WHITEPAPER

MULE ESB Enterprise Performance

Abstract: This document describes highly performant Mule ESB Enterprise production deployments, and explains the architectural features of Mule ESB that contribute to its performance. Performance metrics were captured using industry benchmarks and patterns. Real-world successes of MuleSoft customers validate that the enterprise-proven architecture of Mule ESB Enterprise delivers the performance needed for mission-critical deployment.

Introduction

Assessment of ESB performance and scalability is a complex topic because there are so many possible types of ESB-centered integration deployments. This white paper presents a high level overview of Mule ESB performance, and provides real--world scenarios where Mule ESB Enterprise has been deployed successfully in environments where performance was mission critical.

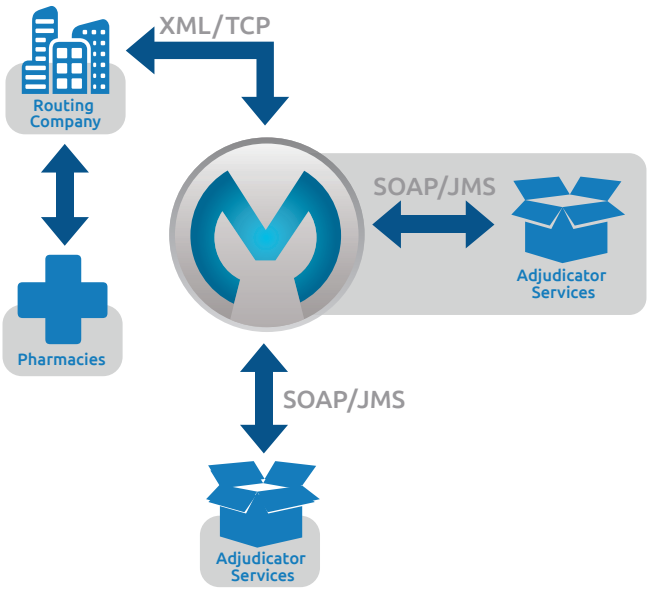
Highly Performant Mule ESB Deployments

Supporting billions of transactions per day, Mule ESB Enterprise is used in production by thousands of enterprises, including MasterCard, Nokia, Nestlé, and Honeywell, and powers integrations with leading SaaS vendors such as Salesforce.com, NetSuite, and Workday. The best proof point of Mule ESB Enterprise performance, therefore, is told through customer success. Selected example stories of these high-performing deployments in mission-critical environments are described below.

Low Response Time

Description

This MuleSoft customer is among the largest pharmacy benefit managers (PBMs) in the United States, processing claims for network pharmacies and their own mail order pharmacies. 60,000 pharmacies turn to this PBM to process 2.5 million transactions per day for filling prescriptions, including prescription adjudication determining insurance coverage for a prescription in real-time. Adjudication is often performed as the patient waits, so response time is critical.



Pharmacies send requests to cover prescriptions via a routing company to the PBM. Requests are sent as XML over TCP. Based on request contents, Mule transforms the data and routes it to on-premises or third-party adjudication servers for various insurers, via JMS queues (for mainframes) or SOAP requests. The adjudicator responds, confirming or denying coverage. The response is then sent back through the routing company to the pharmacy.

Outcome

With additional latency coming from other systems, the PBM's goal was to reduce average transaction processing time from 20 seconds to 5 seconds as part of a larger SOA initiative. In a head-to-head comparison with another vendor, Mule ESB Enterprise delivered significantly improved performance across a range of key metrics in a proof of concept.

Performance scenarios were benchmarked using an HTTP inbound endpoint configured with the request-response pattern, a SOAP-WS consumer, and a simple custom transformer. Initial tests showed an average response time of 6.2ms with 8,000 TPS (which translates to 696 million transactions per day).

Utilizing Mule ESB Enterprise, this reduced average transaction--processing time by 90% from 20 seconds down to 2 seconds, exceeding their project goal and delivering a better experience for retail pharmacies and patients.

On the strength of this performance, MuleSoft's Anypoint™ Platform has been adopted as the backbone for all major integration projects at the PBM, connecting databases, legacy applications, and services.

High Throughput with Low Overhead

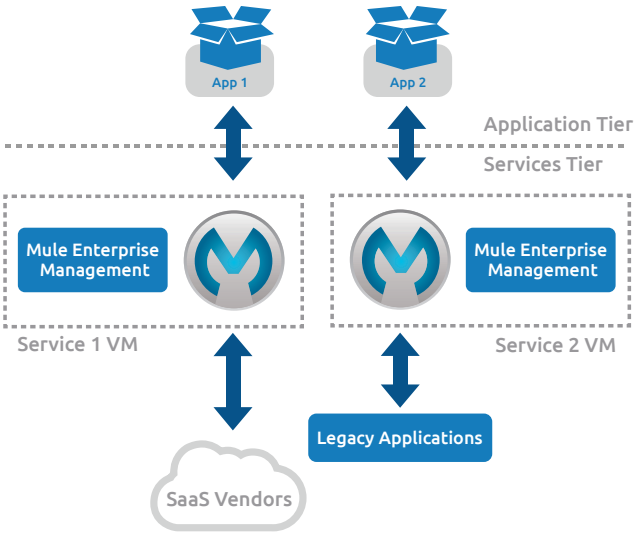
Description

This MuleSoft customer in the financial services industry serves tens of millions of accounts, with local offices and online and mobile services.

The organization standardized on a pure service-oriented architecture encompassing all applications in their data centers across the U.S., with Mule ESB Enterprise as a fundamental element of the infrastructure.

For maximum elasticity and scalability, all services are hosted in virtual machines. Each VM provides endpoints for accessing services; these endpoints are hosted in the VM, on a legacy system, or at a partner outside the firewall. As a particular service needs to scale up, more VM instances are spun up to keep pace with demand.

The following diagram illustrates typical service call paths between applications and services, and the role of Mule ESB Enterprise:



Each service (possibly in multiple versions) is deployed in a VM, along with a Mule application that provides the endpoints for services in that VM.

Outcome

To be a viable solution for such pervasive deployment in this architecture, it was essential that Mule ESB Enterprise be self-contained and easy to set up for horizontal scaling via load balancing as new VMs are spun up to deal with load on individual services. It is also important that Mule ESB Enterprise provides a dashboard for managing service invocation metrics.

Utilizing Mule ESB Enterprise, the customer is able to process almost 10 million messages each day, with a peak rate of almost 600 messages per second and less than 200-ms latency.

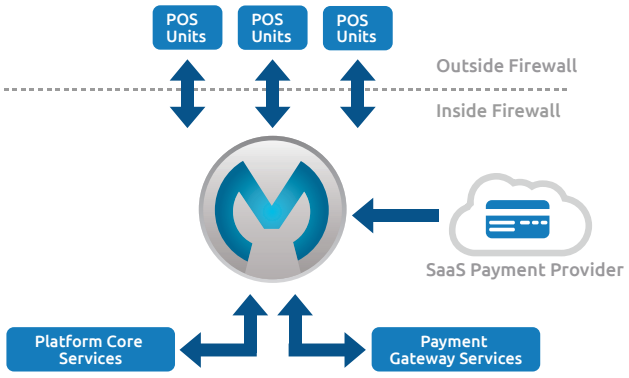
Anypoint™ Platform was chosen by this customer for its easy horizontal scaling, quick self-contained deployment, and manageability for service invocation metrics.

High Concurrency

Description

This MuleSoft customer is one of the leading point-of-sale (POS) solutions vendors worldwide, incorporating deployments both on-premises (Mule ESB on commodity hardware) and in the cloud (Mule ESB in CloudHub) into their primary payment platform. Mule ESB Enterprise serves as a SOAP Web service proxy for all invocations of platform services. Anypoint Platform is used to integrate POS systems, the customer's payment gateways and core platform services, and an external payment processing service.

The following diagram shows the overall architecture:



Outcome

This customer required support for high concurrency with a latency requirement of less than 8 seconds.

Performance scenarios focused on high thread concurrency. Tests were done on a single Mule ESB Enterprise instance running on a 24-core machine. Tests of increasing concurrency up to 800 threads showed an average latency of just under 1 second, sustaining a stable throughput up to 4,000 concurrent threads.

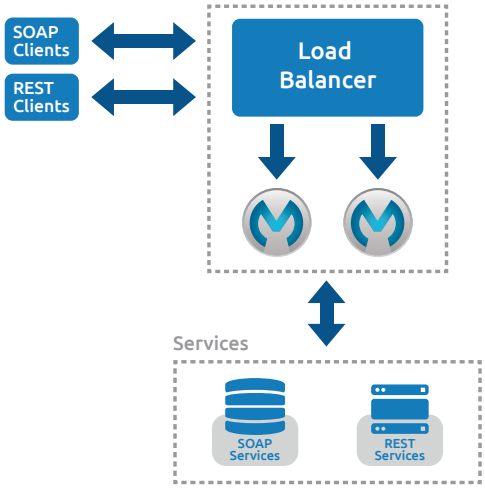
On the strength of these results, this customer created a private cloud offering with hundreds of VMs based on MuleSoft's Anypoint™ Platform. This fully cloud-based service-oriented architecture drives reuse and consistency across all end users and drastically reduced time to market between solutions request and deployment.

Scalable Clustering Solution

Description

This MuleSoft customer uses Mule ESB Enterprise as a Web service gateway. At present most Web services are SOAP, but RESTful services will become more important in the future.

The diagram below presents the general outline of their deployment: Web service consumers call the gateway, which consists of a load balancer in front of a group of Mule instances that call the underlying services.



Outcome

Leveraging Mule ESB's scalable and high performance architecture, this customer was able to process nearly 2 billion transactions per month with a 99.99% success rate. The customer's current deployment handles 85 to 90 million transactions per day (~2100 per second) across 35 services and multiple data centers.

With reliable performance at scale, the customer is able to confidently expose Web services to 3rd party developers and partners. This has unlocked innovation, enabling new products that generated over $100 million. The customer is expecting 5x to 10x growth in message volume in the near future, and is confident that this is possible with Anypoint™ Platform.

Designed to Perform

Mule ESB was architected with performance in mind. The solution delivers high performance in single-node and multi-node deployments, including high-availability active-active clusters that deliver the reliability needed for mission-critical applications. The following section describes the enterprise-proven architectural principles that make Mule ESB the performant solution it is today.

Scaling Up: SEDA Architecture

Fundamental to Mule ESB scalability is its staged event-driven architecture (SEDA). SEDA decomposes a complex, event-driven application into a set of stages connected by queues. SEDA avoids the high overhead associated with thread-based concurrency models, and decouples event and thread scheduling from application logic. By performing admission control on each event queue, the service can be well conditioned to load, preventing resources from being over-committed when demand exceeds service capacity. This ensures that Mule ESB will perform well even under heavy and uneven work load and ensures under extreme load Mule ESB will degrade gracefully. Importantly, Mule ESB was designed to run and scale over commodity hardware because of its low overhead. Stateless applications will scale linearly and stateful applications can be run in up to 16-node clusters with less than 10% overhead. Decomposing services into a set of stages also enables modularity, portability, and code reuse, as well as the use of debugging tools for complex event-driven applications. The latency overhead of the Mule container is between 5 - 7 milliseconds, making it ideal for processing a high volume of small messages as well as larger message processing.

SEDA employs dynamic control to perform basic tuning of runtime parameters (e.g. scheduling parameters of each stage) as well as to manage load through behaviours like adaptive load shedding. Users have control over thread pool sizes and SEDA queue lengths. SEDA queues can be in memory or persisted via diverse means. Users can apply queued-asynchronous flow processing strategies and asynchronous scopes within flows to fully exploit the parallelism available with modern multi-core processors. Mule ESB enables configuring threading profiles globally across your design, or locally on specific flows and message processors.

Refer to the Mule ESB product documentation for relevant information on Mule processing strategies and threading profiles, along with performance tuning basics.

Scaling Out: Stateless Design and Load Balancing

Mule ESB is stateless, so it can serve as a platform for building stateless applications. This makes horizontal scalability for Mule ESB running such applications straightforward, needing only the addition of a load balancer. Adding nodes should result in near-linear scaling, subject to the behaviour of the load balancer and any application state associated with individual message processors.

Scaling and High-Availability Clusters

Mule ESB Enterprise's active-active clustering solution puts a fail-safe foundation under the scalable architecture that can handle the most demanding mission-critical applications without compromising scalability or manageability.

Externally, HA clusters are managed as logical units, much like individual nodes or server groups. Internally, data is shared among all Mule instances, allowing all instances to run the same applications. A data grid shares all transient state information among clustered Mule instances, such as SEDA service event queues and in-memory message queues. Should any Mule instance be lost, the others can handle the load immediately with no loss of service. While there are net performance trade-offs for running in an HA cluster, scaling characteristics are generally similar to those of Mule ESB in non-clustered environments.

Appendix A: Methodology and Environment

Methodology

The performance engineering team at MuleSoft adopts the industry standard for benchmarking performance. For each benchmark test, they first execute the inflection tests, a series of tests with increasing load, to identify the concurrency at which the highest sustainable throughput is achieved. They then execute load tests at the given concurrency to benchmark the performance, followed by stress tests. During all the tests, the system metrics, gc log, and application logs are monitored and collected for further analysis.

To simulate customer's real-world scenarios, the team gathers requirements and studies the input data and environments setup in order to recreate them in the performance lab. Details are input from the field engineers working closely with the customers.

Environment

At MuleSoft, all performance tests are conducted in a controlled environment to ensure stable and repeatable results. It is a dedicated

physical lab with isolated private network with the following configuration:

Mule ESB server	**Red Hat Enterprise Linux Server release 6.5** **24 CPUs** **36GB RAM**	**Java Hotspot 1.7**	**Mule ESB standalone 3.5.0-M4**
Load test server	**Red Hat Enterprise Linux Server release 6.5** **24 CPUs** **36GB RAM**	**Java Hotspot 1.7**	**Apache Bench 2.3** **JMeter**

Appendix B: Performance Benchmark Data

Using a simple web service orchestration scenario as the blueprint, performance tests were run against a Mule application created in the performance lab. The application had the following structure:

HTTP (request response) > SOAP proxy > Data transformation

Using the methodology described in Appendix A, shown below are the results of the performance benchmark.

Mule ESB settings	Threadpool	default
	Heap size	2
	GC settings	default
	Other	max threads 400
Load client settings	Load concurrency	20
Results	throughput (TPS)	8119
	avg response time (ms)	2.5
	90th percentile response time (ms)	3
System resource metrics	avg CPU%	46.1
	max CPU load	2.1
	avg memory utilization (KB)	11
	avg network received (KB/s)	23014
	avg network transmitted (KB/s)	11664
	avg disk utilization %	0.09
	max IO latency (ms)	52
JVM metrics	GC Stall %	3.1
	Obj allocation MB per second	1759
	Thread counts	104
Resource consumptions per request	avg memory needed per request (KB)	221.8
	avg cpu used per request (% of 1 core)	0.14

Table 1: Benchmark report

The charts below show how the system scales with small and large payload sizes with varying amounts of concurrent threads.

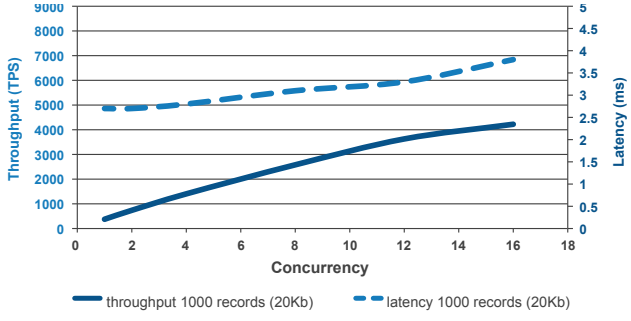


Figure 1: Concurrency vs Throughput/Latency with Large Payload

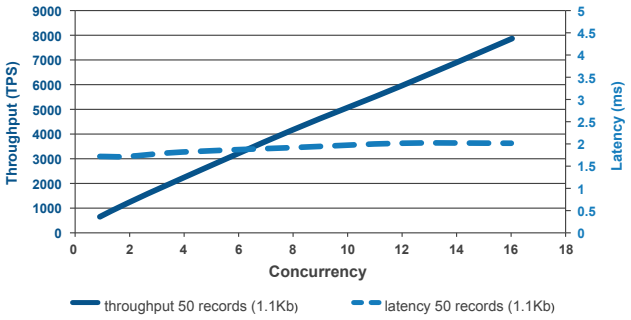


Figure 2: Concurrency vs Throughput/Latency with Small Payload

For both small and large payload sizes, although the respective throughput and latency differs in value, they both share the same trending characteristics as the concurrency and load increase. The throughput lines (dark blue) go up as the concurrency increases, and the latency lines (light blue) only move up marginally. These behaviors validate that the system is stable and linearly scalable.

About MuleSoft

MuleSoft's mission is to connect the world's applications, data and devices. MuleSoft makes connecting anything easy with Anypoint Platform™, the only complete integration platform for SaaS, SOA and APIs. Thousands of organizations in 54 countries, from emerging brands to Global 500 enterprises, use MuleSoft to innovate faster and gain competitive advantage.

For more information:

www.mulesoft.com
info@mulesoft.com
Twitter
Facebook
Google+
LinkedIn