ThingWorx Core 7.x Sizing Guide

Version 1.0

PTC Inc., 140 Kendrick Street, Needham, MA 02494 USA

Document Revision History

Revision Date	Version	Description of Change
December 2016	1.0	Initial document version.

ThingWorx Core 7.x Sizing Guide

Introduction

The intent of this guide is to provide the reader with a useful method to estimate the amount of processing and memory that ThingWorx may need to meet your requirements. Considerations for both on premise and cloud deployments are provided.

The guidance provided in this document is from the analysis of test data. Many performance load test scenarios were executed against various sized ThingWorx systems. Analysis of the results from these tests were used to develop the small, medium, and large thresholds discussed below, as well as other tips and guidance.

NOTES:

- **This guide should not be used as a benchmark that defines ThingWorx scalability limits. It is intended to help size the hardware for ThingWorx instances.**
- **The guidance provided in this document is intended to support a majority of sizing requests. The guidance numbers used to designate small, medium, or large should not be construed as the ceiling for that system. If your requirements exceed the guidance discussed in this document, please contact PTC to review your use case.**
- **This Sizing guide does not discuss scaling options for ThingWorx, such as sharding and federation. For scaling options, please review the ThingWorx Reference Architecture guide.**

ThingWorx Hardware Sizing Steps

In general, hardware sizing is driven by the number of things to be managed, their data streaming frequency, and concurrent user access. Additional sizing considerations are also provided, which depend on your specific use of ThingWorx.

The basic steps for hardware sizing are listed below and are described in further detail in this guide.

1. Collect ThingWorx usage requirements
2. Calculate key sizing criteria
3. Compare sizing parameter value to guidance
4. Select hardware sizing for on premise or cloud deployments
5. Consider additional load impacts. These impacts tend to be specific to your deployment

The Sizing Calculator walks through these steps in an effort to quickly and efficiently provide a ThingWorx server sizing estimate.

1. Collect ThingWorx Usage Requirements

There are three areas of requirements to collect; user access, data ingestion, and architecture/deployment.

User Access Requirements

- **Number of named users (U)**: The number of known users that will access ThingWorx.
- **User Concurrency (C):** A percentage estimate on how many users (U) will login and use ThingWorx at the same time.

Data Ingestion Requirements

For each thing type within a ThingWorx system, estimate the number of things and the data ingestion rates.

- **Number of Things (T)**: The number of things (or devices, sensors, connections, modules, etc.) that will be managed by ThingWorx. The number of things can affect many components of ThingWorx, such as the number of connection servers and platform memory requirements. The number of simultaneous things connecting to ThingWorx will directly affect the number of connection servers required.
- **Properties per Thing (P)**: The number of properties (or attributes) that each thing will send to ThingWorx. Values of these properties are sent from the thing to ThingWorx at a regular frequency. The number of persisted and logged properties will affect write performance to the database.
- **Transmission Frequency (F_D)**: The frequency of writes from each thing to ThingWorx. How often will things submit property values to ThingWorx? This can range from once a week to once per second. The number of properties and their write frequency is the major influence on platform size. It is the largest factor in deciding what database solution is needed by ThingWorx to ingest the content.

Common Transmission Rates	Equivalent Daily Transmission Frequency (F_D)
once per day	1
once per hour	24
every 15 minutes	96
every 5 minutes	288
every minute	1440
every 30 seconds	2880
every second (or 1Hz)	86400

Architecture/Deployment Requirements

These architecture/deployment requirements may be prudent to consider, depending on your use of them in ThingWorx.

- File transfers from things to/from ThingWorx, including software updates.
- Subscriptions, timers, and events generated by your use of ThingWorx.
- The number of normal concurrent tunnel sessions to devices.
- Connections to other services (such as a SCADA, ERP, and other back office systems).
- Customer data retention policies, especially if that retained content is frequently accessed by ThingWorx.

2. Calculate Key Sizing Criteria

After the requirements are collected, use them as inputs to calculate your key sizing criteria. The calculations mainly shape requirements into common units with PTC's sizing guidelines.

User Concurrency

For ThingWorx sizing, user access is recognized as the number of concurrent user-driven HTTP requests. We estimate the number of concurrent HTTP requests through the number of known users (U) and the expected rate of concurrent access (C).

HTTP Requests = (U) x (C)

Data Ingestion

For each thing type, calculate the following:

- Number of Things (T) - This value comes directly from the requirements.
- Properties per Thing (P) - This value comes directory from the requirements.
- Transmission frequency (F_S) - Convert the given daily frequency (F_D) to a per second rate.

  F_S = (F_D) x (1 day / 24 hours) x (1 hour / 60 minutes) x (1 minute / 60 seconds)

- Operations per second (OPS) - The total number of operations that ThingWorx could receive in one second.

  OPS = (T) x (Fs)

- Property writes per second (PWS) - The total number of property writes that ThingWorx may need to write to its database(s).

  PWS = (P) x (OPS)

Take a full count of things and property writes from all thing types that ThingWorx will manage. **Thingcount** is the total number of things to be managed, and **VS queue** is the number of writes coming from the devices to ThingWorx.

Thingcount = ΣT_{types}
VS queue = ΣPWS_{types}

Also for data ingestion, calculate the number of connection servers (CS) needed to support the connections between devices and ThingWorx. A general rule-of-thumb is that a connection server is needed for every 50,000 things.

CS = (Thingcount) / 50,000

Your Key Sizing Criteria

Key Sizing Parameter	Definition	Your Value
CS	Number of Connection servers to be added	
HTTP Requests	Estimated concurrent user requests to ThingWorx	
Thingcount	Estimated number of Things managed by ThingWorx	
VS Queue	Estimated writes per second to ThingWorx	

3. Compare Sizing Criteria to Guidelines

Compare your sizing criteria against the following guidelines to select an appropriate size rating of small, medium, or large.

ValueStream (VS) Queue Rate

VS Queue is the amount of data that ThingWorx receives and manages from all devices. The max write/sec value here is compared to your VS queue criteria. Choose a size where your VS queue criteria is lower than the Max writes/second.

Platform	Max Writes/second (wps)	Max Writes/hour (wph)
ThingWorx/H2 – Extra Small	4,000	14.4 million
ThingWorx /H2 - Small	6,000	21.6 million
ThingWorx /PostgreSQL - Small	6,000	21.6 million
ThingWorx /PostgreSQL - Medium	9,000	32.4 million
ThingWorx /PostgreSQL - Large	15,000	54.0 million
ThingWorx Enterprise (refer to Getting Started with Datastax Enterprise and ThingWorx)	greater than 15,000	Greater than 54.0 million

Compare your ValueStream Queue rate against these guidelines to select an appropriately sized system.

Thing Count Comparison

The number of devices connected to ThingWorx. The number of things managed by ThingWorx has its greatest influence on the memory requirements of the platform and has little bearing on CPU utilization. The general guidelines to follow are:

Platform	No. of Devices (or things)

ThingWorx – Extra Small	10,000
Thingworx - Small	30,000
Thingworx - Medium	100,000
Thingworx - Large	250,000
Contact PTC	Greater than 250,000

Compare your Thing count against these guidelines to select an appropriately sized system.

HTTP Requests Comparison

The number of concurrent user requests to be managed. The number of things managed by ThingWorx has its greatest influence on the memory requirements of the platform and has little bearing on CPU utilization. The general guidelines to follow are:

Platform	Max HTTP Requests
ThingWorx – Extra Small	400
ThingWorx - Small	400
ThingWorx - Medium	950
ThingWorx - Large	1,500

Compare your HTTP requests to these guidelines to select an appropriately sized system.

Connection Servers Estimate

Round down the CS value to get a value to the next whole number. For example, 2.3 rounds down to 2. This value will be the suggested number of connections servers to be used in your ThingWorx system.

For each Connection Server, we recommend the same size server that is recommended for a small Thingworx platform.

4. Select Hardware Sizing for On Premise or Cloud Deployments

Now compare the thing count, value stream, and http request evaluation sizes. The largest size from any of these evaluations should be applied as the overall platform size. In the majority of reviews, PTC expects the Value Stream Queue rate to be the largest factor in determining platform size. The following charts provide comparable AWS, Microsoft Azure, and on premise specification for small, medium, and large sized ThingWorx platforms and databases. With the size now determined, use the charts below to obtain server size metrics.

ThingWorx

Size	AWS EC2	Azure VM	On-premise CPU Cores	On premise Memory (GiB)	Storage Bandwidth (Mbps)
Extra Small/H2	C4.large	F4	4	7.5	750
Small/H2	C4.2xlarge	F8	8	15	1,000
Small	C4.2xlarge	F8	8	15	1,000
Medium	C4.4xlarge	F16	16	30	2,000

Large	C4.8xlarge		36	60	4,000

PostgreSQL Database

Size	AWS EC2	Azure VM	On-premise CPU Cores	On premise Memory (GiB)	SSD Storage (MB)
Small	C3.2xlarge	F8	8	15	2 x 80
Medium	C3.4xlarge	F16	16	30	2 x 160
Large	C3.8xlarge		32	60	2 x 320

Server Terminology

The following content discusses the hardware terminology used in the above charts.

Traditional on Premise Terminology

Traditional or on premise hardware sizes are typically discussed in terms of **CPU cores for** processing power and **RAM** for memory capability. For example, a small ThingWorx platform using the H2 database may be sized at 8 CPU cores and 15 GB RAM.

Amazon Web Services (AWS) Terminology

For EC2 instances, AWS provides a wide selection of instance types to fit your use cases. PTC Performance testing is done using the Compute Optimized instance types, primarily C4 and C3. They are defined here by AWS as "...instances are the latest generation of compute-optimized instances, featuring the highest performing processors and the lowest price/compute performance in EC2."

AWS provides a T-shirt methodology for selecting the size of an EC2 instance in terms of CPU and memory. Typical sizing terms are large, xlarge, 2xlarge, etc. Following the example in the above on premise terminology, a small ThingWorx platform using the H2 database may be sized to run on a C4.2xlarge EC2 instance. Other EC2 instance types, such as General Purpose (M) or Memory Intensive (R), can also be considered, but are not covered in this guide. Please contact PTC to discuss these options.

Microsoft Azure

Azure provides a selection of instance types to fit your use cases. PTC tends to recommend the Compute Optimized instance types, primarily the F series. They are defined here by MS Azure as VMs that "...sport a higher CPU to memory ratio. They feature 2 GB RAM and 16 GB of local solid state drive (SSD) per CPU core, and are optimized for compute intensive workloads."

Azure provides a packaged method for selecting a VM in terms of CPU cores. Typical sizing terms are F2, F4, F8, etc. where the number represents the number of CPU cores in the VM. Following the example in the above on premise terminology, a small ThingWorx platform using the H2 database may be sized to run on a F8 VM.

5. Additional Platform Loads to Consider

The added load from other solution, deployment, and other architecture requirements may be prudent to consider, depending on your use of them. Below are some common architecture decisions and operations that may affect hardware sizing for ThingWorx.

High Availability Requirements for ThingWorx

Most high availability requirements will push a ThingWorx system to incorporate a high availability architecture such as that described in the Thingworx HA guide. The added components of a PostgreSQL High Availability system and added processing (load balancing, replication, etc) can cause a slight reduction in write performance, enough to require consideration when sizing a ThingWorx system.

File Vaulting/Management

Will any file content (images, pdf files, etc.) be transferred from the things? In most Remote Service business scenarios, File Upload (from device to the platform) is basic a requirement. These files contain anything from log files to images generated by the device (needed in the platform for troubleshooting) or other calibration data. Also, files pushed or downloaded (from the platform to the device) is another common use case, such as pushing calibration data, software updates etc.

Subscriptions and Events

Subscriptions, timers, and events can add load, but this is very specific to your implementation and are out of scope for general coverage provided in this document.

Database Choices

A database choice may have been derived previously from high availability requirements, customer comfort and experience, etc. But the database choice does have a role in ThingWorx server sizing as well.

- H2 is an out-of-the-box database supplied as part of ThingWorx. PTC provides it as a useful database for development and small production systems. But its use does not scale well past small implementations.
- PostgreSQL is the database compatible with ThingWorx that will scale for all small, medium, and large implementations. The ThingWorx/PostgreSQL combination is the only system that can meet high availability requirements. Refer to the ThingWorx HA Guide for HA-based architecture and deployment documentation.
- PostgreSQL has write limits. With ThingWorx, we've measured it at 15,000 writes/sec. Higher write frequency criteria would lead to the addition of a Datastax Enterprise (DSE) solution. Refer to Getting Started with Datastax Enterprise and ThingWorx.

Platform Sizing Examples

Here are a few examples that walk through this sizing process.

Example #1: Smart City Monitoring (large number of things, small number of properties, low write frequency)

Scenario

Monitoring the 100,000 water meters throughout the city. Each water meter reports 20 property values to ThingWorx every five minutes. There are up to 500 known users that will access the ThingWorx system, expecting up to 20% to be on simultaneously at times.

Requirements

- Number of Thing Types: 1
- Number of things: 100,000
- Number of properties: 20
- Write Frequency: 288 writes per day, per property
- Number of users: 500
- User concurrency: 20%

Calculations

- Number of Things (T) = 100,000
- **Thingcount = 100,000**
- CS = 100,000 / 50,000
- **CS = 2**
- Number of Properties per Thing (P) = 20
- Transmission Frequency (F_S) = (288 writes/day) (1 day / 24 hours) (1 hour / 60 minutes) (1 minute / 60 seconds) = 0.0033 write/sec
- Operations per Second (OPS) = (100,000 things) (0.003 write/sec) = 330 ops
- Property Writes per Second (PWS) = (330 ops) (20 properties) = 6,600 wps
- **VS Queue rate = 6,000 wps**
- User concurrency = (500)(0.20) = 100 users
- **HTTP Requests = 100**

Criteria Comparison

- Thingcount = 100,000. This estimate is larger than a small thingcount of 30,000, and equal a medium thingcount of 100,000. A medium ThingWorx platform is sufficient.
- CS = 2. Two connection servers are recommended.

- VS Queue rate = 6,600. This estimate is larger than the small queue rate of 6,000 wps, but smaller than the medium VS queue rate of 9,000 wps. A medium ThingWorx platform (with PostgreSQL) is sufficient.
- HTTP Request = 100. A small ThingWorx platform is sufficient.

Sizing

Reviewing all estimates, a medium ThingWorx system will satisfy all criteria. Reviewing the above charts, a medium system is:

ThingWorx

Size	AWS EC2 Instance	Azure VM	On Premise CPU Cores	On Premise Memory (GiB)
Medium	C4.4xlarge	F16	16	30

PostgreSQL Database

Size	AWS EC2 Instance	Azure VM	On Premise CPU Cores	On Premise Memory (GiB)
Medium	C3.4xlarge	F16	16	30

ThingWorx Connection Server

Size	AWS EC2 Instance	Azure VM	On Premise CPU Cores	On Premise Memory (GiB)	Quantity
Medium	C4.2xlarge	F8	8	15	2

Example #2: (small number of things, small number of properties, high write frequency)

Scenario

A medium-sized factory where 250 machines are monitored. Each monitored machine is sending 50 property results to ThingWorx every second. There are 100 users that access ThingWorx, with 25 of them recognized as power users.

Requirements

- Number of Thing Types: 1
- Number of things: 250
- Number of properties: 50
- Write Frequency: 86,400 writes per day, per property
- Number of users: 100
- User concurrency: 25%

Calculations

- Number of Things (T) = 250
- **Thingcount = 250**
- CS = round down (250 / 50,000)
- **CS = 0**
- Number of Properties per Thing (P) = 50
- Transmission Frequency (F_S) = (86,400 writes/day)(1 day / 24 hours)(1 hour / 60 minutes)(1 minute / 60 seconds) = 1 write/sec
- Operations per Second (OPS) = (250 things)(1 write/sec) = 250 ops
- Property Writes per Second (PWS) = (250 ops)(50 properties) = 12,500 wps
- **VS Queue rate = 12,500 wps**
- User concurrency = (100)(0.25) = 25 users
- **HTTP Requests = 25**

Criteria Comparison

- Thingcount = 250. This estimate is smaller than a small thingcount of 30,000. A small ThingWorx platform is sufficient.
- VS Queue rate = 12,500. This estimate is larger than the small queue rate of 6,000 wps and larger than the medium queue rate of 9,000 wps. It is smaller than the large queue rate of 15,000. A large ThingWorx platform (with PostgreSQL) is sufficient.
- HTTP Request = 25. A small ThingWorx platform is sufficient.
- CS = 0. No connection servers are necessary.

Sizing

Comparing all criteria, a large ThingWorx system will satisfy all criteria. Reviewing the above charts, a large system is:

ThingWorx

Size	AWS EC2 Instance	On Premise CPU Cores	On Premise Memory (GiB)
Large	C4.8xlarge	32	60

PostgreSQL Database

Size	AWS EC2 Instance	On Premise CPU Cores	On Premise Memory (GiB)

Large	C3.8xlarge	32	60

Example #3: Remote Monitoring Service (small number of things, large number of properties, low write frequency)

Scenario

Monitoring service where 25,000 devices are remotely monitored. Each monitored device is sending 250 property results to ThingWorx every 10 minutes. There are 100 users that access ThingWorx, with an expected concurrency of 25%.

Requirements

- Number of Thing Types: 1
- Number of things: 25,000
- Number of properties: 250
- Write Frequency: Every 10 minutes (144 writes per day)
- Number of users: 100
- User concurrency: 25%

Calculations

- Number of Things (T) = 25,000
- **Thingcount = 25,000**
- CS = rounddown (25,000 / 50,000)
- **CS = 0**
- Number of Properties per Thing (P) = 250
- Transmission Frequency (FS) = (144 writes/day)(1 day / 24 hours)(1 hour / 60 minutes)(1 minute / 60 seconds) = 0.0017 write/sec
- Operations per Second (OPS) = (25,000 things)(0.0017 write/sec) = 43 ops
- Property Writes per Second (PWS) = (43 ops)(250 properties) = 10,625 wps
- **VS Queue rate = 10,625 wps**
- User concurrency = (100)(0.25) = 25 users
- **HTTP Requests = 25**

Criteria Comparison

- Thingcount = 25,000 < 30,000. A small Thingworx platform is sufficient
- VS Queue rate = 10,625. This estimate is larger than the small queue rate of 6,000 wps and larger than the medium queue rate of 9,000 wps. It is smaller than the large queue rate of 15,000. A large Thingworx platform (with PostgreSQL) is sufficient.
- HTTP Request = 25. A small Thingworx platform is sufficient.
- CS = 0. No connection servers are necessary.

Sizing

Comparing all criteria, a large ThingWorx system will satisfy all criteria. Reviewing the above charts, a large system is:

ThingWorx

Size	AWS EC2 Instance	On Premise CPU Cores	On Premise Memory (GiB)
Large	C4.8xlarge	32	60

PostgreSQL Database

Size	AWS EC2 Instance	On Premise CPU Cores	On Premise Memory (GiB)
Large	C3.8xlarge	32	60

Appendix: PTC Test Run Summaries

The following are descriptions of the test runs used to develop the guidelines for this guide. For configuration and tuning, the ThingWorx Install guide was followed. No other configuration or tuning actions were applied.

Extra Small Server (using H2)

A test of ThingWorx performance with H2 using 4 CPU cores and 7.5 GB RAM.

Hardware Configuration

AWS EC2 Instance Type	C3.xlarge
vCPU	4
Memory	7.5 GB
Storage	80 GB (SSD)

Test Scenario

Basic Configuration	
Number of things	10,000

Number of templates	40				
Number of Properties	20				
Property types	Integer	String			
Number of Services	20				
Properties with Alerts	50%				
Alerts with Subscriptions	50%				
Things with…	**Percent**	**Number of Thing**	**Number of Properties**		
Simple Properties	20%	2,000	40,000		
Logged Properties	68%	6,800	136,000		
Persistent Properties	2%	200	4,000		
Read Only Properties	10%	1,000	20,000		
Total number of things	100%	10,000	200,000		
Write Operations	**Percent**	**Number of Things**			
Chatty	20%	1,800			
Non-Chatty	80%	7,200			
Configuration for Streams					
Number of Streams	20				
Number of Data Shapes	2				
Property Types per Template	Integer, String				
Number of Columns	10				
Data Tables					
Table Type	**Number Tables**	**Data Shapes**	**Initial Rows**	**Property Types**	**Fields/Type**
Large Tables	10	2	1,000	Integer, String	10
Lookup Tables	25	2	10	String	1
Configuration for External Subscriptions					
Number of Alerting Things	20				
Number of Subscriptions	2				
Mashups / Read					

Operation	Total Users	Max Items
Mashup (property, value stream)	500	100
Mashup (stream)	250	100
Mashup (data tables)	500	100
Users		
Administrators	100	
Non-Administrators	1,000	

Test Results

Test Results Summary	
CPU Utilization	41.5% of 4 CPU cores
Memory Utilization	4.4 of 7.3 GB (60.3%)
Websocket Requests (writes)	851 wps
HTTP Requests (reads)	61 rps
Value Stream Queue Rate	6,000 wps
Stream Queue Rate	16 wps
Alerts Queue Rate	155 ops
Events Queue Rate	1 ops

Platform Subsystem- GetPerformanceMetrics		
Name	**Description**	**Value**
eventQueueSize	Event queue size	0
streamQueueSize-ThingworxPersistenceProvider	Stream queue size	0
valueStreamQueueSize-ThingworxPersistenceProvider	Value Stream queue size	0
memoryInUse	Memory in use (bytes)	3,873,615,040
totalMemoryAllocated	Total memory allocated (bytes)	6,847,201,280

thingCount	Thing count	30,169

Value Stream Subsystem - GetPerformanceMetrics		
Name	**Description**	**Value**
maximumWaitTime	ThingworxPersistenceProvider: Maximum wait time before flushing stream buffer (milliseconds)	10,000
sizeThreshold	ThingworxPersistenceProvider: Maximum number of items accumulated before flushing stream buffer	1,000
maximumBlockSize	ThingworxPersistenceProvider: Maximum number of stream writes processed in one block	2,500
scanRate	ThingworxPersistenceProvider: Rate stream queue is checked (milliseconds)	5
maximumQueueSize	ThingworxPersistenceProvider: Maximum number of stream entries to queue	4,000,000
queueSize	ThingworxPersistenceProvider: Number of stream entries currently queued	0
totalWritesQueued	ThingworxPersistenceProvider: Number of stream entries that have been queued	7,373,641
totalWritesPerformed	ThingworxPersistenceProvider: Number of stream entries that have been performed	5,848,072
numberOfProcessingThreads	ThingworxPersistenceProvider: Number of processing threads	15

Stream Subsystem - GetPerformanceMetrics		
Name	**Description**	**Value**
maximumWaitTime	ThingworxPersistenceProvider: Maximum wait time before flushing stream buffer (milliseconds)	10,000

sizeThreshold	ThingworxPersistenceProvider: Maximum number of items accumulated before flushing stream buffer	1,000
maximumBlockSize	ThingworxPersistenceProvider: Maximum number of stream writes processed in one block	2,500
scanRate	ThingworxPersistenceProvider: Rate stream queue is checked (milliseconds)	5
maximumQueueSize	ThingworxPersistenceProvider: Maximum number of stream entries to queue	2,000,000
queueSize	ThingworxPersistenceProvider: Number of stream entries currently queued	0
totalWritesQueued	ThingworxPersistenceProvider: Number of stream entries that have been queued	8,260
totalWritesPerformed	ThingworxPersistenceProvider: Number of stream entries that have been performed	8,260
numberOfProcessingThreads	ThingworxPersistenceProvider: Number of processing threads	15

Small Servers Test (using PostgreSQL)

A test of ThingWorx performance with PostgreSQL using two servers with 8 CPU cores and 15 GB RAM.

Hardware Configuration

Server Purpose	ThingWorx	PostgreSQL
AWS EC2 Instance Type	C4.2xlarge	C3.2xlarge
vCPU	8	8
Memory	15 GB	15 GB
Storage	1,000 Mbps	160 GB (SSD)

Test Scenario

Basic Configuration	
Number of things	30,000

Number of templates	40				
Number of Properties	20				
Property types	Integer	String			
Number of Services	20				
Properties with Alerts	50%				
Alerts with Subscriptions	50%				
Things with…	**Percent**	**Number of Thing**	**Number of Properties**		
Simple Properties	20%	6,000	120,000		
Logged Properties	68%	20,400	408,000		
Persistent Properties	2%	600	12,000		
Read Only Properties	10%	3,000	60,000		
Total number of things	100%	30,000	600,000		
Write Operations	**Percent**	**Number of Things**			
Chatty	20%	5,400			
Non-Chatty	80%	21,600			
Configuration for Streams					
Number of Streams	20				
Number of Data Shapes	2				
Property Types per Template	Integer, String				
Number of Columns	10				
Data Tables					
Table Type	**Number Tables**	**Data Shapes**	**Initial Rows**	**Property Types**	**Fields/Type**
Large Tables	10	2	1,000	Integer, String	10
Lookup Tables	25	2	10	String	1
Configuration for External Subscriptions					
Number of Alerting Things	20				
Number of Subscriptions	2				
Mashups / Read					

Operation	Total Users	Max Items
Mashup (property, value stream)	500	100
Mashup (stream)	250	100
Mashup (data tables)	500	100
Users		
Administrators	100	
Non-Administrators	1,000	

Test Results

Test Results Summary		
Server	**ThingWorx**	**PostgreSQL**
CPU Utilization	59.8% of 8 CPU cores	51.4% of 8 CPU cores
Memory Utilization	3.0 of 14.7 GB (20.1%)	1.3 of 14.7 GB (8.8%)
Websocket Requests (writes)	905 wps	
HTTP Requests (reads)	648 rps	
Value Stream Queue Rate	4,000 wps	
Stream Queue Rate	4 wps	
Alerts Queue Rate	0 ops	
Events Queue Rate	0 ops	

Platform Subsystem- GetPerformanceMetrics		
Name	**Description**	**Value**
eventQueueSize	Event queue size	0
streamQueueSize-ThingworxPersistenceProvider	Stream queue size	0
valueStreamQueueSize-ThingworxPersistenceProvider	Value Stream queue size	0
memoryInUse	Memory in use (bytes)	1,774,655,624
totalMemoryAllocated	Total memory allocated (bytes)	2,147,483,648
thingCount	Thing count	30,171

Value Stream Subsystem - GetPerformanceMetrics		
Name	**Description**	**Value**
maximumWaitTime	ThingworxPersistenceProvider: Maximum wait time before flushing stream buffer (milliseconds)	10,000
sizeThreshold	ThingworxPersistenceProvider: Maximum number of items accumulated before flushing stream buffer	1,000
maximumBlockSize	ThingworxPersistenceProvider: Maximum number of stream writes processed in one block	2,500
scanRate	ThingworxPersistenceProvider: Rate stream queue is checked (milliseconds)	5
maximumQueueSize	ThingworxPersistenceProvider: Maximum number of stream entries to queue	4,000,000
queueSize	ThingworxPersistenceProvider: Number of stream entries currently queued	0
totalWritesQueued	ThingworxPersistenceProvider: Number of stream entries that have been queued	7,370,888
totalWritesPerformed	ThingworxPersistenceProvider: Number of stream entries that have been performed	7,370,888
numberOfProcessingThreads	ThingworxPersistenceProvider: Number of processing threads	50

Stream Subsystem - GetPerformanceMetrics		
Name	**Description**	**Value**
maximumWaitTime	ThingworxPersistenceProvider: Maximum wait time before flushing stream buffer (milliseconds)	10,000
sizeThreshold	ThingworxPersistenceProvider: Maximum number of items accumulated before flushing stream buffer	1,000
maximumBlockSize	ThingworxPersistenceProvider: Maximum number of stream writes processed in one block	2,500
scanRate	ThingworxPersistenceProvider: Rate stream queue is checked (milliseconds)	5
maximumQueueSize	ThingworxPersistenceProvider: Maximum number of stream entries to queue	2,000,000
queueSize	ThingworxPersistenceProvider: Number of stream entries currently queued	0
totalWritesQueued	ThingworxPersistenceProvider: Number of stream entries that have been queued	8,260
totalWritesPerformed	ThingworxPersistenceProvider: Number of stream entries that have been performed	8,260
numberOfProcessingThreads	ThingworxPersistenceProvider: Number of processing threads	50

Medium Servers Test (using PostgreSQL)

A test of ThingWorx performance with PostgreSQL using two servers with 16 CPU cores and 30 GB RAM.

Hardware Configuration

Server Purpose	ThingWorx	PostgreSQL
AWS EC2 Instance Type	C4.4xlarge	C3.4xlarge
vCPU	16	16
Memory	30 GB	30 GB
Storage	2,000 Mbps	320 GB (SSD)

Test Scenario

Basic Configuration			
Number of things	100,000		
Number of templates	40		
Number of Properties	20		
Property types	Integer	String	
Number of Services	20		
Properties with Alerts	50%		
Alerts with Subscriptions	50%		
Things with…	**Percent**	**Number of Thing**	**Number of Properties**
Simple Properties	20%	20,000	400,000
Logged Properties	68%	68,000	1,360,000
Persistent Properties	2%	2,000	40,000
Read Only Properties	10%	10,000	200,000

Total number of things	100%	100,000	2,000,000		
Write Operations	**Percent**	**Number of Things**			
Chatty	20%	18,000			
Non-Chatty	80%	72,000			
Configuration for Streams					
Number of Streams	20				
Number of Data Shapes	2				
Property Types per Template	Integer, String				
Number of Columns	10				
Data Tables					
Table Type	**Number Tables**	**Data Shapes**	**Initial Rows**	**Property Types**	**Fields/Type**
Large Tables	10	2	1,000	Integer, String	10
Lookup Tables	25	2	10	String	1
Configuration for External Subscriptions					
Number of Alerting Things	20				
Number of Subscriptions	2				
Mashups / Read					
Operation		**Total Users**		**Max Items**	
Mashup (property, value stream)		500		100	
Mashup (stream)		250		100	
Mashup (data tables)		500		100	

Users	
Administrators	100
Non-Administrators	1,000

Test Results

Test Results Summary		
Server	ThingWorx	PostgreSQL
CPU Utilization	13.7% of 16 CPU cores	81.3% of 16 CPU cores
Memory Utilization	13.5 of 29.4 GB (45.9%)	23.2 of 29.4 GB (78.9%)
Websocket Requests (writes)	4,090 wps	
HTTP Requests (reads)	137 rps	
Value Stream Queue Rate	9,000 wps	
Stream Queue Rate	16 wps	
Alerts Queue Rate	392 ops	
Events Queue Rate	1 ops	

Platform Subsystem- GetPerformanceMetrics		
Name	**Description**	**Value**
eventQueueSize	Event queue size	0
streamQueueSize-ThingworxPersistenceProvider	Stream queue size	0
valueStreamQueueSize-ThingworxPersistenceProvider	Value Stream queue size	0
memoryInUse	Memory in use (bytes)	5,608,167,664

totalMemoryAllocated	Total memory allocated (bytes)	11,286,872,064
thingCount	Thing count	100,334

Value Stream Subsystem - GetPerformanceMetrics		
Name	**Description**	**Value**
maximumWaitTime	ThingworxPersistenceProvider: Maximum wait time before flushing stream buffer (milliseconds)	10,000
sizeThreshold	ThingworxPersistenceProvider: Maximum number of items accumulated before flushing stream buffer	1,000
maximumBlockSize	ThingworxPersistenceProvider: Maximum number of stream writes processed in one block	2,500
scanRate	ThingworxPersistenceProvider: Rate stream queue is checked (milliseconds)	5
maximumQueueSize	ThingworxPersistenceProvider: Maximum number of stream entries to queue	4,000,000
queueSize	ThingworxPersistenceProvider: Number of stream entries currently queued	0
totalWritesQueued	ThingworxPersistenceProvider: Number of stream entries that have been queued	19,939,962
totalWritesPerformed	ThingworxPersistenceProvider: Number of stream entries that have been performed	19,939,962
numberOfProcessingThreads	ThingworxPersistenceProvider: Number of processing threads	50

Stream Subsystem - GetPerformanceMetrics		
Name	**Description**	**Value**
maximumWaitTime	ThingworxPersistenceProvider: Maximum wait time before flushing stream buffer (milliseconds)	10,000
sizeThreshold	ThingworxPersistenceProvider: Maximum number of items accumulated before flushing stream buffer	1,000
maximumBlockSize	ThingworxPersistenceProvider: Maximum number of stream writes processed in one block	2,500
scanRate	ThingworxPersistenceProvider: Rate stream queue is checked (milliseconds)	5
maximumQueueSize	ThingworxPersistenceProvider: Maximum number of stream entries to queue	2,000,000
queueSize	ThingworxPersistenceProvider: Number of stream entries currently queued	0
totalWritesQueued	ThingworxPersistenceProvider: Number of stream entries that have been queued	20,724
totalWritesPerformed	ThingworxPersistenceProvider: Number of stream entries that have been performed	20,724
numberOfProcessingThreads	ThingworxPersistenceProvider: Number of processing threads	50

Large Servers Test (using PostgreSQL)

A test of ThingWorx performance with PostgreSQL using two servers with 36 CPU cores and 60 GB RAM.

Hardware Configuration

Server Purpose	ThingWorx	PostgreSQL
AWS EC2 Instance Type	C4.8xlarge	C3.8xlarge
vCPU	36	32
Memory	60 GB	60 GB
Storage	4,000 Mbps	640 GB (SSD)

Test Scenario

Basic Configuration			
Number of things	250,000		
Number of templates	40		
Number of Properties	20		
Property types	Integer	String	
Number of Services	20		
Properties with Alerts	50%		
Alerts with Subscriptions	50%		
Things with…	**Percent**	**Number of Thing**	**Number of Properties**
Simple Properties	20%	50,000	1,000,000
Logged Properties	68%	170,000	3,400,000
Persistent Properties	2%	5,000	100,000
Read Only Properties	10%	25,000	500,000

Total number of things	100%	250,000	5,000,000		
Write Operations	**Percent**	**Number of Things**			
Chatty	20%	45,000			
Non-Chatty	80%	180,000			
Configuration for Streams					
Number of Streams	20				
Number of Data Shapes	2				
Property Types per Template	Integer, String				
Number of Columns	10				
Data Tables					
Table Type	**Number Tables**	**Data Shapes**	**Initial Rows**	**Property Types**	**Fields/Type**
Large Tables	10	2	1,000	Integer, String	10
Lookup Tables	25	2	10	String	1
Configuration for External Subscriptions					
Number of Alerting Things	20				
Number of Subscriptions	2				
Mashups / Read					
Operation		**Total Users**		**Max Items**	
Mashup (property, value stream)		1,000		100	
Mashup (stream)		500		100	
Mashup (data tables)		1,000		100	

Users	
Administrators	100
Non-Administrators	3,000

Test Results

Test Results Summary		
Server	**ThingWorx**	**PostgreSQL**
CPU Utilization	52.6% of 36 CPU cores	67.6% of 32 CPU cores
Memory Utilization	48.2 of 59.0 GB (81.6%)	4.4 of 59.0 GB (7.4%)
Websocket Requests (writes)	3,714 wps	
HTTP Requests (reads)	1,545 rps	
Value Stream Queue Rate	17,000 wps	
Stream Queue Rate	11 wps	
Alerts Queue Rate	0 ops	
Events Queue Rate	0 ops	

Platform Subsystem- GetPerformanceMetrics		
Name	**Description**	**Value**
eventQueueSize	Event queue size	0
streamQueueSize-ThingworxPersistenceProvider	Stream queue size	0
valueStreamQueueSize-ThingworxPersistenceProvider	Value Stream queue size	0
memoryInUse	Memory in use (bytes)	30,152,997,728

totalMemoryAllocated	Total memory allocated (bytes)	47,496,298,496
thingCount	Thing count	250,745

Value Stream Subsystem - GetPerformanceMetrics		
Name	**Description**	**Value**
maximumWaitTime	ThingworxPersistenceProvider: Maximum wait time before flushing stream buffer (milliseconds)	10,000
sizeThreshold	ThingworxPersistenceProvider: Maximum number of items accumulated before flushing stream buffer	1,000
maximumBlockSize	ThingworxPersistenceProvider: Maximum number of stream writes processed in one block	2,500
scanRate	ThingworxPersistenceProvider: Rate stream queue is checked (milliseconds)	5
maximumQueueSize	ThingworxPersistenceProvider: Maximum number of stream entries to queue	4,000,000
queueSize	ThingworxPersistenceProvider: Number of stream entries currently queued	0
totalWritesQueued	ThingworxPersistenceProvider: Number of stream entries that have been queued	63,135,423
totalWritesPerformed	ThingworxPersistenceProvider: Number of stream entries that have been performed	63,135,423
numberOfProcessingThreads	ThingworxPersistenceProvider: Number of processing threads	50

Stream Subsystem - GetPerformanceMetrics		
Name	**Description**	**Value**
maximumWaitTime	ThingworxPersistenceProvider: Maximum wait time before flushing stream buffer (milliseconds)	10,000
sizeThreshold	ThingworxPersistenceProvider: Maximum number of items accumulated before flushing stream buffer	1,000
maximumBlockSize	ThingworxPersistenceProvider: Maximum number of stream writes processed in one block	2,500
scanRate	ThingworxPersistenceProvider: Rate stream queue is checked (milliseconds)	5
maximumQueueSize	ThingworxPersistenceProvider: Maximum number of stream entries to queue	2,000,000
queueSize	ThingworxPersistenceProvider: Number of stream entries currently queued	0
totalWritesQueued	ThingworxPersistenceProvider: Number of stream entries that have been queued	33,040
totalWritesPerformed	ThingworxPersistenceProvider: Number of stream entries that have been performed	33,040
numberOfProcessingThreads	ThingworxPersistenceProvider: Number of processing threads	50