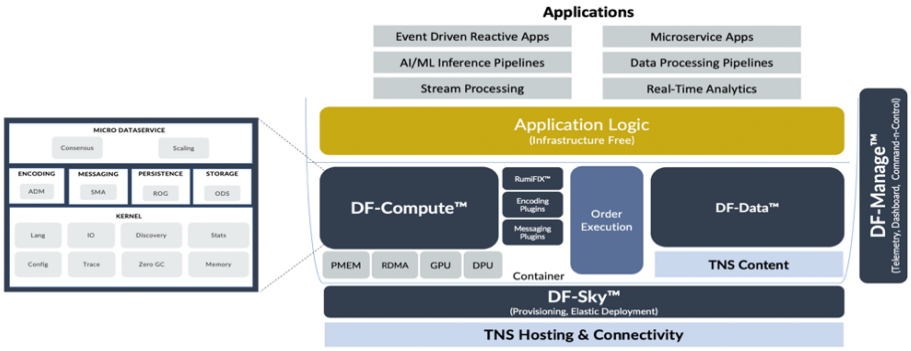
THE SOLUTION

To solve this problem, one needs a mechanism by which the data fetch time is significantly reduced. This would reduce the scale factor needed to achieve the desired throughput and latency.

Given the scale factor of the ISVs existing 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 funds authorization system. Its mission is to make it simple to develop, manage and run data intensive, real-time applications, such as this funds authorization system, 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 funds authorization system, 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 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 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.



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 cost and performance requirements of this funds authorization application. 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 these applications.

Funds Authorization in Retail Banking

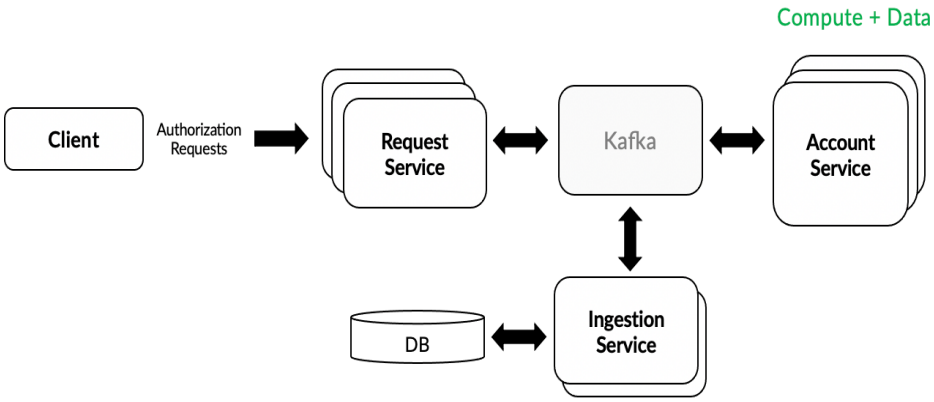
REQUIREMENTS

As a result, this ISV selected Rumi to implement their funds authorization service using the Rumi technology stack. The following were core requirements for Rumi

1. *The ISV'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 new system needs to possess the following performance characteristics*
    a. ***60,000 requests per second**.*
    b. ***Demonstrate ability to scale to 300,000 requests per second** with affordable increase in cost*
    c. ***End-end request processing latency of < 100ms***
    d. ***> 250 TPC (Transactions per CPU Core)***
4. *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)*
5. *The **system needs to horizontally scale***

THE SOLUTION

The following depicts the solution built using Rumi.



With Rumi, the Account Service is a hyperconverged node that contains the request processing logic *and* the data needed for the processing of the authorization requests. This, in essence, is what enables the scale factor for the Rumi based solution to be orders of magnitude lower than the existing implementation since the data fetch time has essentially been brought down to zero.

There is no change to the API offered by the implementation to the client thus making this implementation fully compatible with existing clients without any code change.

RESULTS

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

Category	Result
Time to Market	The existing authorization request logic was reused and ported to the new system. Initial porting sufficient to perform performance tests took 3 weeks.

Performance	TPC
	• Existing implementation: 25 TPC
	• Requirement: > 250 TPC
	• Rumi implementation: **1364 TPC**
	  ○ ➔ **55x** of existing implementation
	Latency
	• Existing implementation: > 300ms
	• Requirement: < 100ms
	• Rumi implementation: **5ms**
	  ○ ➔ **60x** of existing implementation

Reliability	The system demonstrated zero loss recovery within the stipulated MTTR from network, process, machine and DC failures.
Cost	Cost of Underlying Infrastructure (AWS based system) for 60,000 requests/sec • Existing implementation: > $1M/annum • Requirement: >10x reduction • Rumi implementation: **$9k/annum** ○ ➔ **110x** of existing implementation
Footprint	Number of Servers • Existing implementation: 300 • Requirement: >10x reduction • Rumi implementation: **4** ○ ➔ **75X** of existing implementation

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, as such, 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 and cost improvement that Rumi demonstrated in this ISV's funds authorization service clearly demonstrates its suitability in serving as the foundational layer for extreme systems, such as this funds authorization system, that combine real-time performance with processing of large data sets.