WHY USE INFLUXDB IN A SMALL APPLICATION

When to Include InfluxDB in the ThingWorx Development Lifecycle

THE SHORT ANSWER

InfluxDB is a time series database designed specifically for data ingestion. Historically, InfluxDB has been viewed as a high-scale expansion option for ThingWorx: a way to ensure the application works as intended, even when scaled up to the enterprise level. This is certainly one way to view it, because when there are many, many remote things, each with a lot of properties writing to the Platform at short intervals, then InfluxDB is a sure choice. However, what about in smaller applications? Is there still a benefit to using an optimized data ingestion tool in any case? The short answer is: yes, there is!

Using InfluxDB for optimized data ingestion is a good idea even in smaller-sized applications, especially if there are plans to scale the application up in the future. It is far better to design the application around InfluxDB from the start than to adjust the data model of the application later on when an optimized data ingestion process is required. PostgreSQL and InfluxDB simply handle the storage of data in different ways, with the former functioning better with many Value Streams, and the latter with fewer Value Streams. Switching the way data is retained and referenced later, when the application is already on the larger side, causes delays in growing the application larger and adding more devices. Likewise, if the Platform reaches its ingestion limits in a production environment, there can be costly downtime and data loss while a proper solution (which likely involves reworking the application to work optimally with InfluxDB) is implemented.

Don't think that InfluxDB is for expansion only; it is an optimized ingestion database that has benefits at every level of the application development lifecycle. From the end to end, InfluxDB can ensure reliable data ingestion, reduced risk of data loss, and reduced memory and CPU used by the deployment overall. Preliminary sizing and benchmark data is provided in this article to explain these recommendations. Consider how ThingWorx is ingesting data now, how much CPU and other resources are used just for acquiring the data, and perhaps InfluxDB would seem a benefit to improve application performance.

THE LONG ANSWER

In order to uncover just how beneficial InfluxDB can be in any size application, the IoT Enterprise Deployment Center has run some simulations with small and medium sized applications. The use case in the simulation is simple with user requests coming from a collection of basic mashups and data ingestion coming from various numbers of things, each with a collection of "fast" and "slow" properties which update at different rates. This synthetic load of data does not include a more complete application scenario, so the memory and CPU usage shown here should not be used as sizing recommendations. For those types of recommendations, stay tuned for the soon-to-release ThingWorx 9.0 Sizing Guide (or check out the current 8.5 Sizing Guide).

COMPARING RUNS

When determining the health of the ThingWorx Platform, there are several categories to inspect: Value Stream Queue Rate and Queue Size, HTTP Requests, and the overall Memory and CPU use for each server. Using Grafana to store the metrics results in charts like those below which can easily be compared and contrasted, and used to

evaluate which hardware configuration results in the best performance. The size of the numbers on the vertical axis indicate total numbers of resources used for that metric, while the slope or trend of each chart indicates bottlenecks and inadequate resource allocation for the use case.

In this case, all darker charts represent data from PostgreSQL ONLY configurations, while the lighter charts represent the InfluxDB instances. Because this is not a sizing guide, whether each of these charts comes from the small or medium run is unimportant as long as they match (for valid comparisons between with Influx and without it). The smaller run had something like 20k Things, and the larger closer to 60k, both with 275 total Platform users (25 Admins) and 3 mashups, which were each called at various refresh rates over the course of the 1-hour testing period. Note that in the PostgreSQL ONLY instances, there were more Thing Templates and corresponding Value Streams. This change is necessary between runs because only with fewer Value Streams does InfluxDB begin to demonstrate notable improvements.

The most important thing to note is that the lighter charts clearly demonstrate better performance for both size runs. Each section below will break down what the improvement looks like in the charts to show how to use Grafana to verify the best performance.

Value Stream Queue

The vertical axis on the Value Stream Queue Rate chart shows how many total writes per minute (WPS) the Platform can handle. The average is 10 WPS higher using InfluxDB in both scenarios, and InfluxDB is also much more stable, meaning that the writes happen more reliably. The Value Stream Queue Size chart demonstrates how well the writes within the queue are processed. Both of these are necessary to determine the health of data ingestion.

If the queue size were to increase and trend upward in the lighter Queue Size chart, then that would mean the Platform couldn't handle the higher ingestion rate. However, since the Queue Size is stable and close to 0 the entire time, it is clear that the Platform is capable of clearing out the Value Stream Queue immediately and reliably throughout the entire test.

FIGURE 1 – THESE REPRESENT THE DATA GETTING STORED INTO THE DATA PROVIDER. NOTE: THE FORMER IS MUCH LOWER THAN THE LATTER.

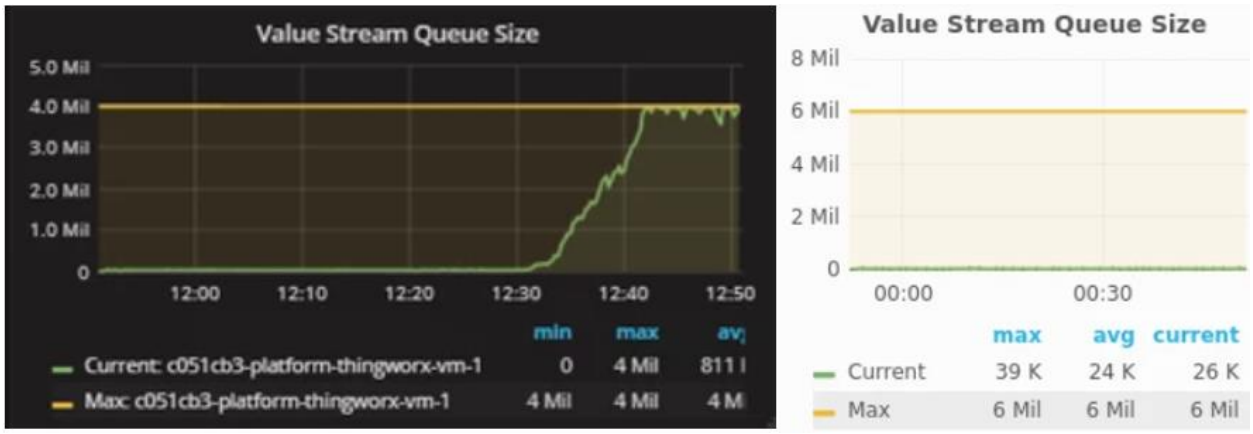

FIGURE 2 – NOTE THE DATA LOSS IN THE NON-INFLUX INSTANCE (THE QUEUE IN GREEN REACHES THE MAX IN YELLOW). THE INFLUX INSTANCE HAS LESS TROUBLE CLEARING OUT THE QUEUE, AS DEMONSTRATED BY THE CONSISTENTLY LOW QUEUE SIZE.

HTTP REQUESTS

Taking the strain of ingestion off of the Platform's primary database frees its resources up for other activities. This in turn improves the performance and reliability of the Platform to respond to HTTP requests, those which in a typical application are used to aggregate data into smaller data stores (depending on the use case) and which render the mashups for the end users. The business logic and mashups can be more complex when there is one database designated for ingestion (InfluxDB) and one for everything else (PostgreSQL).

Likewise, the nature of Postgres lends well towards this differentiation, given that there are many more database tables required for supporting the HTTP requests, something Postgres does well. That leaves Influx to handle the time-series data and ingestion, and those are the primary strengths of that software as well. So, splitting the load across multiple servers in this way results in smaller server sizes overall, each which is stream-lined and optimized to handle exactly what it is given by the Platform.

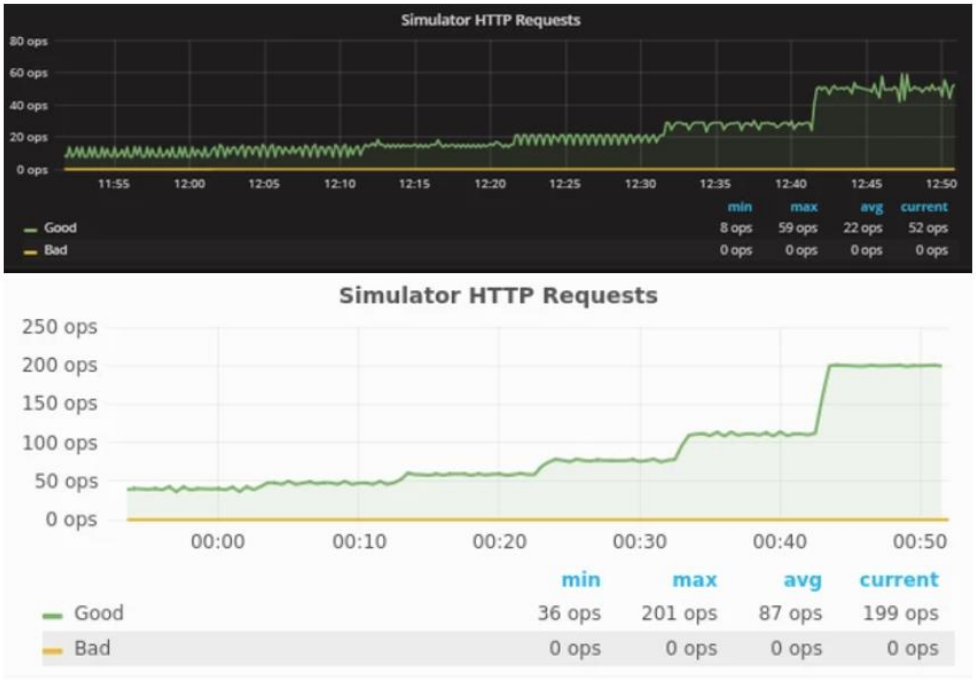

FIGURE 3 – THE DARKER CHART SHOWS A LOT OF CHOPPINESS, MEANING THAT WHILE THE PLATFORM WAS RESPONDING THE WHOLE TIME, IT WAS NOT DOING SO RELIABLY. THE SMOOTHER SECOND CHART SHOWS HOW MUCH EASIER THE PLATFORM CAN HANDLE THESE REQUESTS WHEN THE LOAD IS DISTRUBITED INTELLIGENTLY ACROSS MULTIPLE SERVERS, EACH OPTIMIZED FOR THE TYPE OF DATA THEY RECEIVE. THE "STAIRCASE" SHAPE OCCURS BECAUSE THE SIMULATOR INCREASES THE WORK LOAD EVERY 10 MINUTES UNTIL IT BREAKS.

Note that in both of these charts, there are no bad requests, so both would seem to be successful runs. However, as future charts will demonstrate more clearly, there is a catastrophic failure when the load is increased around 12:30p. The simulation ends before the server begins to show any real symptoms of

the issue, and that is why there are no bad requests. The maximum Operations Per Second (OPS) in the Hardware Specifications and Performance section is taken from before the failure begins.

Clearly the InfluxDB instance has better performance given that the average Operations Per Second (OPS) is substantially higher, nearly 4 times what is seen in the PostgreSQL ONLY instance. Obviously how well the Platform manages the business logic and mashup loading will depend on a lot of factors. In this test scenario, the OPS was increased by increasing the mashup refresh rate on the InfluxDB instances (which could handle over double the operations). Likewise, the number of Stream writes to the PostgreSQL database could be double what it was when PostgreSQL was the only database. Therefore, configuring InfluxDB for the data ingestion and leaving Postgres for the rest of the application certainly makes the load much easier on the Platform, and the same would be true even in a much more complex scenario.

Memory and CPU

The important thing here is to keep the memory use low enough that any spikes in usage won't cause a server malfunction. CPU Usage should stay at or below around 75%, and Memory should never exceed around 80% of the total allocated to the server. The sizing guides can help determine what this allocation of memory needs to be.

Of note in these charts is the slight, upward slope of the CPU usage in the darker chart, indicating the start of a catastrophic failure, and the difference in the total memory needed for the ThingWorx Platform and Postgres servers when Influx is used or not. As is apparent, the servers use much less memory when the database load is split up intelligently across multiple servers.

FIGURE 4 – THE THINGWORX CPU IS ABOUT THE SAME HERE AS IN THE INFLUXDB CONFIGURATION BELOW BUT LOOK AT HOW MUCH MORE MEMORY BOTH THE PLATFORM AND THE POSTGRES DATABASE NEED ALLOCATED TO THEM IN THIS CONFIGURATION (64 GB A PIECE).

ALSO NOTE THE JUMP IN CPU AND MEMORY USAGE AFTER 12:30P. THIS IS REFERENCED IN THE PREVIOUS SECTION, AND THE SLOPE UPWARD OF THE USAGE AFTER THAT POINT INDICATES THE START OF A CATASROPHIC FAILURE. THE TEST ENDS TOO SOON TO SEE ANY SYMPTOMS OF FAILURE, BUT IT IS A SURE THING AFTER THE INCREASE IN LOAD AROUND 12:30P.

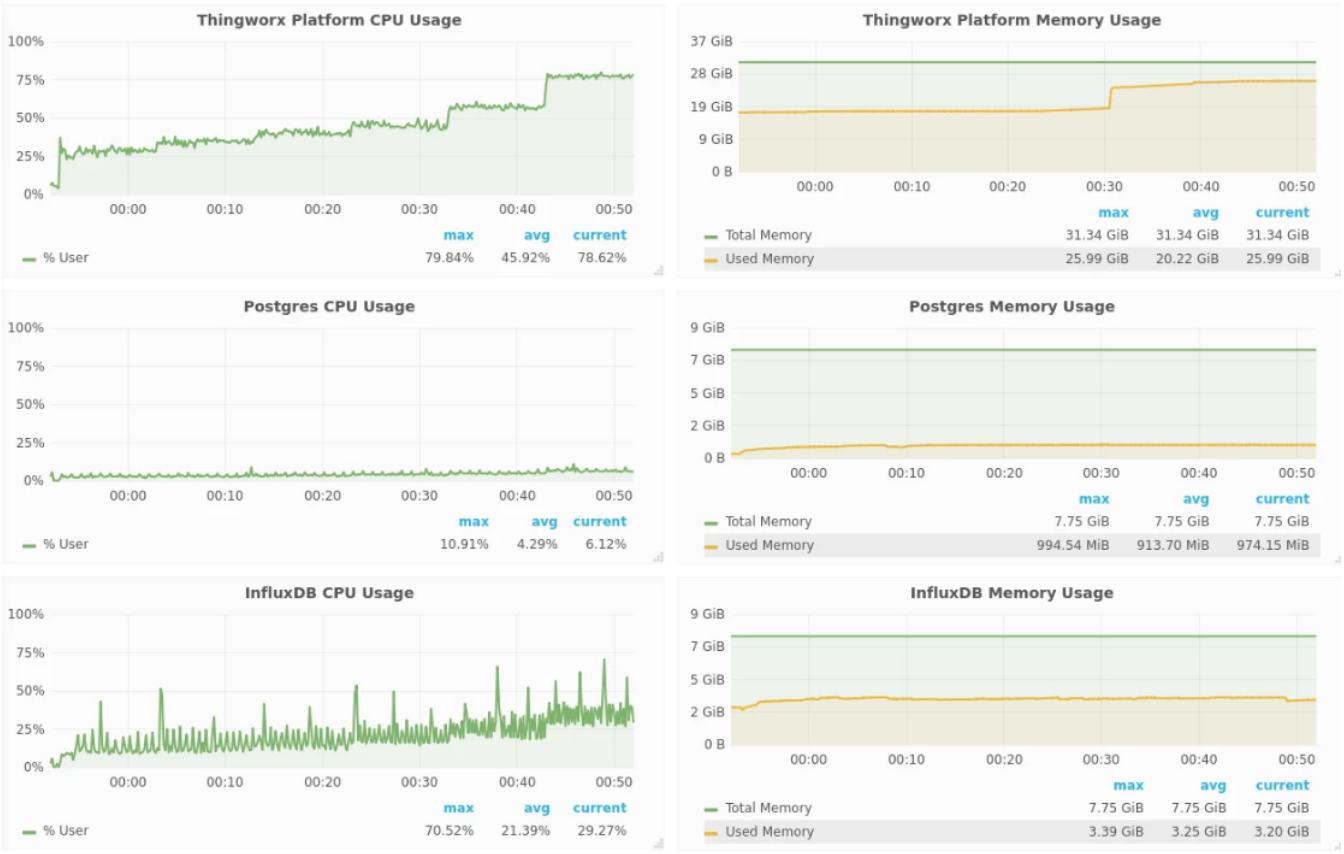

FIGURE 5 – INFLUX NEEDS AN EXTRA SERVER, BUT THE SIZE OF THE INFLUX AND POSTGRES SERVERS TOGETHER IS LESS THAN HALF THE SIZE AS THAT REQUIRED FOR THE SINGLE POSTGRES DATABASE IN THE POSTGRES ONLY CONFIGURATION (8 GB). THINGWORX IS SMALLER TOO (32 GB).

HARDWARE SPECIFICATIONS AND PERFORMANCE

These are the exact specifications for each simulated instance, broken down by size and whether InfluxDB is configured or not. Note that some of the hardware specifications may be more than is necessary real-world use case depending. As stated previously, this document is not a sizing guide (use the official ThingWorx Sizing Guide).

Note that the maximum number of WPS and OPS are shown here. The maximums are higher in the InfluxDB scenarios, meaning that even with smaller-sized servers, the InfluxDB configurations can handle much greater loads.

Description	ThingWorx	PostgreSQL	InfluxDB	Ingest WPS	HTTP OPS
Medium w/o InfluxDB	16 vCPU, 64 GiB	16 vCPU, 64 GiB	n/a	26k	20*
Medium **with InfluxDB**	16 vCPU, 32 GiB	4 vCPU, 8 GiB	4 vCPU, 8 GiB	61k	200
Small w/o InfluxDB	8 vCPU, 16 GiB	8 vCPU, 32 GiB	n/a	12k	31
Small **with InfluxDB**	8 vCPU, 16 GiB	2 vCPU, 4 GiB	2 vCPU, 4 GiB	21k	73

* note that this is not the largest number in the HTTP Requests chart, but it is the max before the start of the test failure after 12:30p.

SUMMARY

In conclusion, if InfluxDB may at some point be needed in the lifecycle of an application, because the expected number of things or the number of properties on each thing is large enough that it will max the limitations of the Platform otherwise, then InfluxDB should be used from the very start. There are benefits to using InfluxDB for data ingestion at every size, from performance to reliability, and of course the obviously improved scalability as well.

Reworking the application for use with InfluxDB later on can be costly and cause delays. This is why the benefits and costs associated with an InfluxDB-centric hardware configuration should be considered from the start. More servers are required for InfluxDB, but each of these servers can be sized smaller (depending on the use case), and all of this will affect the overall cost of hosting the ThingWorx application. The benefits of InfluxDB are especially pronounced when used in conjunction with clusters, which will be demonstrate fully in the 9.0 Sizing Guide (soon to be released). If InfluxDB is used to interface with the clusters, then there are even more resources to spare for user requests.

It is considered ThingWorx best practice for high ingestion customers to make use of InfluxDB in applications of any size. Note, though, that this will mean the number of Value Streams per Influx Database will need to be limited to single digits. We hope this helps, and from everyone here at the EDC, happy developing!