IOT EDC Reference Benchmark:

Leveraging Dell and VMWare for Asset Monitoring in Connected Factories

Document Version 1.0
October 2020

PTC Inc., 121 Seaport Boulevard, Boston, MA 02210 USA

Table of Contents

Document Version History

Revision Date	Version	Description of Change
October 2020	1.0	Initial document version

Acknowledgements

Please join us in thanking Bhagyashree Angadi, Brian Anzaldua, Todd Edmunds, Mike Hayes, and the Dell Customer Solution Center team in Limerick, Ireland for working with the IOT Enterprise Deployment Center on this benchmark.

What is a Reference Benchmark?

A great way to evaluate how an IOT implementation will perform is to compare it against a known reference. This can help you to:

- Understand the results and limitations in a known reference scenario
- Identify what differences exist between the implementation and the reference
- Evaluate how those differences change the behavior of the system

The purpose of this document is to provide a known reference scenario that can be used for these purposes and is targeted at a reader familiar with ThingWorx architecture and implementations.

Scenario Overview

As an extension of our Connected Factory Reference Benchmark performed on Microsoft Azure, PTC partnered with Dell Technologies to create a baseline that illustrates the effectiveness of ThingWorx and Kepware combined with Dell and VMWare technologies to create solutions for on-premises and hybrid Connected Factory implementations.

Like most asset monitoring use-cases, Edge size largely defines the scalability of a Connected Factory scenario. Variations in Edge size are made by adjusting the number of connected assets, the number of properties or data items per asset, and the frequency at which these properties are sent to ThingWorx.

This Reference Benchmark will focus on the first two configurations in Figure 1 – smaller Connected Factory implementations with one to three ThingWorx Kepware Server instances connected to a single-node ThingWorx Foundation server. Future benchmarks will illustrate the capabilities of combined high availability capabilities offered by Dell, VMWare and PTC.

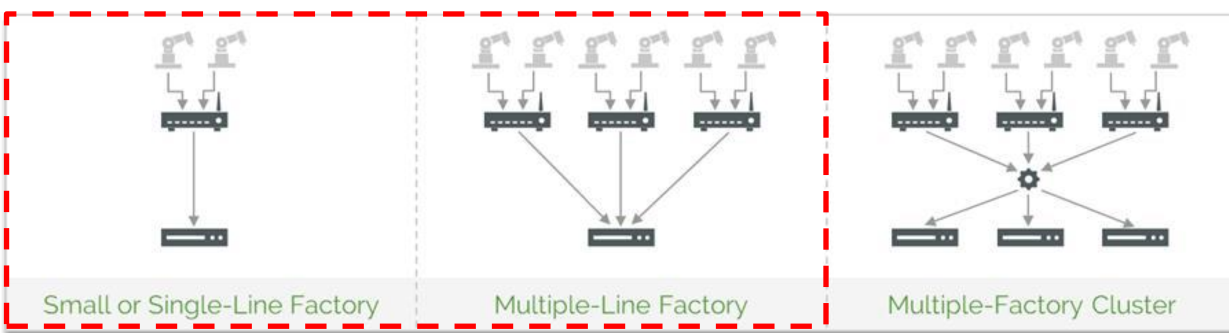


Figure 1 – Common asset monitoring implementation scenarios in a Connected Factory

The business logic and variables used in this simulation are identical to the Connected Factory benchmark performed on Microsoft Azure - the deployment architecture is held constant throughout these tests to help demonstrate the limits of a given configuration. Deployment changes that may improve the results of an unsuccessful simulation (such as adding CPUs or Memory to a specific virtual machine) may be discussed but will not be validated as part of this benchmark document.

Use Case Overview

The deployment architecture for a healthy Connected Factory implementation often looks similar in design and function to a Connected Product scenario. Generally, there are fewer individual edge devices in a Connected Factory, but each edge devices sends more frequent property updates to the ThingWorx Foundation server.

Asset Monitoring is typically achieved through application logic that checks if one or more changed property values indicate that an alarm should be triggered. These alarms are added to a stream which is monitored by Operator users via ThingWorx mashups. In addition, there is logic that runs once every 30 minutes to create a status roll-up of all Factory Assets for Manager users.

The overall implementation must have enough resources to handle this steady state workload, plus headroom for any brief spikes in either edge device or user activity. A scenario is deemed unsuccessful when data loss or delays in event or user request processing are observed.

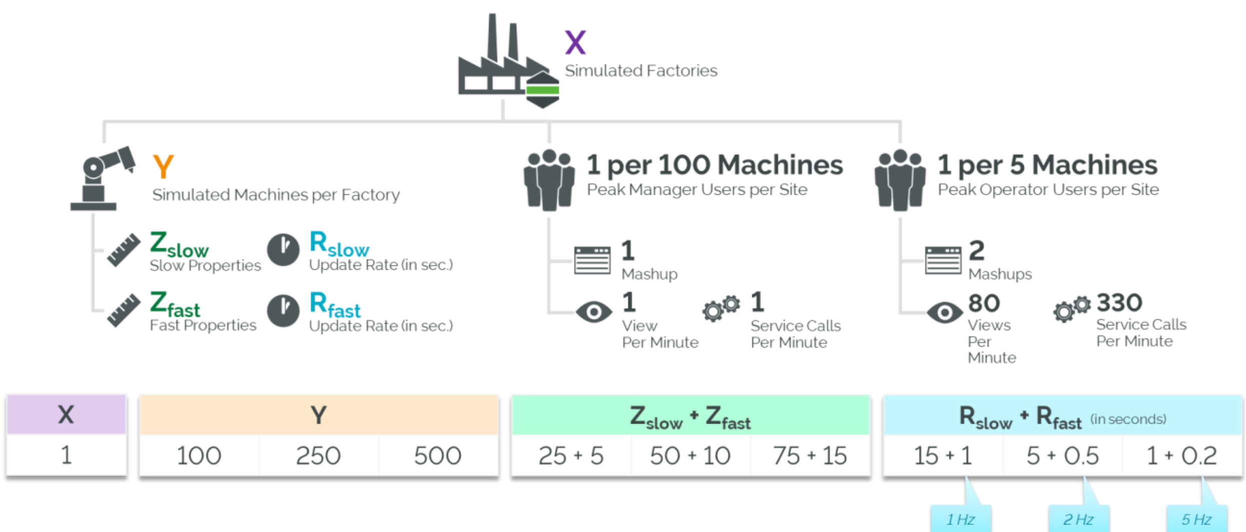


Figure 2 - This infographic outlines the benchmark scenario. Variations come from changing the number of assets (Y) per Factory (X), the number of properties per asset (Z), and the rate of property changes (R).

User Load

In a factory asset monitoring use-case, the typical user workload is to view historical device data and respond to triggered alarms. However, the simulated use-case also includes a real-time monitoring view, like seeing property values in a display as they come in (current state of properties, included in the operator view), and status roll-ups which run less frequently and depict the state of an entire line or factory (included in the manager view).

The operator mashup therefore contains real-time property information via the Property Display widget and historical property information via the Time Series Chart widget (with drop-down menus fueling both of these charts). There is also a Grid widget displaying all the alarms for a particular Thing, and a List widget allowing operators to switch from one asset to another. A secondary mashup can be opened from this which allows operators to add notes, effectively acknowledging an alarm in the process. This mashup is called half as often, and the updates to the alarm tracking stream occur only 20% of the time.

The manager mashup shows the status of the entire factory, including a query to sort by factory and region (which does not apply in the first scenario) and a Grid widget containing all of the information about each factory: how many of the total Things are connected (a percent) and how many unacknowledged alarms there are. The roll-up logic for this runs once per hour, populating a data table for more rapid querying.

In this Connected Factory simulation, is was assumed that the number of operators and managers at the factory increases proportionally with in the number of assets. See Figure 2 above for a visual of the number of managers, the number of operators, and the corresponding traffic which they generated via their various activity.

Edge Load

Two sets of properties were simulated in this Connected Factory scenario:

- “Fast” properties which had no logic upon ingestion, but high scan rates
- “Slow” properties with lower scan rates but have associated business logic runs upon data change.

Assets (Y)	**Slow Prop** (Z_{slow})	**Fast Prop** (Z_{fast})	**Slow Freq.** (R_{slow})	**Fast Freq.** (R_{fast})	**Series Count** ($T\ x\ (Z_{fast} + Z_{slow})$)	**Expected WPS** ($(T \times Z) \div R$)
100	25	5	15 sec	1 sec	3,000	660
100	25	5	5 sec	0.5 sec	3,000	1,500
100	25	5	1 sec	0.2 sec	3,000	5,000
100	50	10	15 sec	1 sec	6,000	1,300
100	50	10	5 sec	0.5 sec	6,000	3,000
…	…	…	…	…	…	…
250	50	10	5 sec	0.5 sec	15,000	7,500
250	75	15	15 sec	1 sec	22,500	5,000
…	…	…	…	…	…	…
500	75	15	15 sec	1 sec	45,000	10,000

Chart 1 – A sample of the tests; the ingestion rate was adjusted by the variables in Figure 2.

Note that the scan rate on the ThingWorx Foundation server was set two times faster to protect against the possibility that tag value changes were missed between sample intervals. For example, if a tag is expected to change once per second, scan rate should be set to 500 milliseconds (to a fastest recommended scan rate of 100ms).

Simulation Parameters and KPIs

To confirm the success of the tests, the following KPIs were monitored:

	Ingestion	Processing	Visualization
Primary KPI	Value Stream Writes Per Second	Event Rate	HTTP Requests Per Second
Secondary KPIs	Value Stream Queue Size "Lost" data points (failed writes)	Platform CPU Utilization Event Queue Size (i.e. backlog)	HTTP Request Response times "Bad" HTTP Requests

In addition to these KPIs, Kepware Server log output was reviewed to ensure that there were no indications of lost data. Tests that failed with this pattern would contain an error message similar to the following in the Kepware Server logs:

```
One or more value change updates lost due to insufficient space in the connection
buffer. | Number of lost updates = #####.
```

Each simulation consisted of a four-hour execution of various Edge configurations, with the same business logic and user workload in place throughout.

Simulation Scenario

Implementation Architecture

With the support of Dell Technologies laboratory teams and equipment, the following Connected Factory implementation was deployed using Dell hardware, with ThingWorx Foundation and one or more Kepware Servers each deployed on VMWare virtual machines within the same rack-mounted physical hardware.

As all virtual machines were implemented within the same physical system, network bandwidth and latency considerations were not a factor in these results.

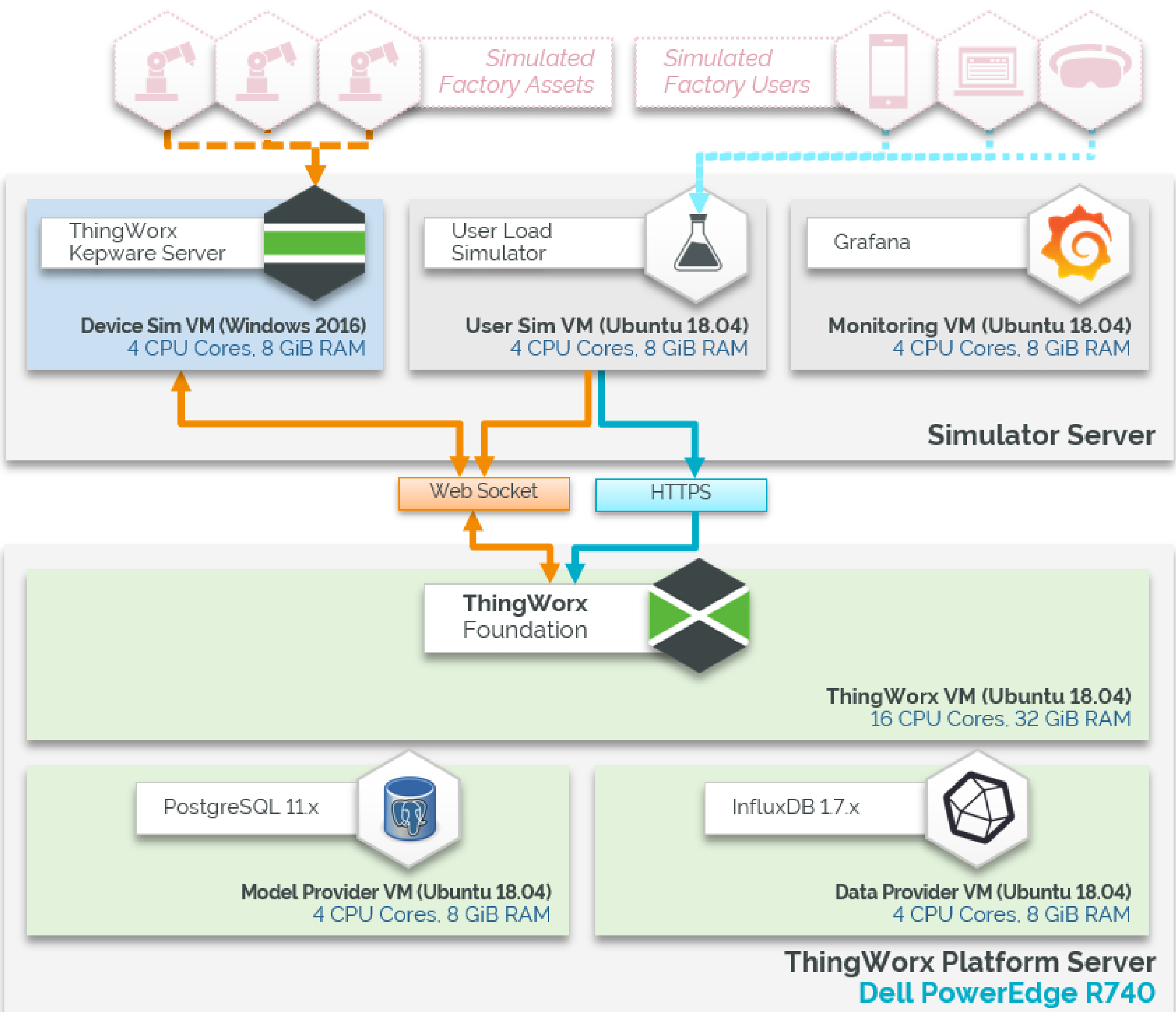


Figure 3 - The architecture: multiple Factory Assets from one Factory location connect to the Foundation server via one or more Kepware Servers.

The results tables that follow are grouped by property update frequency: All slow properties in that chart will use the larger "S" frequency, and all fast properties the smaller "F" frequency, regardless of other variations.

ThingWorx Model Configuration

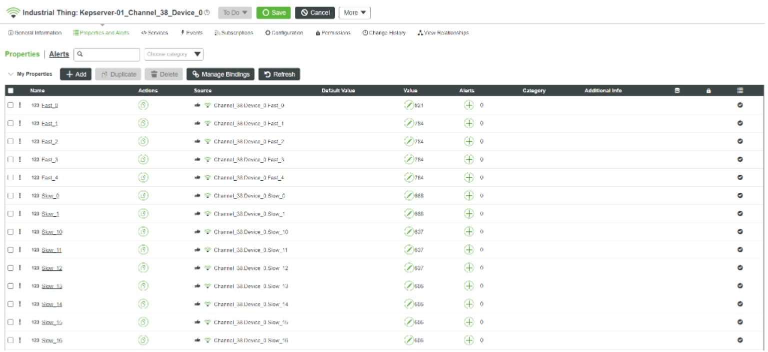


Figure 4 - *This image shows the property configuration within ThingWorx.*

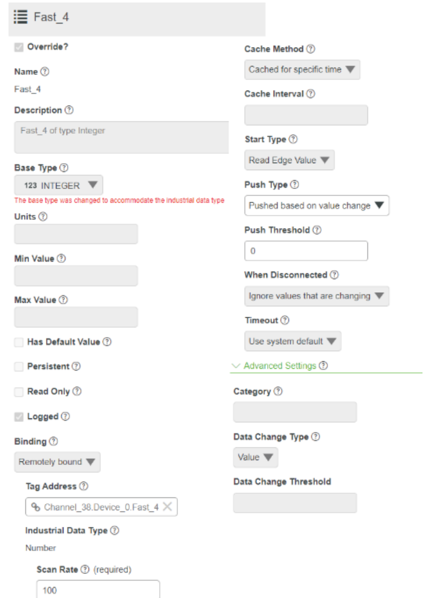


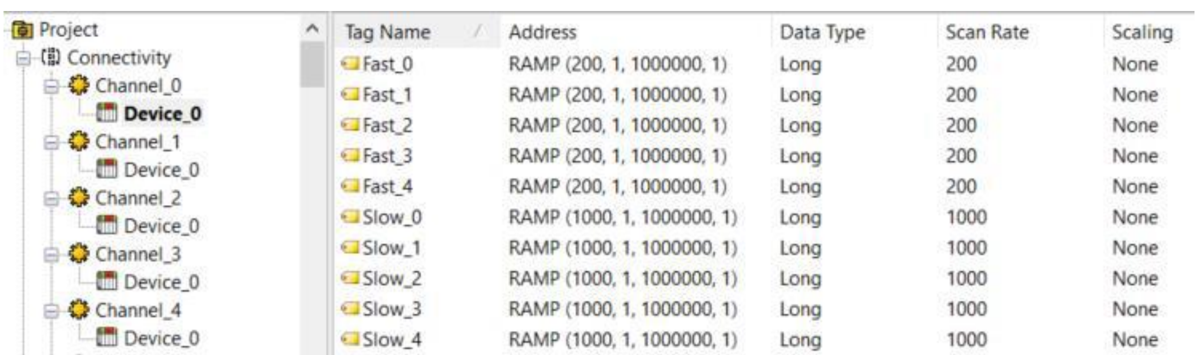
Figure 5 – ThingWorx property configuration. Note that the scan rate is 2x faster, aligning with the Kepware Server configuration in Figure 6 below.

Kepware Server Configuration

For these tests, Kepware Server's simulation driver was used to create changing data to send to ThingWorx.

This provides a level of data throughput that can be measured for these tests but note that this data does not fully represent the real-world scenario of polling industrial controllers and PLCs across a network.

Each simulation device in Kepware Server is analogous to a Thing in the Thing Model, while each tag in Kepware Servers configuration represents a property for that Thing. The tags were generated using an automated script run from a different server (with specifications shown in Figure 3).



Tag Name	Address	Data Type	Scan Rate	Scaling
Fast_0	RAMP (200, 1, 1000000, 1)	Long	200	None
Fast_1	RAMP (200, 1, 1000000, 1)	Long	200	None
Fast_2	RAMP (200, 1, 1000000, 1)	Long	200	None
Fast_3	RAMP (200, 1, 1000000, 1)	Long	200	None
Fast_4	RAMP (200, 1, 1000000, 1)	Long	200	None
Slow_0	RAMP (1000, 1, 1000000, 1)	Long	1000	None
Slow_1	RAMP (1000, 1, 1000000, 1)	Long	1000	None
Slow_2	RAMP (1000, 1, 1000000, 1)	Long	1000	None
Slow_3	RAMP (1000, 1, 1000000, 1)	Long	1000	None
Slow_4	RAMP (1000, 1, 1000000, 1)	Long	1000	None

Figure 6 - A screenshot from Kepware Server showing the tag configuration. This run had 30 properties total, 5 fast and 25 slow. Note that while a scan rate can be set within Kepware Server, when integrated with ThingWorx this value will be overridden by the Scan Rate set in the ThingWorx Model Configuration (as shown in Figure 5).

Simulation Summary

Matrix 1 – 15 Second Slow Properties + 1 Second Fast Properties

S: 15s F: 1000ms		Frequency (R) Number of Things (Y) 100	250	500
Frequency (R) Properties per Thing (Z)	25 + 5	**WPS:** 667 **CPU Min/Avg/Max:** 3% / 4% / 14% **Memory Min/Avg/Max:** 11% / 11% / 12%	**WPS:** 1,667 **CPU Min/Avg/Max:** 7% / 8% / 19% **Memory Min/Avg/Max:** 11% / 12% / 12%	**WPS:** 3,333 **CPU Min/Avg/Max:** 12% / 13% / 19% **Memory Min/Avg/Max:** 38% / 39% / 39%
	50 + 10	**WPS:** 1,333 **CPU Min/Avg/Max:** 5% / 8% / 14% **Memory Min/Avg/Max:** 28% / 28% / 28%	**WPS:** 3,333 **CPU Min/Avg/Max:** 13% / 14% / 20% **Memory Min/Avg/Max:** 46% / 46% / 47%	**WPS:** 6,667 **CPU Min/Avg/Max:** 22% / 23% / 25% **Memory Min/Avg/Max:** 12% / 12% / 13%
	75 + 15	**WPS:** 2,000 **CPU Min/Avg/Max:** 9% / 12% / 14% **Memory Min/Avg/Max:** 38% / 38% / 38%	**WPS:** 5,000 **CPU Min/Avg/Max:** 16% / 18% / 23% **Memory Min/Avg/Max:** 29% / 29% / 29%	**WPS:** 10,000 **CPU Min/Avg/Max:** 35% / 38% / 41% **Memory Min/Avg/Max:** 26% / 58% / 61%

Matrix 1 Analysis

For the hardware configuration used in these simulations, all tests were successful.

The 10,000 WPS test configuration would represent a well-sized implementation under steady state load, with headroom that could be used to implement more complex IOT application logic for a specific use-case, and/or to handle spikes in activity from users or edge devices.

The other test scenarios performed in this matrix were somewhat under-sized for the hardware configuration selected and would likely be successful with fewer CPU and Memory resources allocated to the ThingWorx Platform virtual machines.

Matrix 2 – 5 Second Slow Properties + 500 Millisecond Fast Properties

S: 5s F: 500ms	Frequency (R) — Number of Things (Y): 100	250	500
Frequency (R) — Properties per Thing (Z): 25 + 5	**WPS:** 1,500 **CPU Min/Avg/Max:** 7% / 8% / 26% **Memory Min/Avg/Max:** 11% / 11% / 22%	**WPS:** 3,750 **CPU Min/Avg/Max:** 13% / 15% / 21% **Memory Min/Avg/Max:** 12% / 13% / 13%	**WPS:** 7,500 **CPU Min/Avg/Max:** 24% / 26% / 30% **Memory Min/Avg/Max:** 46% / 46% / 47%
50 + 10	**WPS:** 3,000 **CPU Min/Avg/Max:** 13% / 14% / 27% **Memory Min/Avg/Max:** 11% / 12% / 12%	**WPS:** 7,500 **CPU Min/Avg/Max:** 26% / 29% / 34% **Memory Min/Avg/Max:** 28% / 28% / 28%	**WPS:** 14,650 (expected 15,000) **CPU Min/Avg/Max:** 47% / 49% / 56% **Memory Min/Avg/Max:** 46% / 61% / 66% *(**Note**: 2 Kepware Servers)*
75 + 15	**WPS:** 4,500 **CPU Min/Avg/Max:** 22% / 23% / 31% **Memory Min/Avg/Max:** 15% / 16% / 16%	**WPS:** 11,250 **CPU Min/Avg/Max:** 43% / 45% / 51% **Memory Min/Avg/Max:** 38% / **73%** / **76%** *(**Note**: 2 Kepware Servers)*	**WPS:** 21,480 *(expected 22,500)* **CPU Min/Avg/Max:** **85%** / **88%** / **90%** **Memory Min/Avg/Max:** 40% / **75%** / **82%** *(**Note**: 3 Kepware Servers)*

Matrix 2 Analysis

The three runs over 10,000 WPS on this page all used more than one instance of Kepware to distribute network communication and avoid bottlenecks. While Kepware Server can generally handle 10,000 WPS in ideal network conditions, it is advisable to design your implementation with enough headroom for spikes or less-than ideal network bandwidth or latency.

While the 11,250 WPS run was successful, memory utilization was above 70% under steady state load. This configuration could be sensitive to spikes in edge or user activity.

The 21,480 WPS run was not successful. ThingWorx CPU utilization was above 80%, leaving too little headroom for spikes in edge or user activity. Data loss was also reported in the Kepware Server instances as ThingWorx struggled to keep up.

Thread dumps confirmed the high CPU was caused by the volume of business logic at these data rates. Options to overcome this could include one or more of the following:

- Vertical scale (or "sizing up") by adding CPU and Memory to the ThingWorx Foundation VM. Faster physical CPUs could also be considered if available.
  Note: *This same test is successful when executed with a 32-core, 64 GiB ThingWorx Foundation VM.*
- Horizontal scale (or "sizing out") by deploying a ThingWorx cluster with multiple nodes operating in parallel to distribute load (and also provide high availability options at a software level).
- If adjusting the hardware footprint is not possible, reducing the frequency or complexity of the business logic within your ThingWorx application could also be considered (For example, trigger more complex, multi-property rules on a timer, instead of automatically with every data change).

Matrix 3 – 1 Second Slow Properties + 200 Millisecond Fast Properties

S: 1s F: 200ms			Frequency (R) — Number of Things (Y): 100	250	500
Frequency (R)	Properties per Thing (Z)	25 + 5	**WPS:** 4,960 (expected 5,000) **CPU Min/Avg/Max:** 20% / 22% / 28% **Memory Min/Avg/Max:** 38% / 38% / 38%	**WPS:** 11,836 (expected 12,500) **CPU Min/Avg/Max:** 42% / 43% / 51% **Memory Min/Avg/Max:** 15% / 49% / 50% *(**Note**: 2 Kepware Servers)*	**Target WPS:** 25,000 Not executed *(Requires 3 Kepware Servers + larger Foundation instance)*
		50 + 10	**WPS:** 9,920 (expected 10,000) **CPU Min/Avg/Max:** 44% / 45% / 49% **Memory Min/Avg/Max:** 47% / 47% / 47%	**Target WPS:** 25,000 Not executed *(Requires 3 Kepware Servers + larger Foundation instance)*	**Target WPS:** 50,000 Not executed *(Requires 5-6 Kepware Servers + larger Foundation instance)*
		75 + 15	**WPS:** 14,282 (expected 15,000) **CPU Min/Avg/Max:** 65% / 66% / 68% **Memory Min/Avg/Max:** 57% / 58% / 58% *(**Note**: 2 Kepware Servers)*	**Target WPS:** 37,000 Not executed *(Requires 4 Kepware Servers + larger Foundation instance)*	**Target WPS:** 75,000 Not executed *(Requires 8 Kepware Servers + larger Foundation instance)*

Matrix 3 Analysis

While the observed writes-per-second in these tests was slightly below the expected value, the Kepware Servers did not report data loss. Based on this, the tests are considered successful as the slightly reduced rate is being caused by the simulation setup itself, not Kepware Server or ThingWorx.

The 25,000+ WPS tests at this data rate were not executed as they would fail for the same reasons as the 22,500 WPS test from the prior page without allocating additional hardware resources.

Analysis and Conclusions

The deployment architecture selected for these simulations performed best on Edge configurations between 9,000 and 11,250 writes per second.

As Edge data ingestion rates approached 12,000 WPS, ThingWorx Foundation CPU and Memory consumption became the primary limiting factor. These limits were encountered due to the amount and complexity of business logic being used as part of this simulation.

Increasing the CPU and Memory allocated to the ThingWorx Foundation virtual machine, and/or reducing the complexity or frequency of business logic execution, would enable this deployment to scale to higher data ingestion rates.

In a Dell/VMWare architecture, the close proximity of Kepware Server and ThingWorx Foundation provides ideal conditions for network throughput between these components. Combined with the ability to easily monitor and resize virtual machines as your business needs evolve, these hardware configurations can be very effective in on-premises or hybrid deployment scenarios.