VIZURI

RED HAT JBOSS BPM SUITE PERFORMANCE BENCHMARK

WHITE PAPER – TECHNICAL SUPPLEMENT

The Red Hat JBoss BPM Suite is the JBoss platform for Business Process Management (BPM). It enables enterprise business and IT users to document, simulate, manage, automate and monitor business processes and policies.

This document details a performance benchmark of the Red Hat JBoss BPM Suite (BPMS) version 6.3.

The benchmark tests utilize the KIE Performance Kit and Apache JMeter to execute load tests and gather metrics used to analyze the performance.

Information on the KIE Performace Kit can be found on github at: https://github.com/droolsjbpm/droolsjbpm-integration/wiki/KIE-Performance-Kit

Information of JMeter can be found at: http://jmeter.apache.org/

13880 Dulles Corner Lane
Suite 300
Herndon, VA 21071

(703) 318-7800 Phone
solutions@vizuri.com
vizuri.com

© Vizuri 2016. All rights reserved.

BENCHMARK METRICS

The key metric types utilized for the BPMS benchmark tests are meters and timers.

- Meters - A meter measures the rate of events over time.
- Timers - A timer measures both the rate that a particular piece of code is called and the distribution of its duration.

For the benchmark analysis, we executed several test scenarios over 500-1000 iterations in escalating numbers of threads to capture performance in several BPMS configurations.

The key meters and timers analyzed are as follows.

COMPLETED PROCESSES PER SECOND

Completed Processes per Second values are determined by a meter that is measuring the throughput. This is the number of process instances that are being completed per second as shown in the "Mean Rate" figures. Note that the figures depict "Events/Second".

AVERAGE TIME TO COMPLETE

Average Time to Complete values are determined by timers that measure the time it takes to perform an individual event. These values are aggregated and displayed in the "Mean Time" figures. Key timer events utilized for the benchmark include:

- Time it takes to start and complete a process instance.
- Time it takes to start a process instance and halt at a signal node.
- Time it takes to signal a process instance to the completion of the process instance.
- Time it takes to start a process instance to halt a human task node.
- Time it takes to query for a human task.
- Time it takes to claim a human task.
- Time it takes to complete a human task to the completion of the process instance.

CONTENTS

© Vizuri 2016. All rights reserved.

NOTES ON THE FIGURES

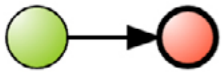
- Figures that include "External Signal" depict an aggregation of both the time spent starting and ending the signal process using the values from individual timers. The arbitrary wait time is omitted.
- Figures depicting Human Tasks are broken down into the individual events that make up the processing of a Human Task. Similar to the Event Signal, any wait time is not part of the readings.

BENCHMARK PROCESS SCENARIOS

The following BPMS Process Definition scenarios were utilized in the performance bench-mark tests. These processes represent common BPMS patterns. They are meant to test the performance of the BPMS platform. Performance in a real world situation can vary based on the processing steps implemented.

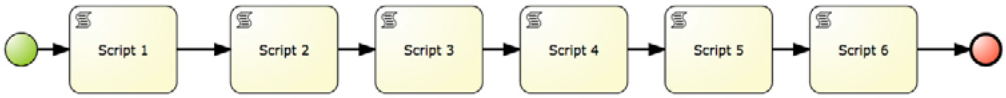
START-END

The simplest possible process, the Start-End task functions as both a sanity check and baseline by only providing the bare minimum.



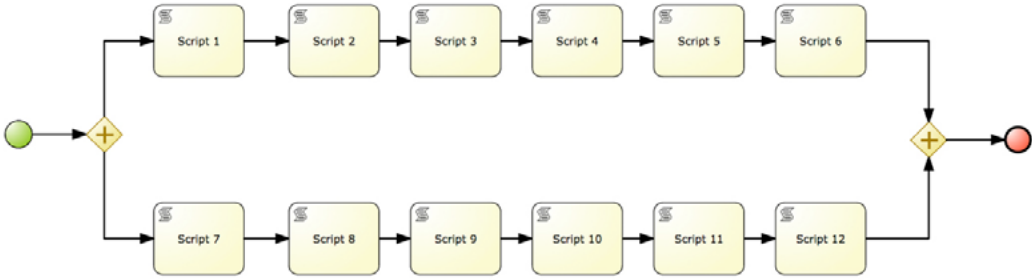
SEQUENTIAL FLOW

The Sequential Flow Process executes a series of script tasks in succession.



PARALLEL FLOW

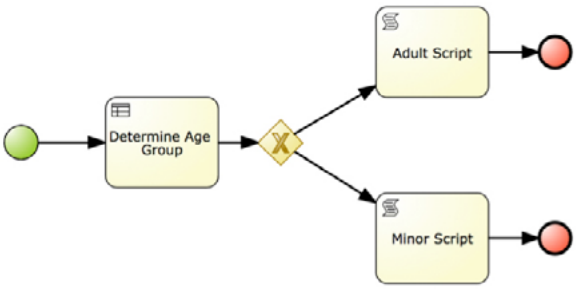
The Parallel Gateway executes two series of script tasks in parallel. This examines the effect of splitting the token.



© Vizuri 2016. All rights reserved.

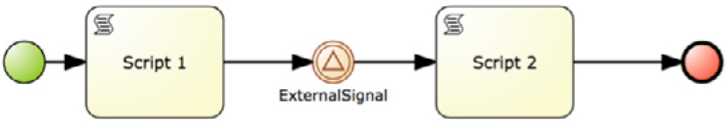
RULE TASK

The Rule Task Process executes a business rule task and performs a decision task based on the results. This highlights the context switch from process to rules.



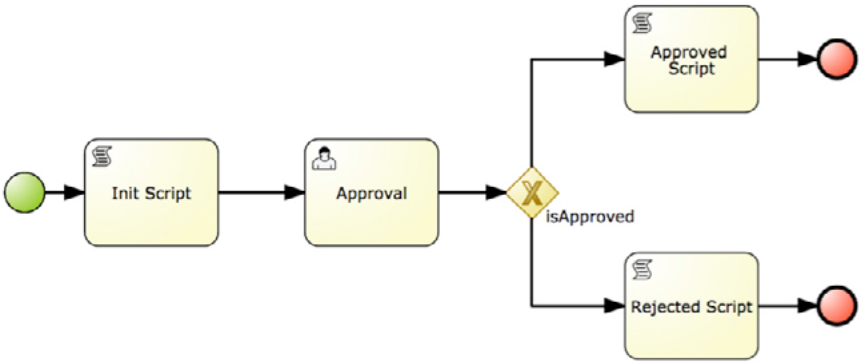
EXTERNAL SIGNAL

The Signal Process waits for a signal from an external source. This produces a "save point" while it waits for an external signal to restart the process. Each save point causes the pro-cess instance context to be serialized and persisted in the database.



HUMAN TASK

The Human Task Process assigns a task to a user. It waits for that user to complete the human task and then performs a decision task based on the input from the user. Like the simple signal, a human task is a "save point", but has more steps as the task is created, claimed and completed.



BENCHMARK TEST SCENARIOS

The BPM Suite supports many combinations of deployment and runtime strategies. The following configurations were utilized for the Benchmark tests to represent the most common deployment strategies.

© Vizuri 2016. All rights reserved.

- Embedded Java API - The Java API executes the BPMS runtime engine "In Process" or embedded in the same Java Virtual Machine (JVM). The Embedded runtime supports the following persistent strategies.
  - No Persistence - Embedded Java API configured with No Persistence or Auditing
  - Persistence with No Auditing - Embedded Java API configured to perform persistence but no auditing.
  - Persistence with Auditing - Embedded Java API configured to perform persistence and auditing.
- Business Central Remote REST API - Execute the BPMS runtime in a BPMS Business Central Web Console utilizing the Remote REST API.
  - Persistence with Auditing - The default configuration for the Business Central is Persistence with Auditing. So this is the only configuration tested for the Remote API.

In addition to the deployment strategies, the BPMS execution environment supports three runtime managers.

- Single - One singleton session is used to execute all requests.
- PerProcessInstance - Each process instance has its own session context; all commands for that process instance are automatically executed in that context.
- PerRequest - A new session is used for each request (and destroyed afterwards).

The PerProcessInstance is usually the best runtime manager to utilize when performance and functionality are a concern.

BENCHMARK TEST RESULTS

The BPMS Benchmark Tests were run in the Amazon Web Service (AWS) environment. Environments were created to simulate a medium and large scale execution environment.

EMBEDDED RUNTIME BENCHMARK RESULTS

The Embedded Runtime tests utilize the KIE Performance Kit to bootstrap a Java Runtime to execute the test scenarios. Each test scenarios is run with 1, 10, 25 and 50 concurrent threads to gather metrics. Each scenario is executed 1000 times by each thread to ensure accurate metrics are gathered.

A thread is not equal to the number of concurrent users. Threads are shared resources that are utilized by all users. Each thread can support many users. As the number of concurrent threads is increased, the time it takes to perform a single task also increases. It should be noted this increase is due to the number of tasks being performed simultaneously. Even though it takes longer to complete a single task, more tasks are performed in aggregate.

MEDIUM SERVER CONFIGURATION TEST RESULTS

The medium server configuration test analyzes the performance of BPMS in a medium sized server configuration.

© Vizuri 2016. All rights reserved.

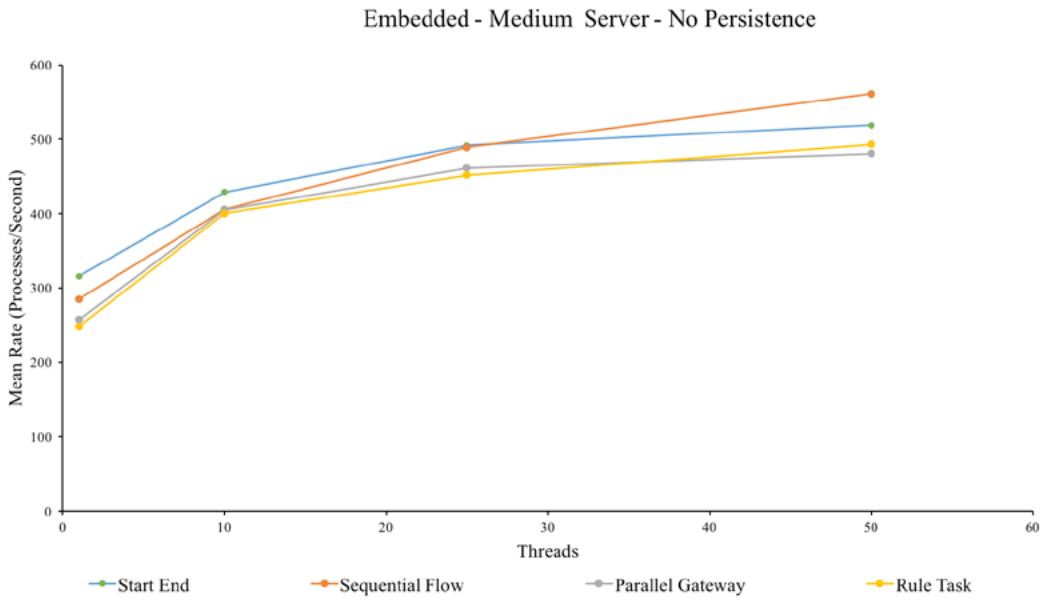
The specification for the medium sized environment is as follows.

DATABASE SERVER CONFIGURATION	
Server Type	db.m4.2xlarge
Database	PostgreSQL 9.4.7
CPU	8 vCPU
Memory	32 GB
Storage	20 GB
RUNTIME SERVER CONFIGURATION	
Server Type	m4.large
Operating System	Red Hat Enterprise Linux 7.2
CPU	2 vCPU
Memory	8 GB
JAVA VIRTUAL MACHINE (JVM) CONFIGURATION	
Version	OpenJDK 1.8.0_91
Heap	6 GB Min/Max
Database Connection Pool	80 Connections
BPMS Runtime Manager	PerProcessInstance

BENCHMARK RESULTS - MEDIUM SERVER CONFIGURATION - NO PERSISTENCE

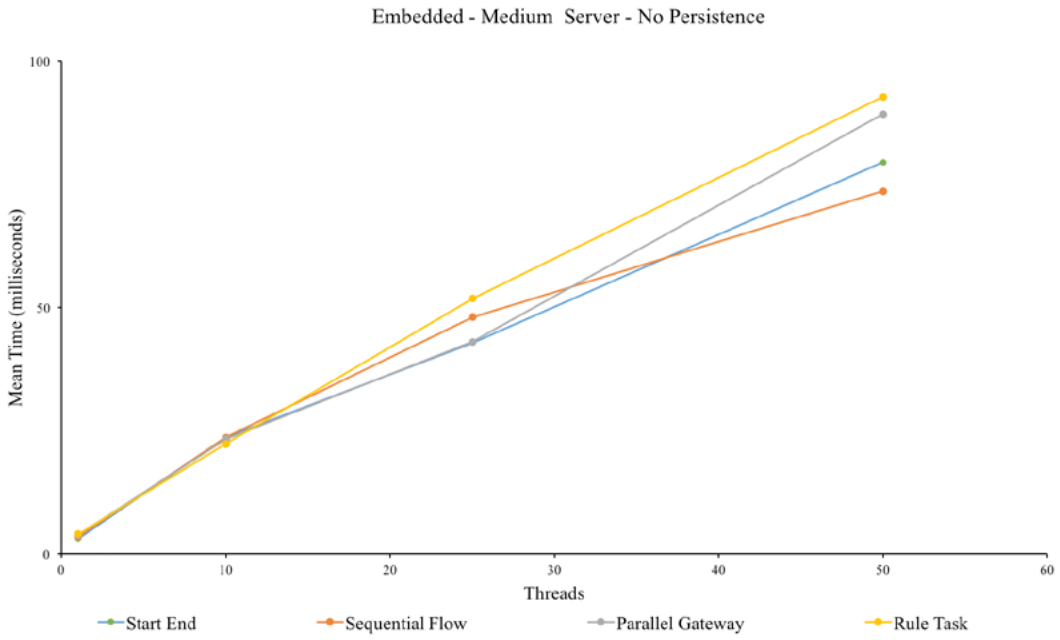
The following are the benchmark test results of the embedded Java API on a medium server configuration with no persistence.

FIGURE 1. PROCESS INSTANCES COMPLETED PER SECOND



© Vizuri 2016. All rights reserved.

FIGURE 2. AVERAGE TIME TO COMPLETE A PROCESS INSTANCE

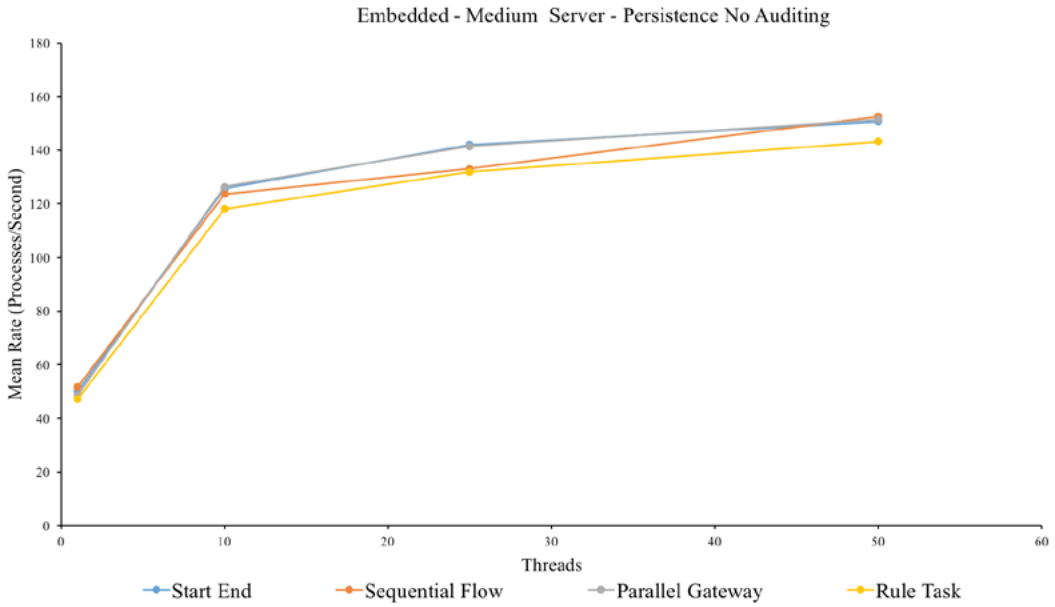


We can see a steady increase in the time it takes to execute a process instance as the number of threads is increased. The throughput benefit of additional threads is maxed out at around 50 threads. This is due to the increase in the time it takes for each process to complete.

BENCHMARK RESULTS - MEDIUM SERVER CONFIGURATION - PERSISTENCE NO AUDITING

The following are the benchmark test results of the embedded Java API on a medium server configuration with persistence but no auditing.

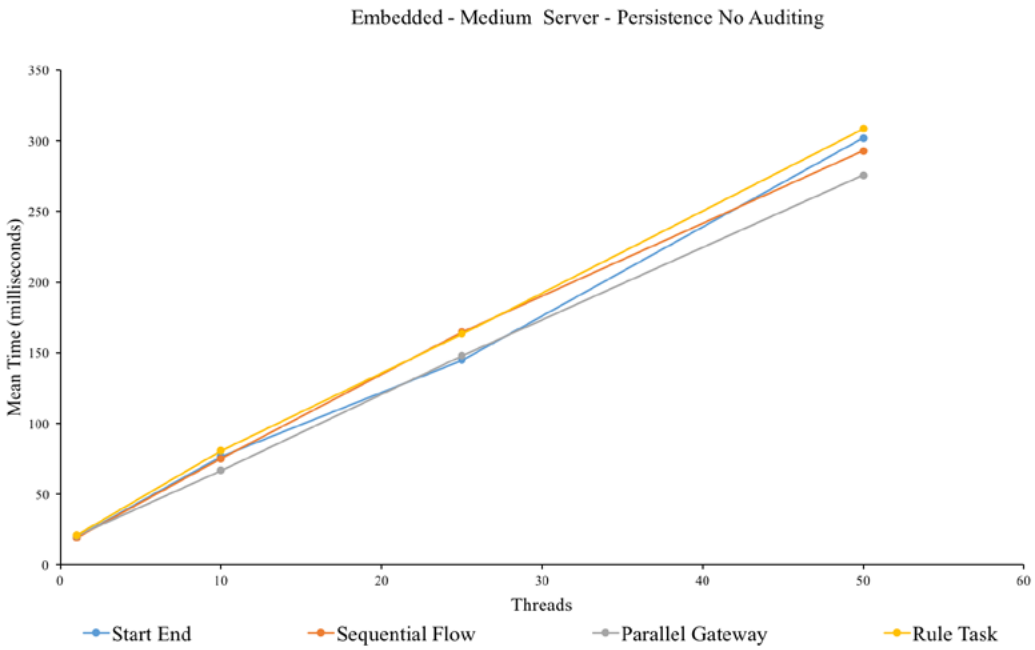
FIGURE 3. PROCESS INSTANCES COMPLETED PER SECOND



© Vizuri 2016. All rights reserved.

The the number of process instances completed per second across all threads in the persistence with no auditing test is 114. This is a 72% decrease over the 418 completed per second in the no persistence test scenario.

FIGURE 4. AVERAGE TIME TO COMPLETE A PROCESS INSTANCE



The average time to complete a process instance over all threads for the persistence with no auditing is 136.22 milliseconds. This is a 247% increase over the 39.27 milliseconds of the no persistence test scenario.

With the addition of persistence, we can add the Human Task and External Signal test scenario, to the performance benchmark. The Human Task and External Signal service of BPMS utilizes persistence to save the state of human tasks and signals as they wait for completion.

The Human Task scenario tracks the mean time it takes to perform three actions.

- Task Start - The time it takes to start the process instance and start a human task.
- Task Query - The time it takes to query and return the task associated with the created process instance
- Task Complete - The time it takes to complete the task and trigger the completion of the process instance.

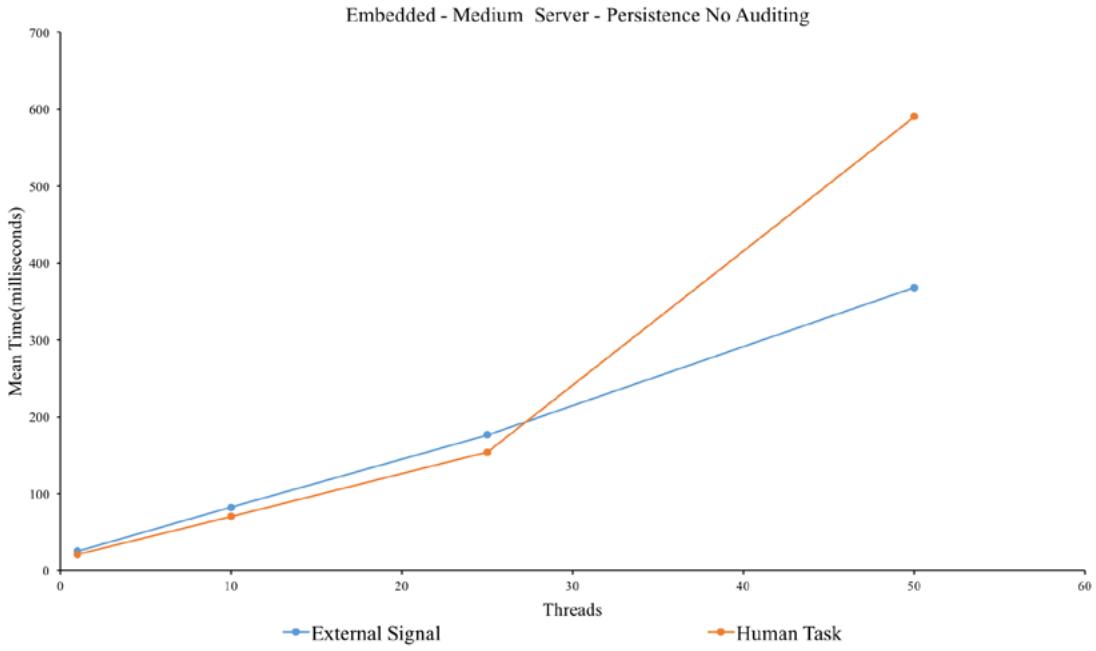
Similarly, the External Signal tracks the mean time to perform two actions:

- Process Start - The time it takes to get to a wait state.
- Process completion - The time it takes to complete once external signal is received.

These "complex tasks" show the aggregated values, excluding wait times not related to processing.

© Vizuri 2016. All rights reserved.

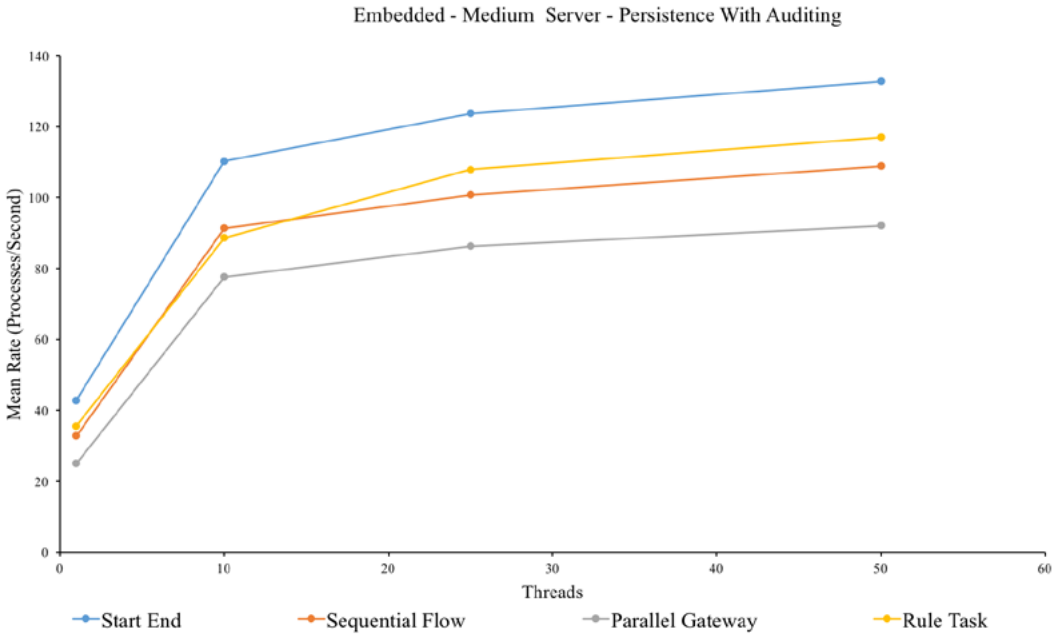
FIGURE 5. AVERAGE TIME TO COMPLETE "COMPLEX" OPERATION



BENCHMARK RESULTS - MEDIUM SERVER CONFIGURATION - PERSISTENCE WITH AUDITING

The following are our findings running the embedded Java API on a medium server configuration with persistence and auditing.

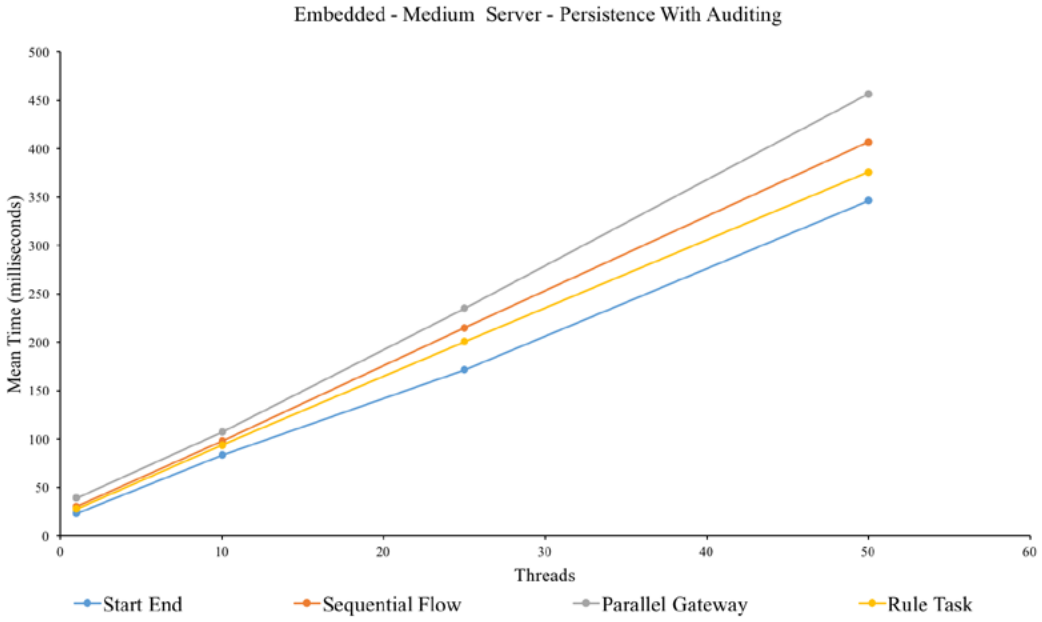
FIGURE 6. PROCESS INSTANCES COMPLETED PER MILLISECOND



© Vizuri 2016. All rights reserved.

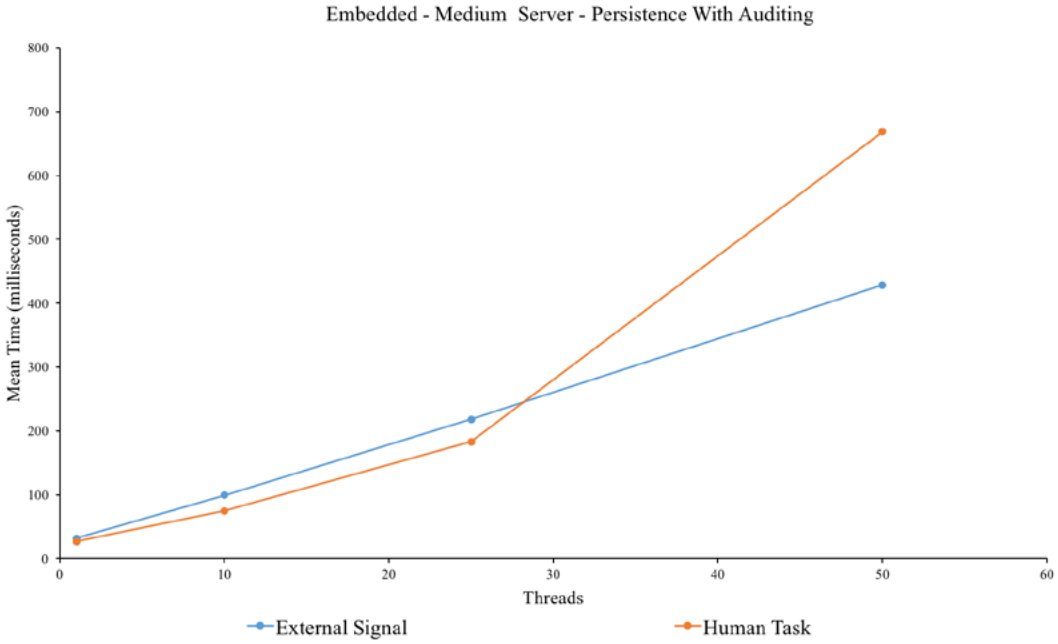
The the number of process instances completed per second in the persistence with audit-ing test is 85. This is a 38% decrease over the 136 completed per second in the no auditing test scenario.

FIGURE 7. AVERAGE TIME TO COMPLETE A PROCESS INSTANCE



The average time to complete a process instance over all thread for the persistence with auditing is 181.73 milliseconds. This is a 58% increase over the 114.88 milliseconds of the no auditing test scenario.

FIGURE 8. AVERAGE TIME TO COMPLETE "COMPLEX" OPERATION



© Vizuri 2016. All rights reserved.

The average time to perform a "complex" event in the persistence with auditing test is 216.3 milliseconds. This is a 16% increase over the 186.03 it takes without auditing.

LARGE SERVER CONFIGURATION

The large server configuration test scenario analyzes the performance of BPMS in a large sized server configuration.

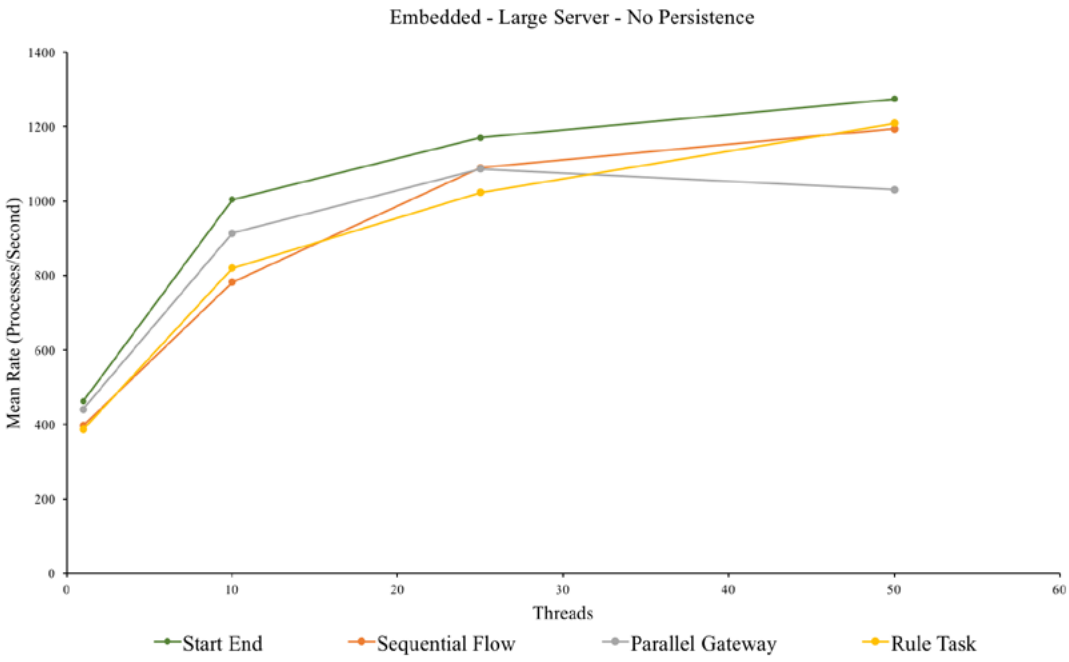
The specification for the large sized environment is as follows.

DATABASE SERVER CONFIGURATION	
Server Type	db.m4.2xlarge
Database	PostgreSQL 9.4.7
CPU	8 vCPU
Memory	32 GB
Storage	20 GB
RUNTIME SERVER CONFIGURATION	
Server Type	c4.xlarge
Operating System	Red Hat Enterprise Linux 7.2
CPU	4 vCPU
Memory	7.5 GB
JBPM RUNTIME AND JAVA VIRTUAL MACHINE (JVM) CONFIGURATION	
Version	OpenJDK 1.8.0_91
Heap	6 GB Min/Max
Database Connection Pool	80 Connections
BPMS Runtime Manager	PerProcessInstance

BENCHMARK RESULTS - LARGE SERVER CONFIGURATION - NO PERSISTENCE

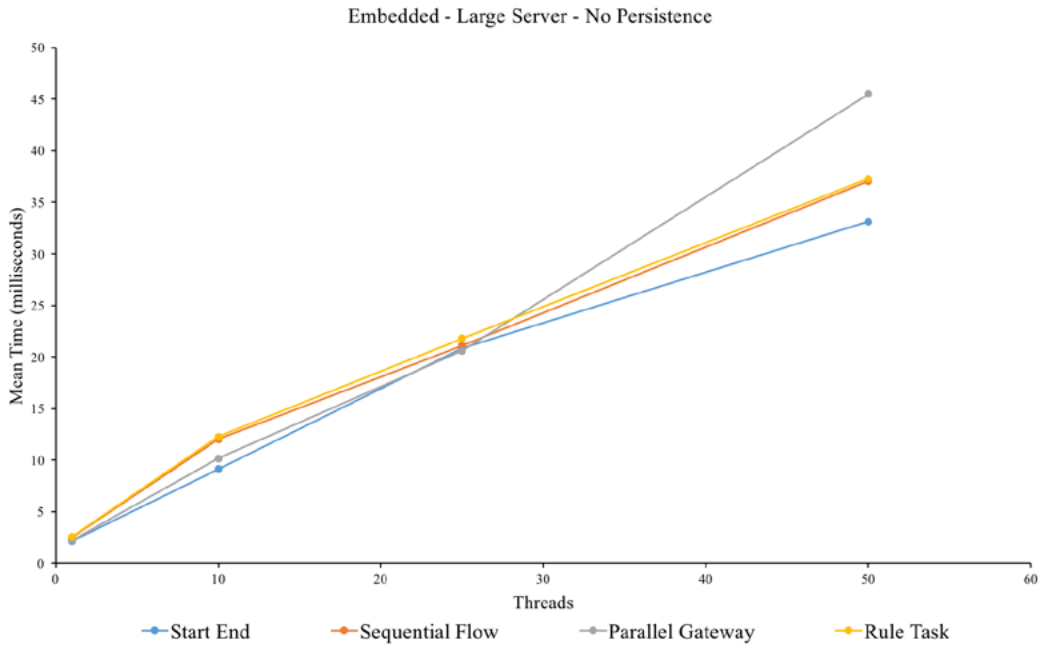
The following are the benchmark test results of the embedded Java API on a large server configuration with no persistence.

FIGURE 9. PROCESS INSTANCES COMPLETED PER SECOND



© Vizuri 2016. All rights reserved.

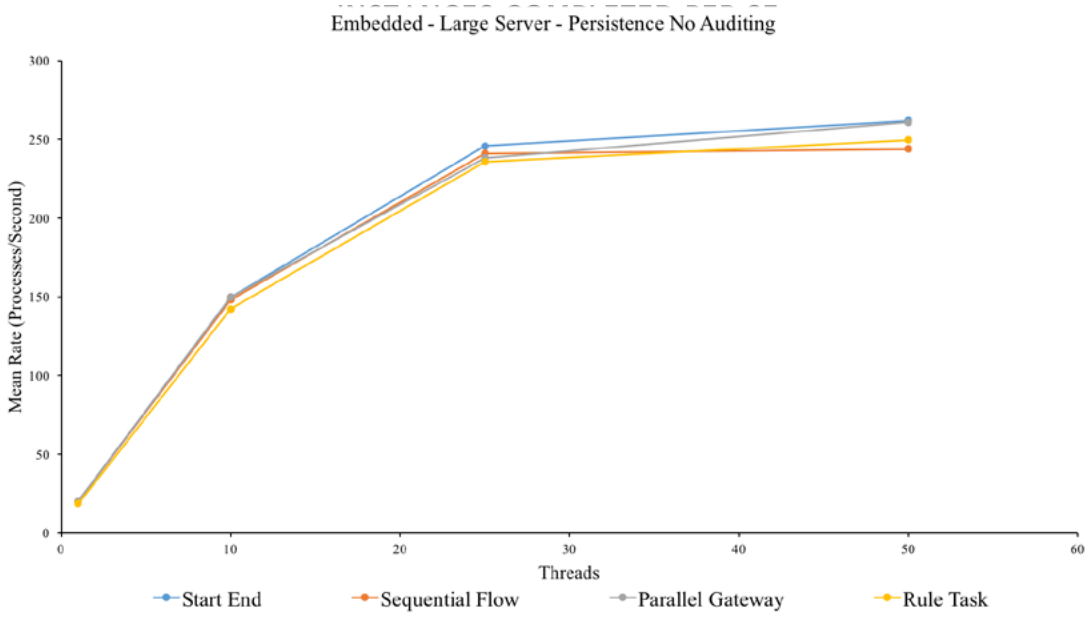
FIGURE 10. AVERAGE TIME TO COMPLETE A PROCESS INSTANCE



We can see a steady increase in the time it takes to execute a process instance as the number of threads is increased. The throughput benefit of additional threads is maxed out at around 50 threads. This is due to the increase in the time it takes for each process to complete.

BENCHMARK RESULTS - LARGE SERVER CONFIGURATION - PERSISTENCE NO AUDITING

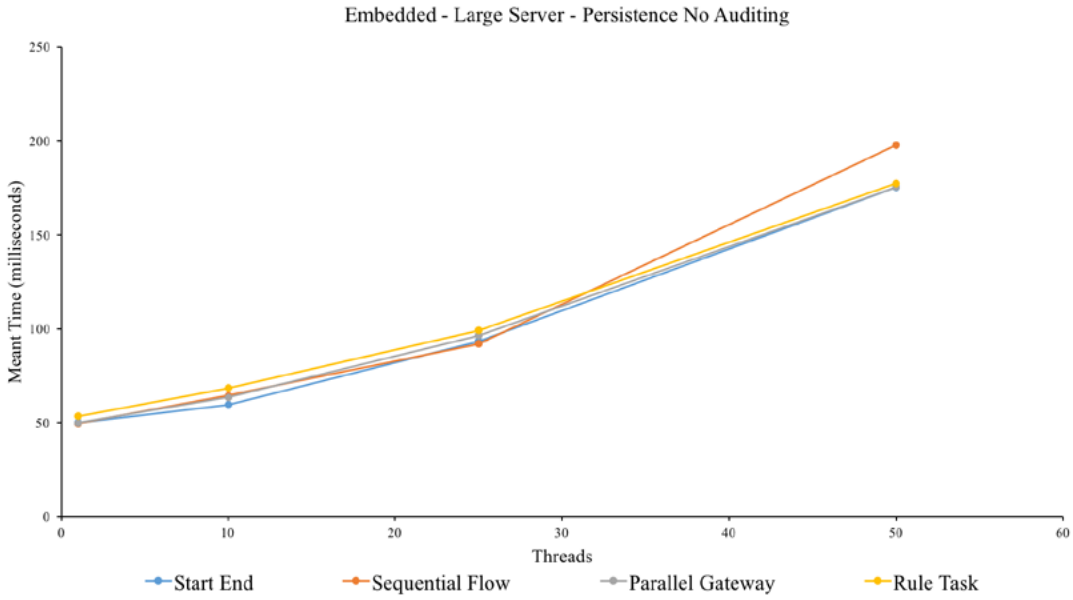
The following are the benchmark test results of the embedded Java API on a large server configuration with persistence but no auditing.



© Vizuri 2016. All rights reserved.

The the number of process instances completed per second in the persistence with no auditing test is 165. This is a 82% decrease over the 892 completed per second in the no persistence test scenario.

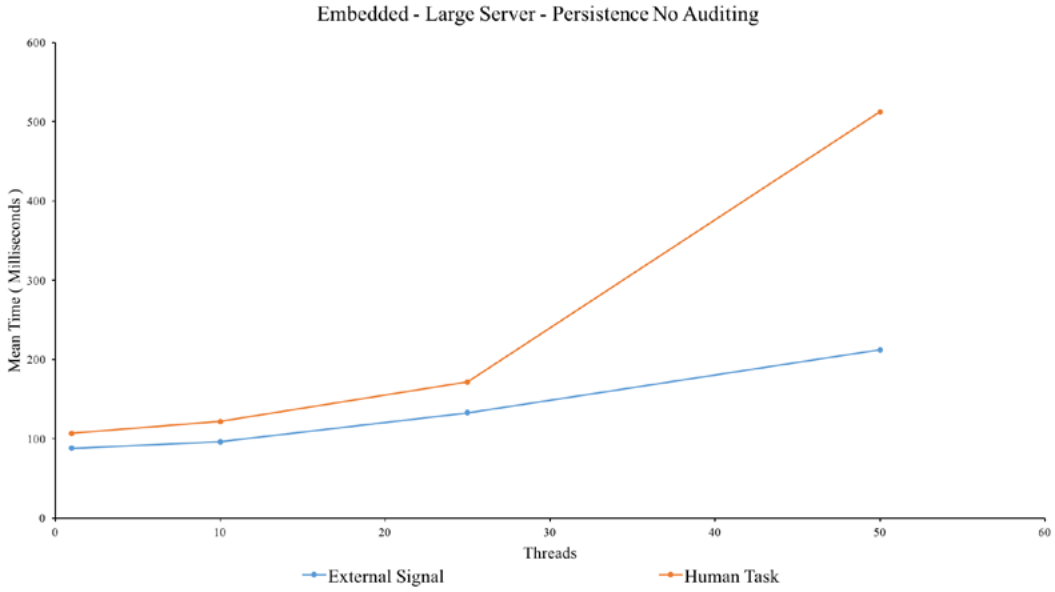
FIGURE 12. AVERAGE TIME TO COMPLETE A PROCESS INSTANCE



The average time to complete a process instance over all thread for the persistence with no auditing is 97.8 milliseconds. This is a 440% increase over the 18.15 milliseconds of the no persistence test scenario.

The figure below show the results of the human task service test scenario.

FIGURE 13. AVERAGE TIME TO COMPLETE "COMPLEX" OPERATION

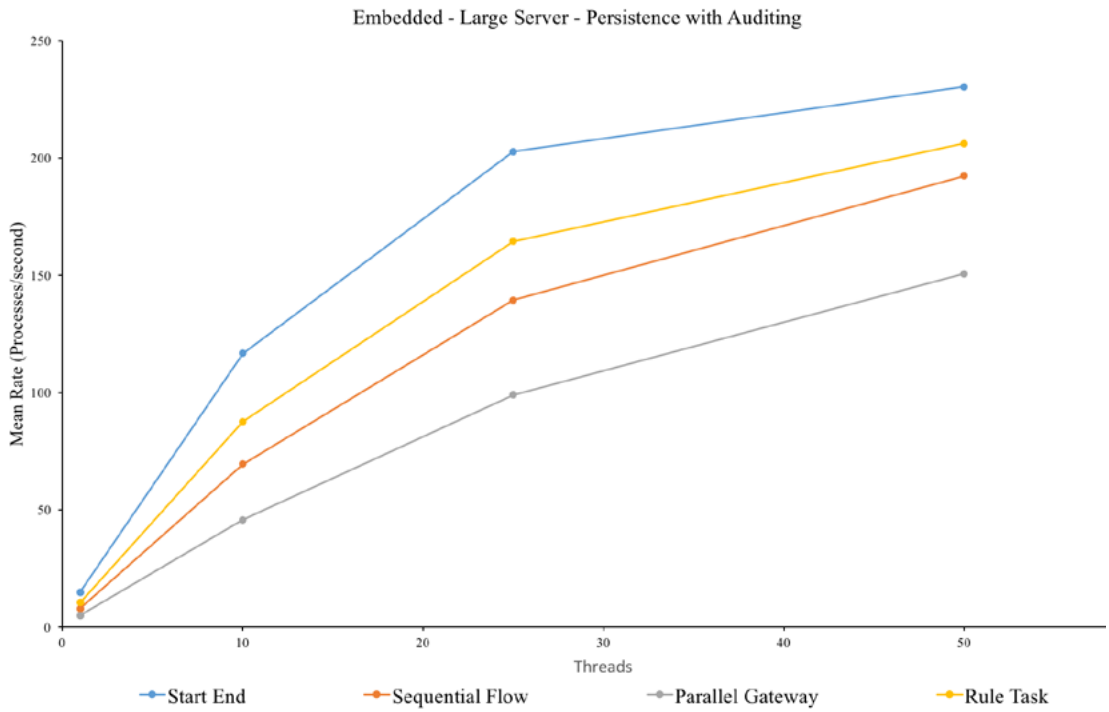


© Vizuri 2016. All rights reserved.

BENCHMARK RESULTS - LARGE SERVER CONFIGURATION - PERSISTENCE WITH AUDITING

The following are our findings running the embedded Java API on a large server configuration with persistence and auditing.

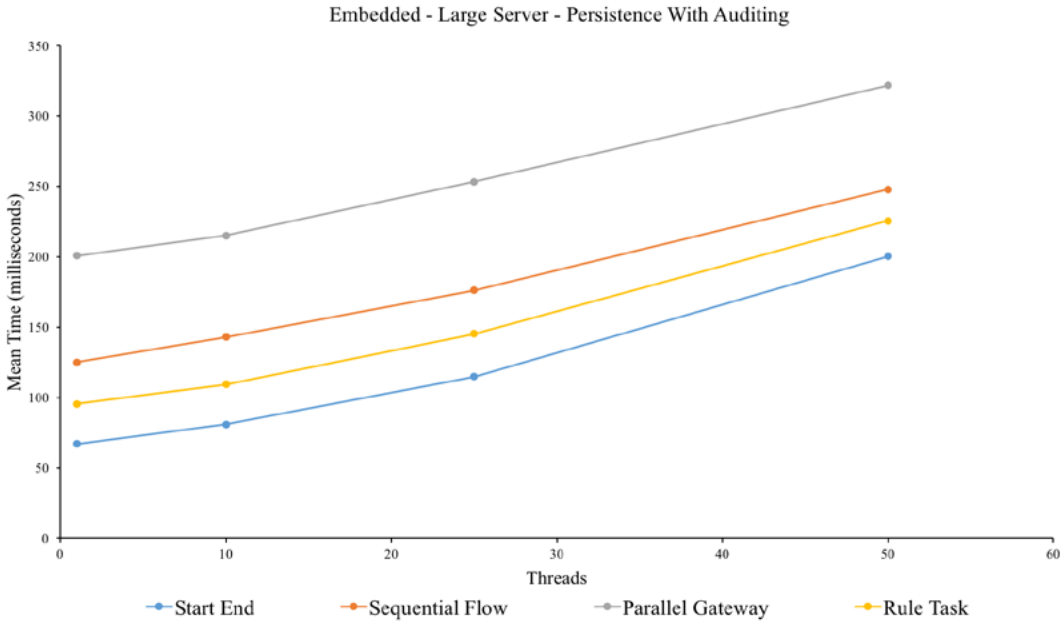
FIGURE 14. PROCESS INSTANCES COMPLETED PER



The the number of process instances completed per second in the persistence with auditing test is 108. This is a 35% decrease over the 165 completed per second in the no auditing test scenario.

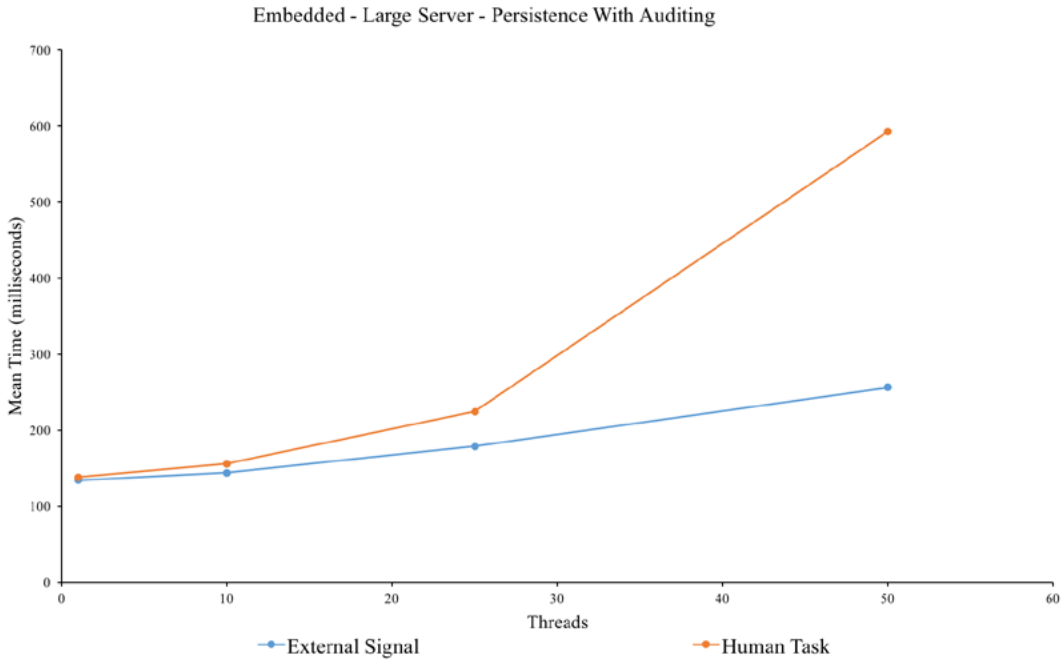
© Vizuri 2016. All rights reserved.

FIGURE 15. AVERAGE TIME TO COMPLETE A PROCESS INSTANCE



The average time to complete a process instance over all thread for the persistence with auditing is 170 milliseconds. This is a 74% increase over the 97.8 milliseconds of the no auditing test scenario.

FIGURE 16. AVERAGE TIME TO COMPLETE "COMPLEX" OPERATION



The average time to perform a human task event in the persistence with auditing test is 228.16 milliseconds. This is a 26% increase over the 180.25 it takes without auditing.

© Vizuri 2016. All rights reserved.

EMBEDDED RENTIME PERFORMANCE RESULTS ANALYSIS

As expected, the embedded runtime performance test shows that no persistence is the fastest configuration followed by persistence with no auditing and the persistence with auditing. This is attributed to the fact that the biggest bottleneck in the BPMS process is the performance of the backend database.

The following figures illustrate the performance difference between the medium and large server configurations as well as the different persistence strategies. These figures show the aggregate of all "simple" scenarios run over all thread counts for a particular configuration.

FIGURE 17. PROCESSES COMPLETED PER SECOND COMPARISON

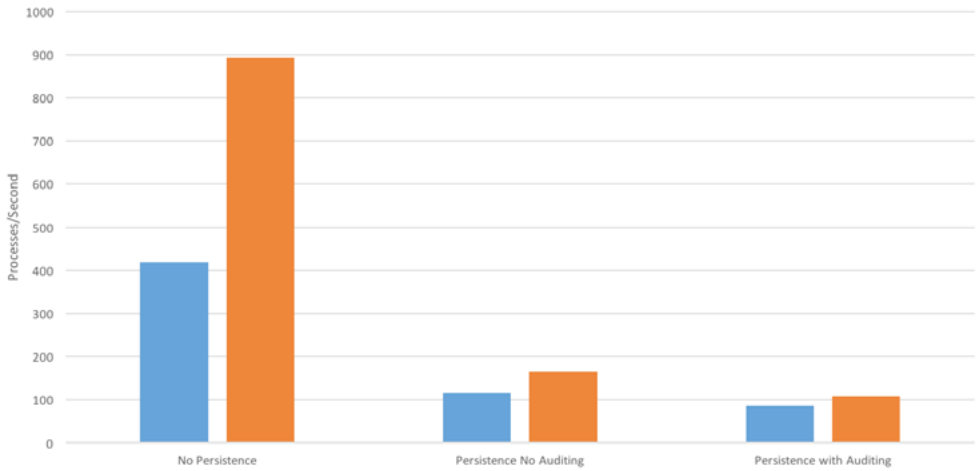
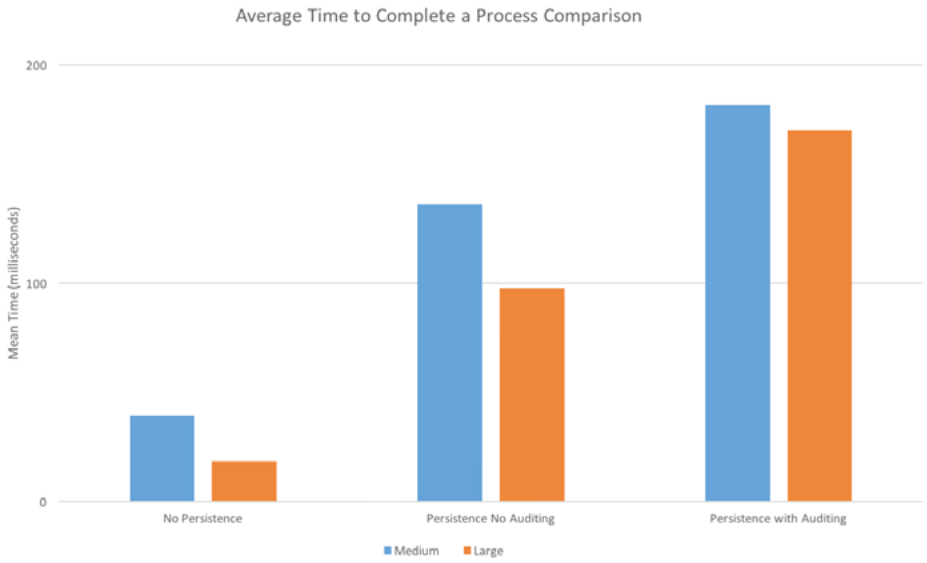


FIGURE 18. AVERAGE TIME TO COMPLETE PROCESS COMPARISON



© Vizuri 2016. All rights reserved.

BUSINESS CENTRAL REMOTE REST API

The Business Central exposes REST and JMS Remote APIs for running business processes. The default configuration of the BPMS Business Central is to enable persistence with auditing so that is the only scenario tested. The REST API is the only remote API tested.

The Apache JMeter load testing framework was utilized to execute the BPMS REST API over 1, 10, 25 and 50 threads to capture metrics to be analyzed. Each thread executes the test scenarios 500 times.

The test scenarios were executed from machines in the same AWS zone to reduce network latency. It should be noted that these test results reflect the time it takes to perform the full round trip REST calls to the BPMS server.

Note that for our considerations, a "medium"configuration is running on an AWS Large instance, whereas a "large" configuration is on an AWS XLarge instance.

MEDIUM SERVER SINGLETON RUNTIME MANAGER TEST RESULTS

The medium server test scenario analyzes the performance of the BPMS business cen-tral in a medium sized server configuration. This first test utilizes the singleton runtime manager, which is the default runtime manager in the Business Central. The singleton runtime manager is not meant to be used to support many concurrent users because all request are processed one at a time through the singleton runtime manager. These tests are meant to show when you should start to consider moving to another runtime manager based on the concurrent load expected in your configuration.

The following server configurations were utilized.

DATABASE SERVER CONFIGURATION	
Server Type	db.m4.2xlarge
Database	PostgreSQL 9.4.7
CPU	8 vCPU
Memory	32 GB
Storage	20 GB
TEST SERVER CONFIGURATION	
Server Type	c4.xlarge
Operating System	Red Hat Enterprise Linux 7.2
CPU	4 vCPU
Memory	7.5 GB
BUSINESS CENTRAL SERVER CONFIGURATION	
Server Type	m4.large
Operating System	Red Hat Enterprise Linux 7.2
CPU	2 vCPU
Memory	8 GB
BUSINESS CENTRAL JAVA VIRTUAL MACHINE (JVM) CONFIGURATION	
Version	OpenJDK 1.8.0_91
Heap	6 GB Min/Max
Database Connection Pool	20 Min 80 Max Connections
BPMS Runtime Manager	Singleton

© Vizuri 2016. All rights reserved.

AVERAGE TIME TO COMPLETE A PROCESS INSTANCE

FIGURE 19. PROCESS INSTANCES COMPLETED PER

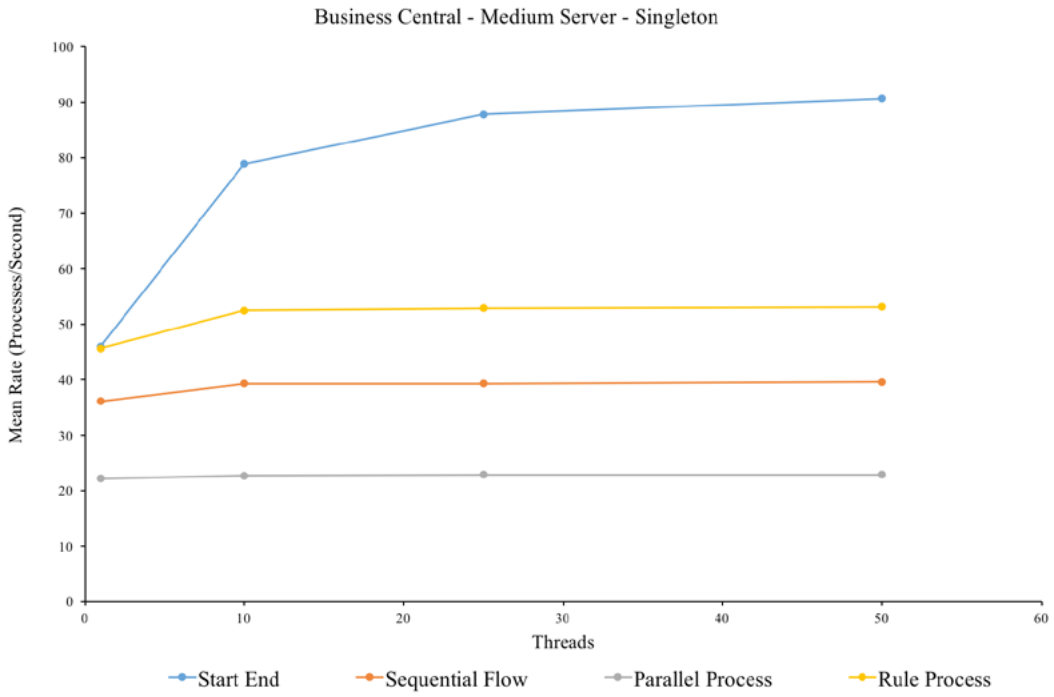
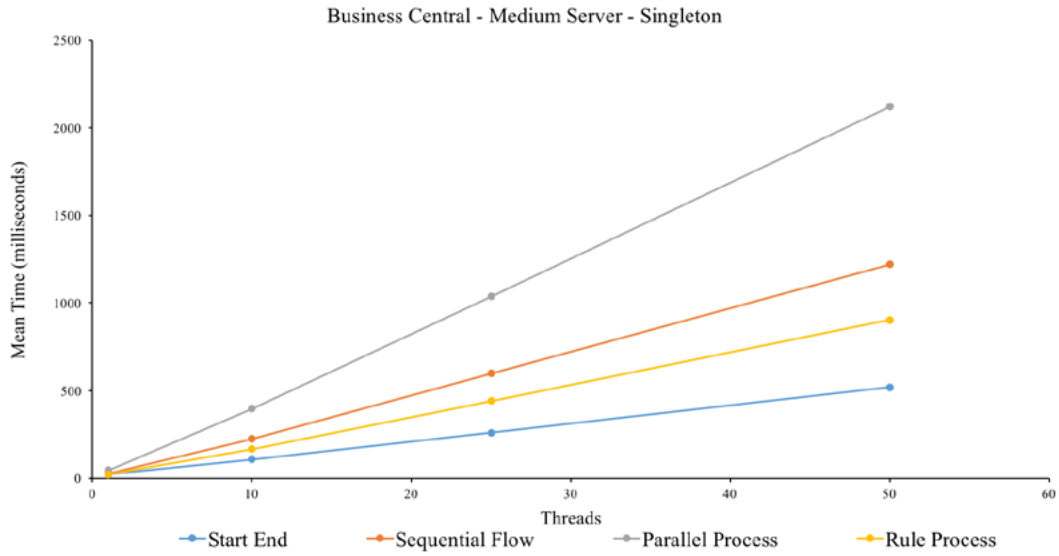
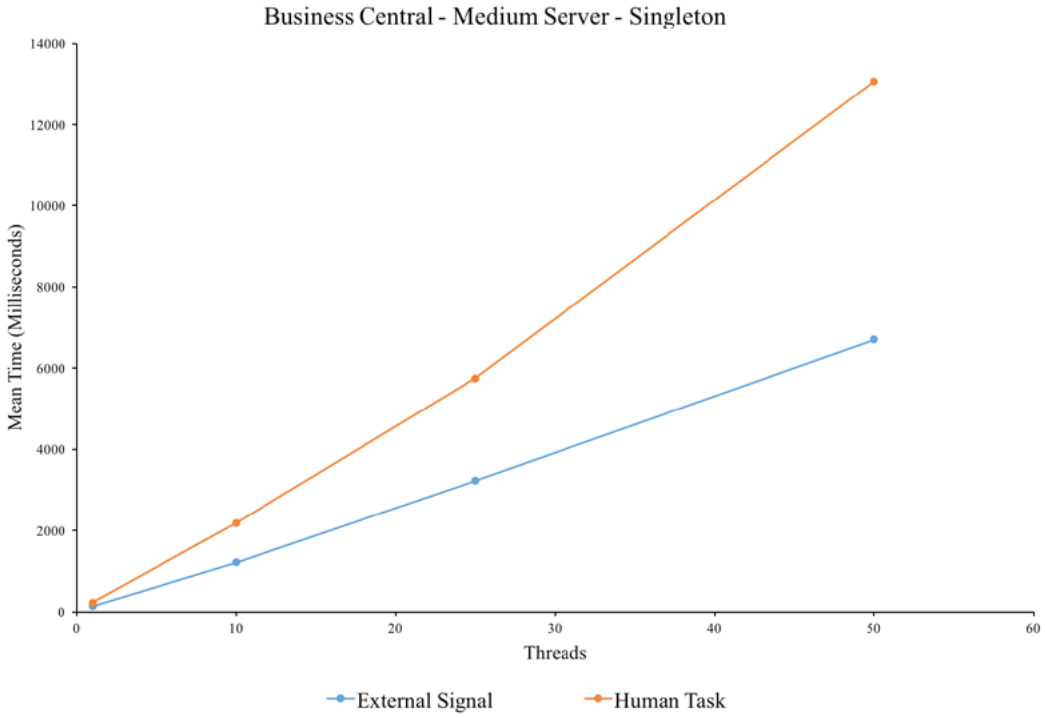


FIGURE 20. AVERAGE TIME TO COMPLETE A PROCESS INSTANCE



© Vizuri 2016. All rights reserved.

FIGURE 21. AVERAGE TIME TO COMPLETE "COMPLEX" OPERATION



As the number of concurrent requests increases in the singleton test case, the response time steadily degradates. This is due to the fact that each request is being processed sequentially by one runtime manager.

© Vizuri 2016. All rights reserved.

MEDIUM SERVER TEST RESULTS

The medium server test scenario analyzes the performance of the BPMS Business Central in a medium sized server configuration.

The following server configurations were utilized.

DATABASE SERVER CONFIGURATION	
Server Type	db.m4.2xlarge
Database	PostgreSQL 9.4.7
CPU	8 vCPU
Memory	32 GB
Storage	20 GB
TEST SERVER CONFIGURATION	
Server Type	c4.xlarge
Operating System	Red Hat Enterprise Linux 7.2
CPU	4 vCPU
Memory	7.5 GB
BUSINESS CENTRAL SERVER CONFIGURATION	
Server Type	m4.large
Operating System	Red Hat Enterprise Linux 7.2
CPU	2 vCPU
Memory	8 GB
BUSINESS CENTRAL JAVA VIRTUAL MACHINE (JVM) CONFIGURATION	
Version	OpenJDK 1.8.0_91
Heap	6 GB Min/Max
Database Connection Pool	20 Min 80 Max Connections
BPMS Runtime Manager	PerProcessInstance

© Vizuri 2016. All rights reserved.

FIGURE 22. PROCESS INSTANCES COMPLETED PER SECOND

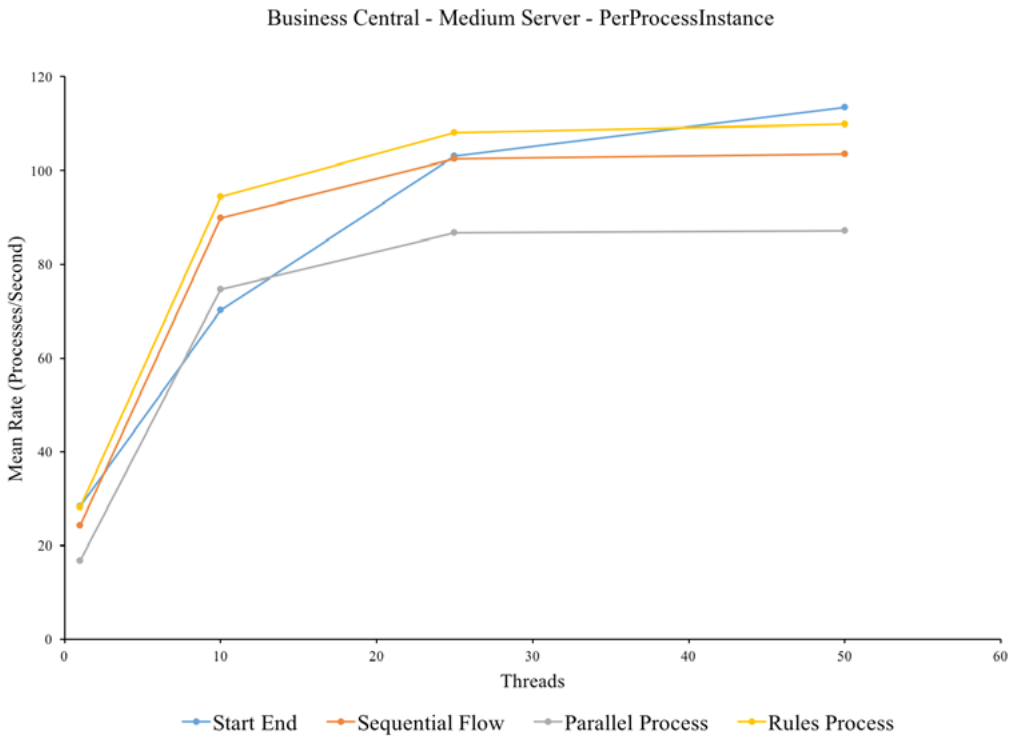
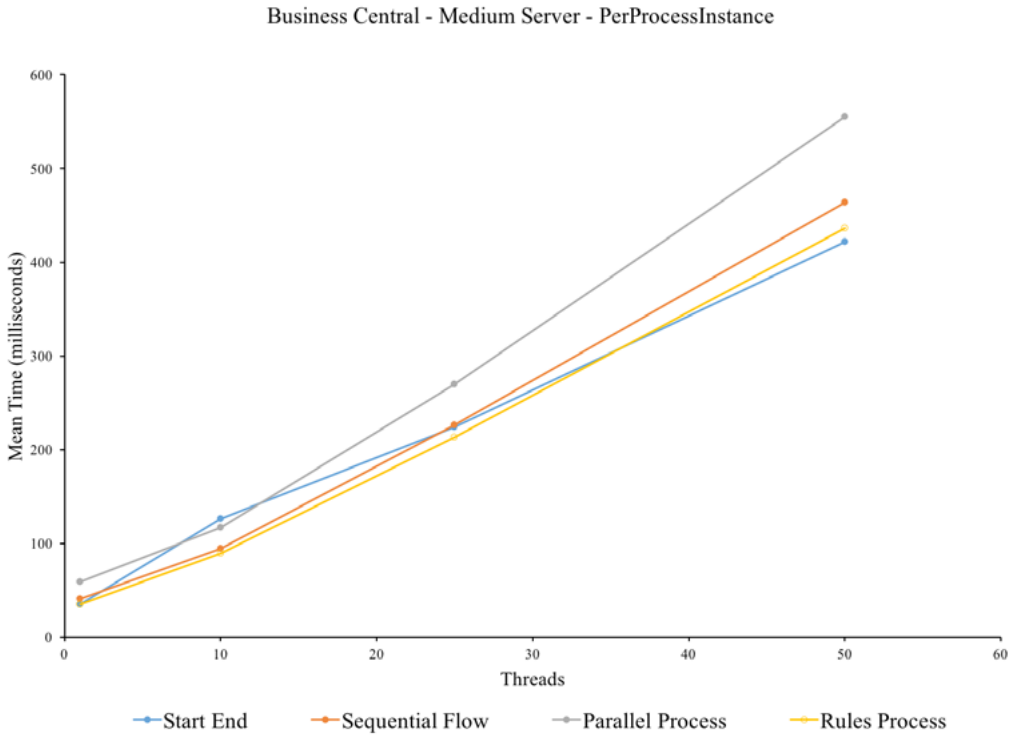


FIGURE 23. AVERAGE TIME TO COMPLETE A PROCESS INSTANCE

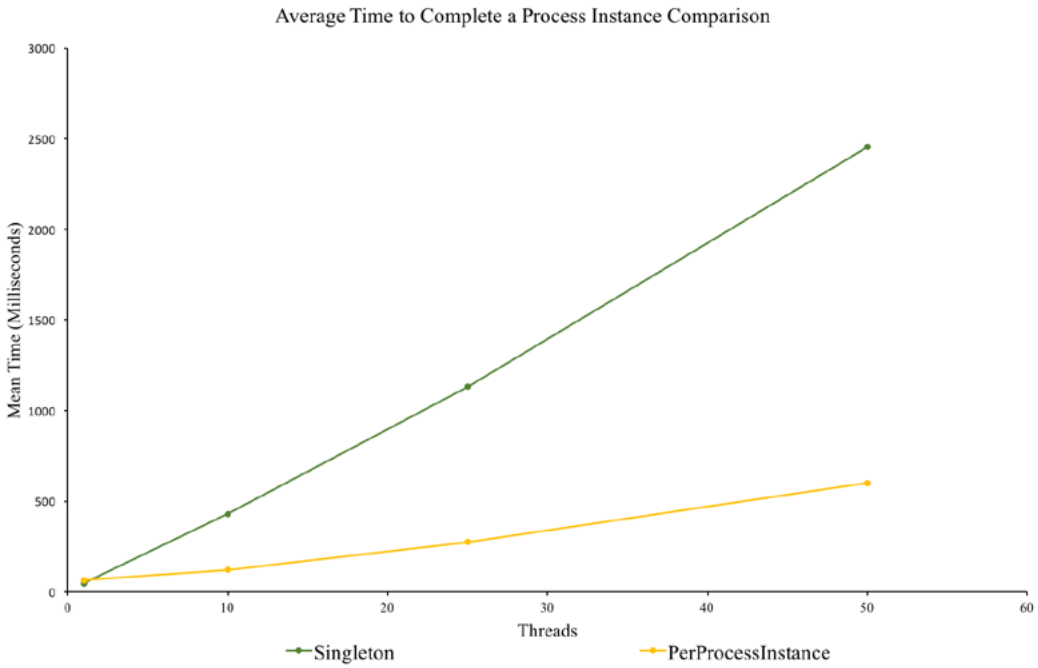


© Vizuri 2016. All rights reserved.

A comparison of the singleton vs per process instance runtime managers shows that the average time to complete a process instance across all threads for the singleton runtime manager is 506 milliseconds while the per process instance runtime manager's average time to complete is 213 milliseconds. As the number of concurrent threads is increased this gap gets bigger.

The following diagram illustrates this.

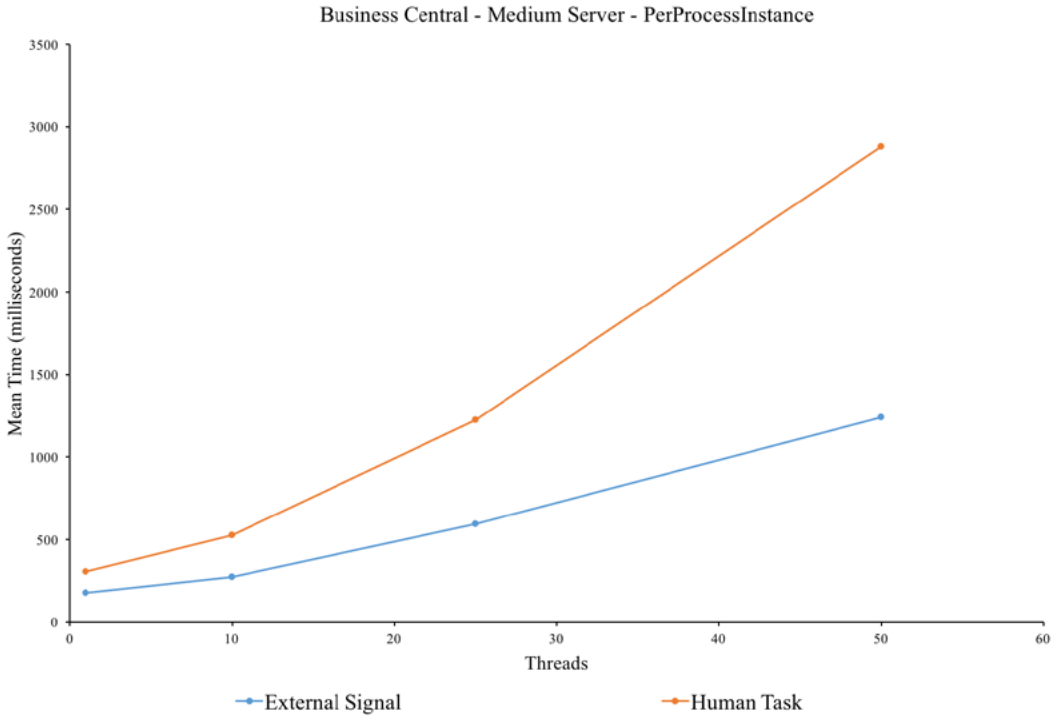
FIGURE 24. SINGLETON VS PERPROCESSINSTANCE COMPARISON



The next graphs show the average times to complete external signal and human task actions using the business central REST API.

© Vizuri 2016. All rights reserved.

FIGURE 25. AVERAGE TIME TO COMPLETE "COMPLEX" OPERATION



LARGE SERVER TEST RESULTS

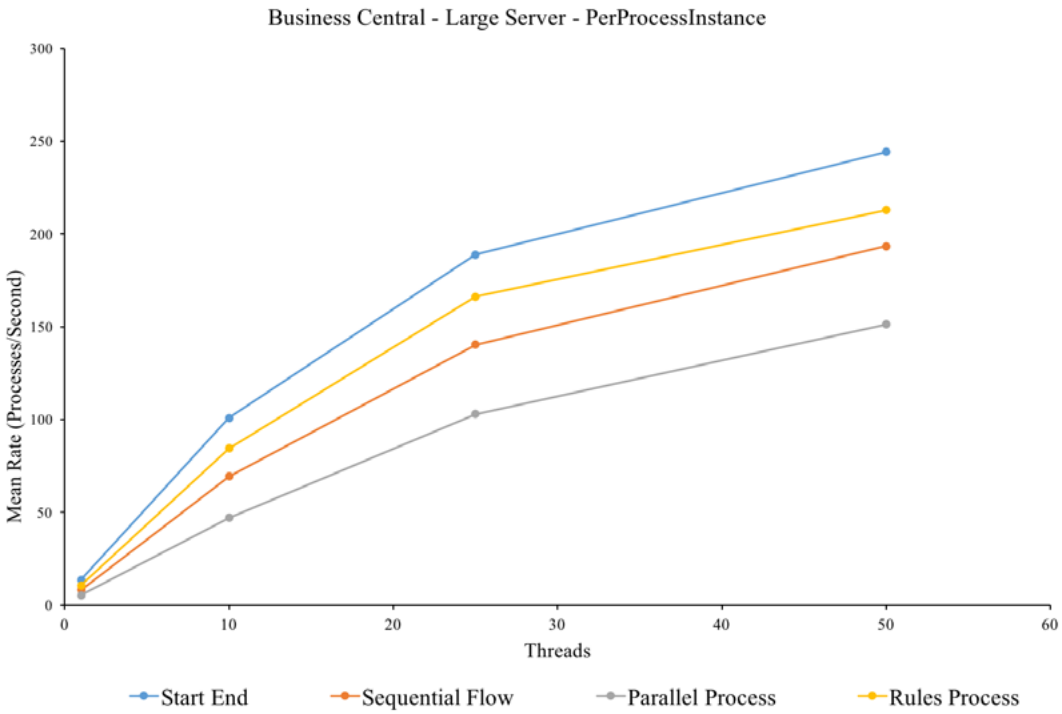
The large server test scenario analyzes the performance of the BPMS Business Central in a large sized server configuration.

The following server configurations were utilized.

© Vizuri 2016. All rights reserved.

DATABASE SERVER CONFIGURATION	
Server Type	db.m4.2xlarge
Database	PostgreSQL 9.4.7
CPU	8 vCPU
Memory	32 GB
Storage	20 GB
TEST SERVER CONFIGURATION	
Server Type	c4.xlarge
Operating System	Red Hat Enterprise Linux 7.2
CPU	4 vCPU
Memory	7.5 GB
BUSINESS CENTRAL SERVER CONFIGURATION	
Server Type	c4.xlarge
Operating System	Red Hat Enterprise Linux 7.2
CPU	4 vCPU
Memory	7.5 GB
BUSINESS CENTRAL JAVA VIRTUAL MACHINE (JVM) CONFIGURATION	
Version	OpenJDK 1.8.0_91
Heap	6 GB Min/Max
Database Connection Pool	20 Min 80 Max Connections
BPMS Runtime Manager	PerProcessInstance

FIGURE 26. PROCESS INSTANCES COMPLETED PER SECOND



© Vizuri 2016. All rights reserved.

FIGURE 27. AVERAGE TIME TO COMPLETE A PROCESS INSTANCE

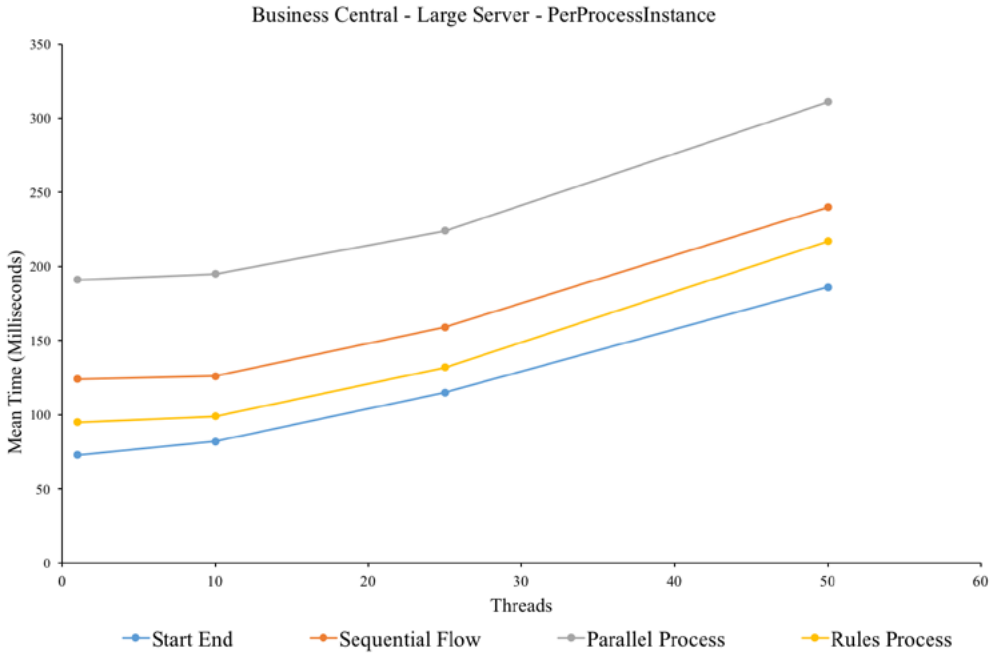
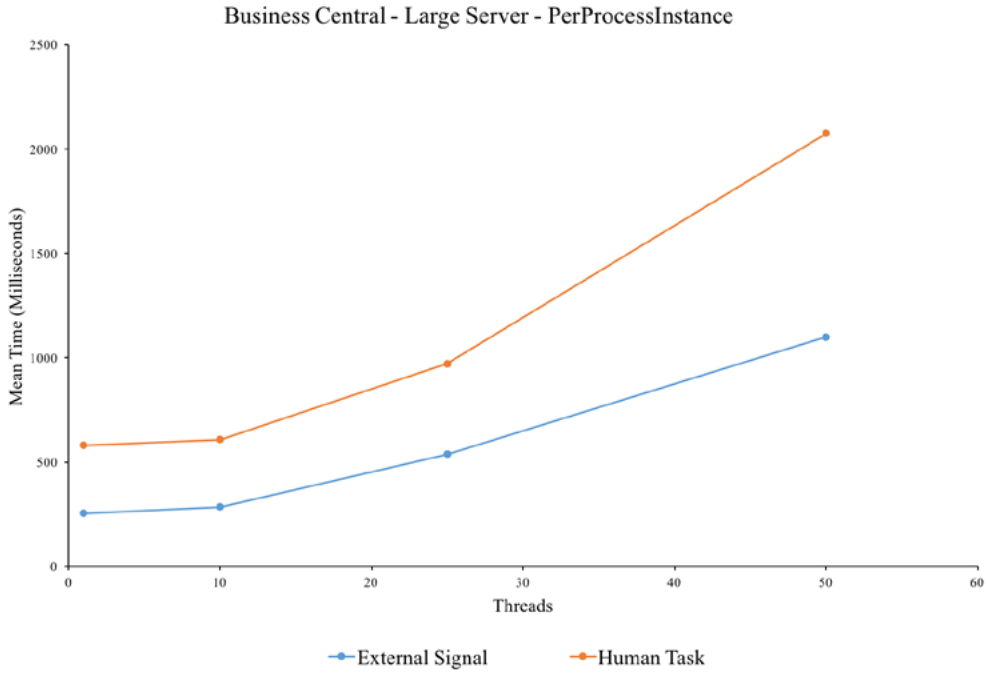


FIGURE 28. AVERAGE TIME TO COMPLETE "COMPLEX" OPERATION



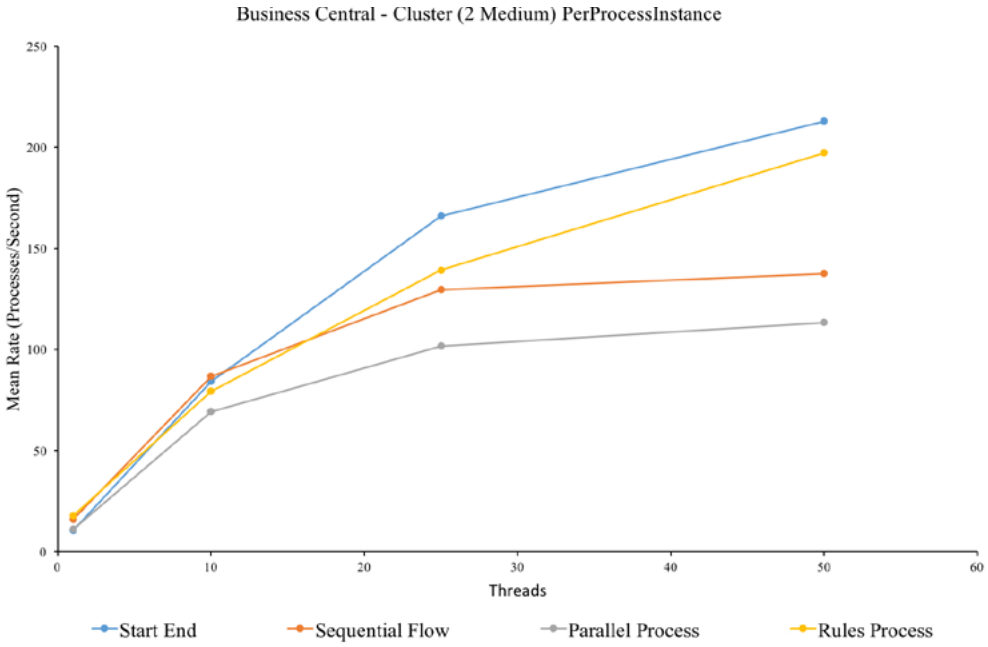
CLUSTERED BUSINESS CENTRAL SERVER TEST RESULTS

The clustered test scenario analyzes the performance of the BPMS Business Central in a clustered configuration. The clustered configuration consists of two medium sized Business Central Servers fronted by a HAProxy load balancer.

© Vizuri 2016. All rights reserved.

DATABASE SERVER CONFIGURATION	
Server Type	db.m4.2xlarge
Database	PostgreSQL 9.4.7
CPU	8 vCPU
Memory	32 GB
Storage	20 GB
TEST SERVER CONFIGURATION	
Server Type	c4.xlarge
Operating System	Red Hat Enterprise Linux 7.2
CPU	4 vCPU
Memory	7.5 GB
LOAD BALANCER CONFIGURATION	
Server Type	m4.large
Operating System	Red Hat Enterprise Linux 7.2
CPU	2 vCPU
Memory	8 GB
Server Type	HAProxy
BUSINESS CENTRAL SERVER CONFIGURATION (2)	
Server Type	m4.large
Operating System	Red Hat Enterprise Linux 7.2
CPU	2 vCPU
Memory	8 GB
BUSINESS CENTRAL JAVA VIRTUAL MACHINE (JVM) CONFIGURATION	
Version	OpenJDK 1.8.0_91
Heap	6 GB Min/Max
Database Connection Pool	20 Min 80 Max Connections
BPMS Runtime Manager	PerProcessInstance

FIGURE 29. PROCESS INSTANCES COMPLETED PER SECOND



© Vizuri 2016. All rights reserved.

FIGURE 30. AVERAGE TIME TO COMPLETE A PROCESS INSTANCE

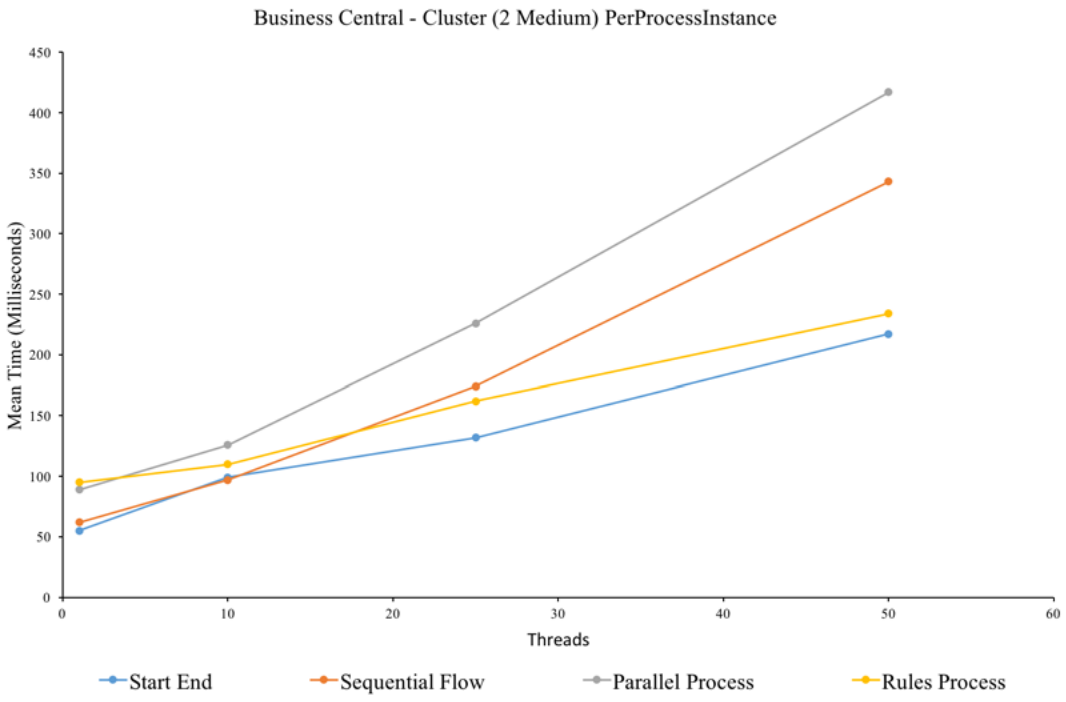
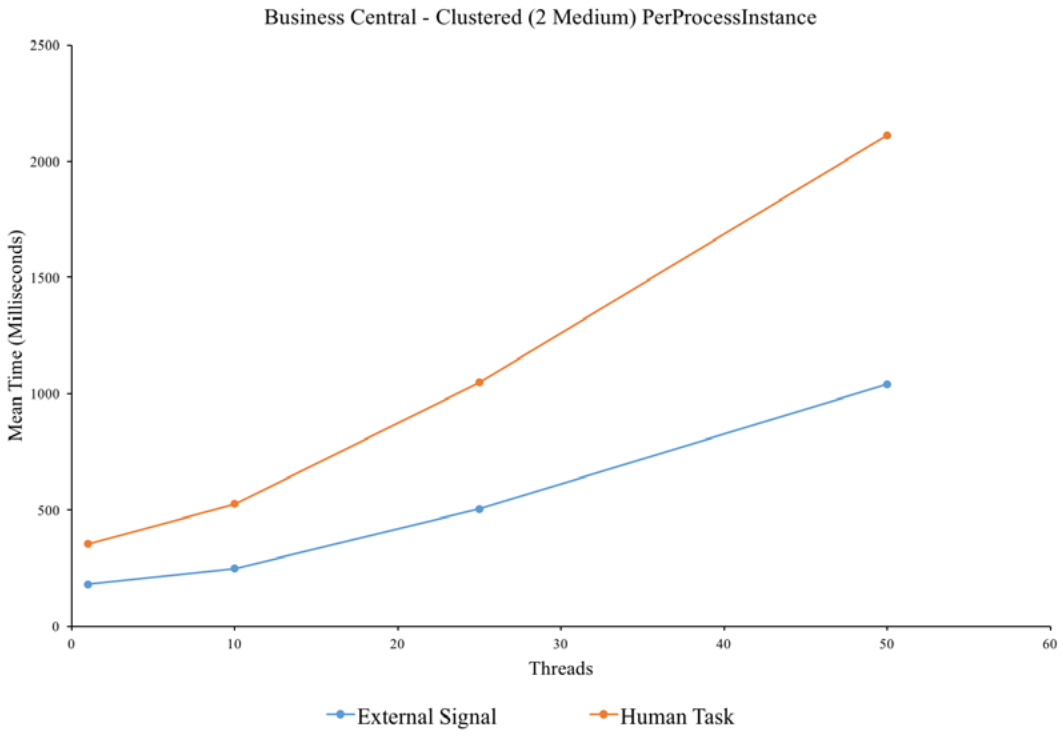


FIGURE 31. AVERAGE TIME TO COMPLETE A "COMPLEX" OPERATION

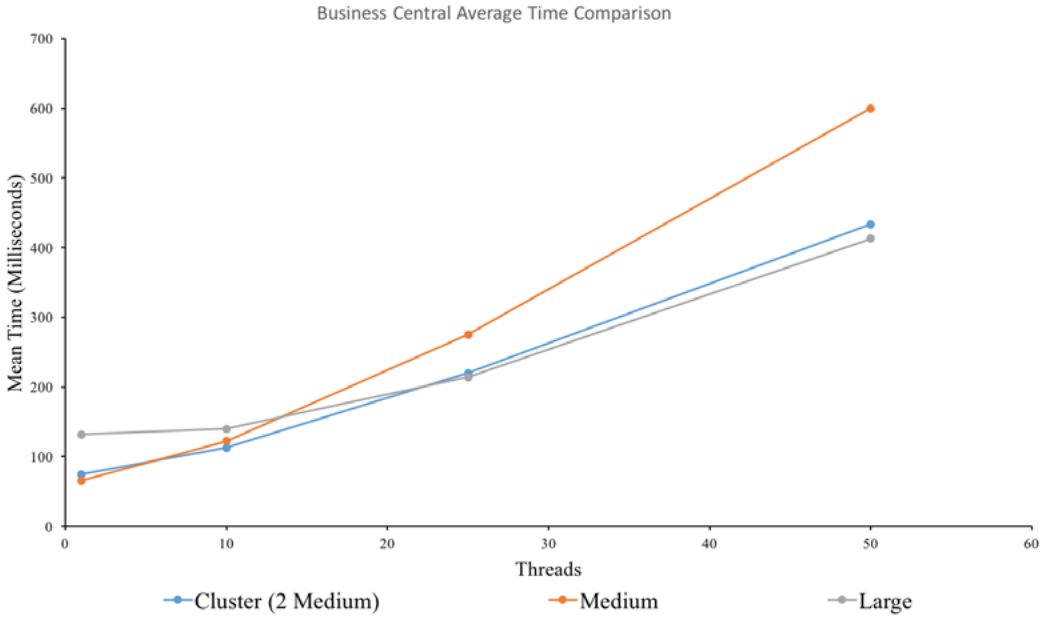


© Vizuri 2016. All rights reserved.

BUSINESS CENTRAL TEST SUMMARY

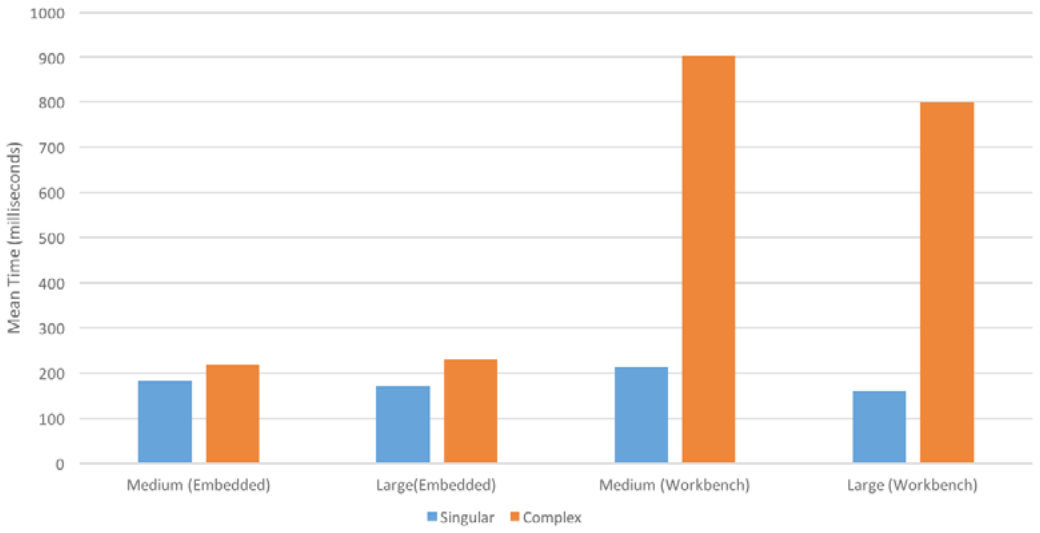
When comparing the Business Central test results, it interesting to note that the results show that the number of CPU's available for processing is the biggest factor in throughput when all else is equal as the one (1) 4-core machine vs. a cluster of two (2) 2-core machines achieves essentially the same throughput.

FIGURE 32. BUSINESS CENTRAL TEST COMPARISON



And finally, the average time to complete a process instance on a medium workbench server configuration is 270 milliseconds. The average time for the large and clustered configurations are relatively the same at 210-220 milliseconds. The following figure shows how these grand averages compare with the Embedded large and medium instances, all running with persistence with auditing enabled.

FIGURE 33. EMBEDDED VS. WORKBENCH COMPARISON



© Vizuri 2016. All rights reserved.

ABOUT VIZURI

Vizuri is an open source consulting group founded in 2002 that has had a long standing affiliation with JBoss that predates its acquisition by Red Hat. Vizuri has been a premier partner of Red Hat since 2005, and continues to deliver innovative solutions that often uti-lize their supported software offerings.

Vizuri provides expert consulting in 3 areas (Business Rules, Enterprise Integration and Cloud Enablement), and offers services that range from health check assessments through turn-key development projects and platform configurations.

ABOUT THE AUTHORS

Kent Eudy is Vizuri's Technical Director responsible for overseeing our Service Oriented Architecture (SOA), Enterprise Integration and Java Enterprise Edition (JEE) practice areas. He has over 27 years of experience including 16 as a consultant. He has led development efforts on these technologies for clients in various industry sectors such as retail, medical, government, and insurance.

Ken Spokas is Vizuri's Technical Director responsible for overseeing our Business Rules and Processes practice area. He has over 17 years of experience leading successful projects of various sizes and scopes. He particularly enjoys leveraging open-source frameworks to solve complex problems.

They both have deep experience delivering solutions utilizing the Red Hat JBoss BPM Suite and its underlying framework jBPM, both the current version and several of it's earlier incarnations.

© Vizuri 2016. All rights reserved.