PNNL-SA-26406

Pacific Northwest
NATIONAL LABORATORY
Proudly Operated by **Battelle** *Since 1965*

VOLTTRON™ Deployment and Scalability

Lessons and Enhancements in FY16/FY17

April 2017

J Haack
B Akyol
W Cowley
C Allwardt
K Monson
T Kuruganti
J Sanyal
J Nutaro
D Fugate
O Ozmen
C Winstead

U.S. DEPARTMENT OF ENERGY

Prepared for the U.S. Department of Energy
under Contract DE-AC05-76RL01830

VOLTTRON™ Deployment and Scalability

J Haack
B Akyol
W Cowley
C Allwardt
K Monson
C Winstead
T Kuruganti
J Sanyal
J Nutaro
D Fugate
O Ozmen

April 2017

Prepared for
the U.S. Department of Energy
under Contract DE-AC05-76RL01830

Pacific Northwest National Laboratory
Richland, Washington 99352

Executive Summary

Throughout Fiscal Years (FY) 2016 and early 2017, Pacific Northwest National Laboratory (PNNL) and Oak Ridge National Laboratory investigated the effects of scaling on VOLTTRON™ and its components. The VOLTTRON team identified and made recommendations to performance issues that impacted the ability of early versions of VOLTTRON to scale effectively. Throughout FY16, and as an ongoing strategy, scaling and management-at-scale remain a primary design consideration for the platform.

This document captures the experiments and practical lessons gathered through early FY17. We performed stress tests to identify potential bottlenecks in the system and learned numerous practical lessons through providing support to external organizations deploying the system, especially the commercial companies Intellimation and Transformative Wave Technologies. The real-world applications and interactions with the VOLTTRON community drove many development and testing priorities described in this report.

Through this work, we made improvements to essential systems of the platform including:

- Reducing duplicate driver publishes by at least half to decrease traffic on the message bus and improve speed and latency
- Spreading out driver data collection to prevent overloading the underlying devices
- Refactoring the ActuatorAgent to allow for controlling thousands of points at a time
- Improving Historian performance by a factor of four by reducing extraneous metadata
- Establishing alternative Historian deployment scenarios to meet the needs of commercial deployments
- Greatly simplifying the registration process for managing VOLTTRON instances through the management user interface

Additionally, we are creating infrastructure to allow for testing numerous large-scale VOLTTRON deployment topologies using the PNNL Research Cloud. This will allow tests to use the full platform and not merely provide simulated results. These investigations provided data on performance under load at various scales and recommendations in application program interface changes for automated installation.

This report summarizes testing scenarios used, performance improvements implemented and their results, and our plans and recommendations for maintaining momentum in improving VOLTTRON scalability moving forward. We will continue to gather valuable lessons learned through practical deployments and formal testing that will be added to this document and lead to increased performance and usability of the platform. Although we have improved the scaling and identified approaches to make the current version of VOLTTRON scale in most deployments, scaling issues will always exist and we hope to work with the community to resolve these challenges.

Acknowledgments

The authors would like to thank U.S. Department of Energy Buildings Technology Office for its continued support. We also thank the VOLTTRON community for providing practical lessons and experience throughout the year—VOLTTRON grows in capability and impact with each interaction.

Contents

Figures

Tables

1.0 Introduction

The VOLTTRON™ platform is built on five key features: flexibility, usability, security, scalability, and interoperability. While each feature is important, they are frequently in tension with each other; an increase in security can lead to a decrease in usability and scalability. Yet, they also complement and reinforce one another; flexibility can mean deploying VOLTTRON in ways that divide the load of a large building and therefore increase scalability. PNNL and ORNL sought to measure scalability in multiple dimensions. This report focuses on the different elements of VOLTTRON's scalability, how it interacts with other priorities, and how it can be enhanced by leveraging them.

This report is a living document that will be enhanced as more studies are performed and more practical lessons are learned as VOLTTRON fulfills increasingly varied use cases. Ideally the community will contribute their experience to this document and provide a wide range of perspectives on what VOLTTRON scalability means and how it can be achieved. This document also highlights where the platform could be improved to further support needs for large-scale deployments.

The software improvements made in Fiscal Year (FY) 2016 to 2017 resulted in the following improvements:

- Improved message bus performance by 4-10 times compared to the version 4.0 bus, bringing it closer to the base performance of the messaging library without sacrificing the VOLTTRON security improvements.
- Typical settings reduced traffic to, at most, ¼ the traffic of the original behavior. A reduction to 1/20 of the original volume is common.
- Prevented overwhelming of BACnet devices (especially serial devices connected to router). Smooths out message load on message bus.
- Allowed for commanding 4,000 simulated devices from a single VOLTTRON instance.
- Improved Historian performance by factor of four.
- On secure networks, Historians can be configured to write to a network accessible DB and increase throughput.
- In cases of long-term issues recording data, the backup cache will not grow infinitely.
- Increased cache performance by approximately 10 times and date time parsing by 30 times.
- Dramatically reduced the time to set up a platform from tens of minutes to almost no time.

Our results are summarized in Section 5.0. Section 7.0 includes our deployment recommendations for VOLTTRON systems in the field.

2.0 Current State

Developers at Pacific Northwest National Laboratory (PNNL) have deployed VOLTTRON numerous times, collecting data from buildings or researchers in their laboratories. Additionally, in FY16 larger scale practical deployments were performed at laboratories, universities, and commercial companies. Two exemplar deployments are described in this section to demonstrate how the platform was applied and what lessons were learned. These deployments informed development and scalability in ways that are not possible in a simulated environment.

In FY17, PNNL is performing additional testing at various scales using synthetic data in a private cloud environment.

2.1 Clean Energy Transactive Campus – PNNL Deployment

As part of the Clean Energy Transactive Campus (CETC) Project (Figure 1), PNNL performed a campus-wide deployment of VOLTTRON. VOLTTRON instances installed on 10 PNNL buildings are now pushing data to a centralized instance for storage and forwarding to external collaborators. Given the number of network boundaries crossed, this deployment is currently the most complex known deployment.

Within the CETC Project, VOLTTRON collection boards deployed on the isolated facilities network collect data from buildings and forward it to the main PNNL network via a firewall exception. A Historian on the central VOLTTRON instance sends data to Mongo databases on another network. A ForwardHistorian agent on this instance also forwards data to another VOLTTRON instance on a machine that is accessible from outside the PNNL network. This VOLTTRON then forwards the data once more to VOLTTRON instances running on external collaborators' networks. In all, data collected from the PNNL buildings passes through four VOLTTRON instances before reaching its ultimate destination. The observed time for archiving data including passing through four Historians is less than a second.

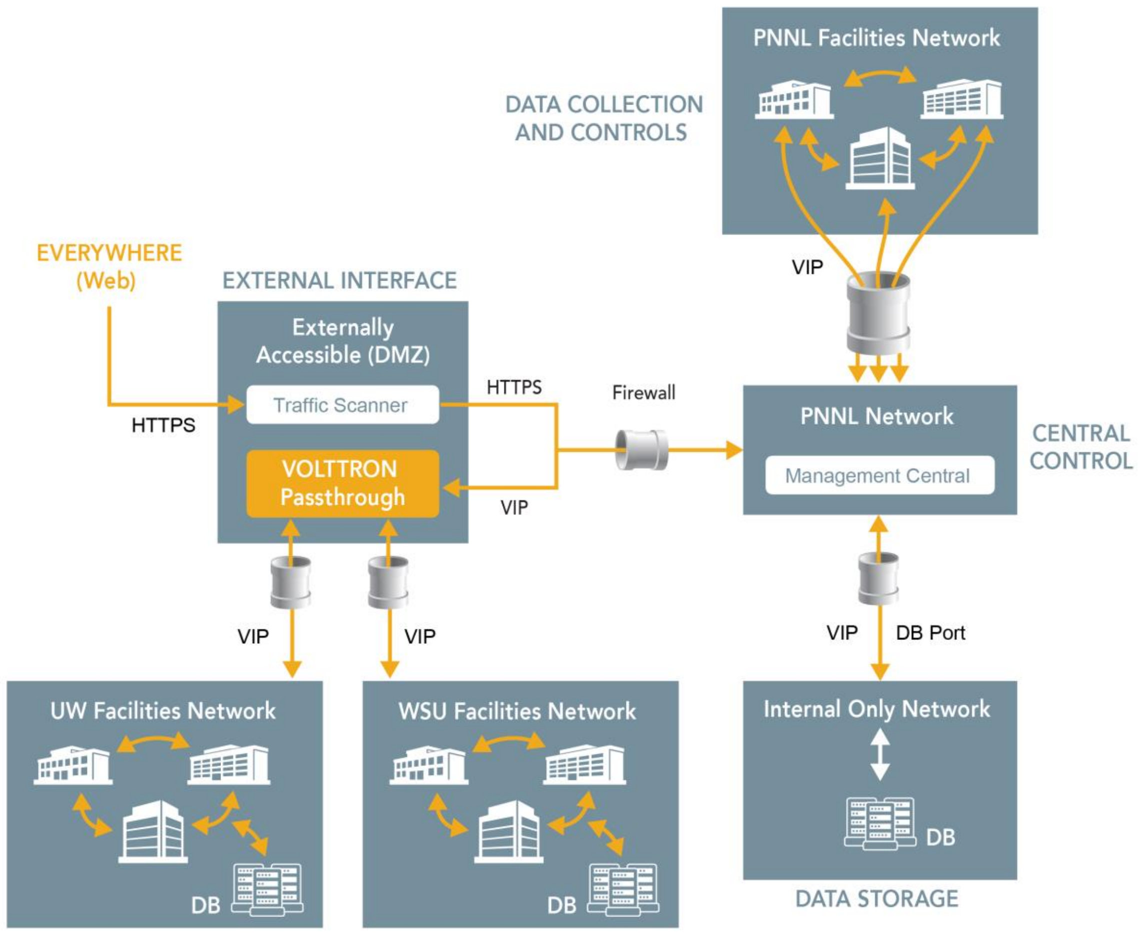

Figure 1. Clean Energy Transactive Campus Deployment on PNNL campus

2.2 Intellimation

Intellimation is an external company using VOLTTRON as part of their business to collect and analyze data for building owners. Their use case has provided a wealth of practical experience for deploying VOLTTRON in a real-life business use case. Intellimation has deployed VOLTTRON as part of their business model to collect data from customer buildings for analysis. The real-world experience gained by working with Intellimation proved invaluable to the VOLTTRON team.

Intellimation's deployment was very different from instances the VOLTTRON team had worked on previously. Intellimation is connecting to multiple geographically separated buildings and tying them together securely using an OpenVPN network. VOLTTRON is deployed on a small form-factor computer at the building site and communicates back to the server over a cellular network.

For the largest building from which Intellimation is collecting, 9,000 data points are collected every 15 minutes. Initial experiences with performance at this scale, using the default configuration, overwhelmed the building controller and the message bus on the collection box. To address this issue, the team enabled staggered driver startup. Drivers were set up to stagger the collection over a minute. Staggering the collection improved both the overloading of the building controller and the message bus.

Another performance issue encountered was the performance of the default MongoDB installation. The initial installation of the MongoDB at the start of the Intellimation installation was only configured to have data in tables with no indexes. As the size of the database grew, the database processing caused CPU usage to spike due to inefficient checks for uniqueness on data inserts. This would cause the entire system to become unresponsive and data to be lost. Applying an index relieved the problem and indexes are now part of the default database installation profiles.

The use of OpenVPN in the network communication puts the Intellimation deployment in a different category than typical deployments. OpenVPN protects communication by encrypting traffic and allowing machines to have trusted communication in a normally untrusted network space. The overhead of this protection in addition to the VOLTTRON vip layer created additional latency in the network transmission. As shown in Figure 2, given that configuration, using the vip layer is not a critical requirement. This allows Historians on the collection boxes to use direct connections to the MongoDB, which increases performance by skipping steps that would be required for the ForwardHistorian. MongoDB performance can be further increased by creating multiple replicated instances of the database. PNNL is using this improvement in the CETC Project.

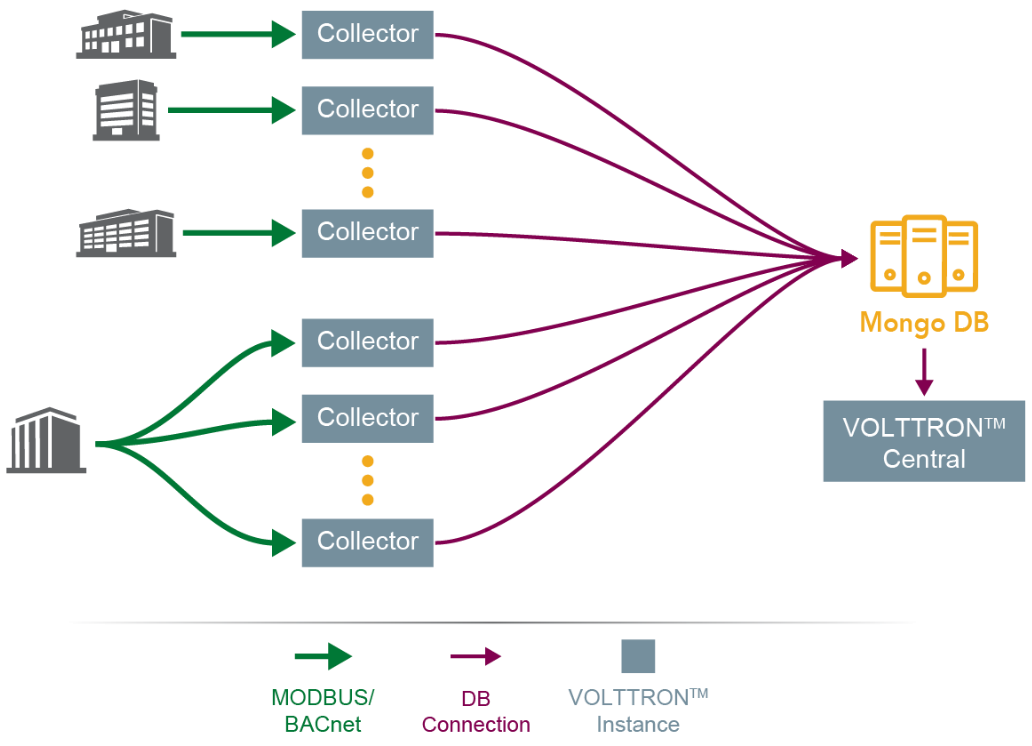

Figure 2. Deployment with Historians Writing Directly to a Remote Database

3.0 Components of a VOLTTRON Deployment

In this section, we describe the components of a VOLTTRON deployment that need to be considered for the scalability of the overall system. VOLTTRON is designed to scale with different computing hardware; therefore, hardware that runs VOLTTRON needs to be selected based on the functions it needs to perform. A different hardware requirement would be expected between the small board doing data collection versus the central server providing high-volume analytics. Different elements of the VOLTTRON software stack will be installed to support different topologies and use cases.

3.1 VOLTTRON™ Software

The resources needed by the VOLTTRON platform depend on the deployment mode. Services can be included or excluded depending on what actions that instance is expected to take. For instance, if no control actions are needed or allowed for connected devices, the ActuatorAgent need not be deployed. If no local storage of data is needed, then no database Historian is needed. The major components of VOLTTRON software that affect scalability will be discussed in the following sections.

3.1.1 Applications

Applications running on VOLTTRON are called “agents.” Agents are where the real work happens in a VOLTTRON deployment; the rest of the platform supports these agents as they perform their tasks. The purpose and needs of agents affect the needs of the rest of the platform. Do the agents work from local data or do they need a global view of the entire deployment? Do they require significant memory and/or processing power to do analysis? These considerations determine whether agents can run on the small edge devices or require a high-end centralized server.

3.1.2 Drivers

VOLTTRON drivers are configured to communicate with devices in the deployment. In cases where the devices work with BACnet and MODBUS, the configuration files need to be set up for the specifics of that deployment. VOLTTRON provides a script for finding BACnet devices on a network and then collecting information on available data points from those devices. This could potentially cause a temporary high load on the facilities network as this is a broadcast request to which all devices will respond.

In cases where VOLTTRON interacts with a building controller or devices with numerous points, it is more efficient to separate the collection into smaller collection subgroups. This scalability improvement can also prevent segmentation errors where the BACnet devices throw errors due to message sizes exceeding their limits.

3.1.3 Message Bus/Router

The underlying messaging technology of VOLTTRON is implemented on top of the ZeroMQ message bus with Publish/Subscribe (Pub/Sub) being the most central to the platform. VOLTTRON provides a security layer on top of Pub/Sub to authenticate and authorize publishers and subscribers. As is frequently the case, increasing security imposes a performance cost. Despite this added cost, the VOLTTRON team recommends (and sets the default to) the high-security mode of the message bus. Research into mitigating this cost was done for FY17 and is discussed in Section 5.0.

3.1.4 Historian (BaseHistorian, ForwardHistorian, MySQL, MongoDB, etc.)

Several types of Historian agents are available in the VOLTTRON system. The type of Historian being used depends on the need for data to reach specific networks and storage systems.

The BaseHistorian is the foundation for all Historians in the VOLTTRON platform and provides a backup cache for preventing data loss. The BaseHistorian pulls data from the message bus and immediately writes it to the SQLite-based backup cache. It then reads from this cache and calls the specific implementation (MySQL, Mongo, etc.). Since all data is first stored in this backup cache database, it can be a key bottleneck for data storage.

For VOLTTRON instances collecting data for remote storage, the ForwardHistorian fulfills the need to send data to a remote location. However, the ForwardHistorian becomes a bottleneck when it cannot forward data to a remote instance at the rate data is being published on the local instance. The ForwardHistorian may become backlogged and never catch up. This behavior was observed on installations where numerous devices were being collected at a high rate.

From this initial use case, the Forwarder has become the primary mechanism for securely transferring data from one instance to another for storage in a central repository. Alternate design options exist for when the data has no requirement to be presented as real-time collection.

When the platform needs to store data in a database, either locally or over the network, then a database Historian is appropriate (SQLiteHistorian, MySQLHistorian, MongoHistorian).

3.1.5 Database (MySQL, MongoDB)

If the platform needs to store data, a database installation and the matching database Historian are required. VOLTTRON supports multiple databases. If the performance of the writes to the database slows down a Historian and data cannot be written at the rate of collection, the Historian could fall behind. An overworked database could also cause performance issues for the platform as a whole by consuming too many system resources. This was observed in early MongoDB installations. Especially in cases where a central database is storing all data for a deployment, dedicating a machine for the database is a more efficient solution. For large-scale deployments, the VOLTTRON team recommends a multi-node, replicated database installation. The front-end load can be distributed evenly using a load balancer.

3.1.6 I/O

Performance of the underlying system storage greatly effects the performance of the Historian since all data is first written to the backup cache. Especially in the small boards, the speed of the solid state storage greatly impacts the performance of the system. For instance, stress tests showed that eMMC allowed for twice as much throughput as slower speed SD UHS-3 storage, which itself performed twice as well as standard SD cards.

3.1.7 Network

Network speed can provide a bottleneck and potentially impact the performance of some components of the platform. For instance, the MongoHistorian failed when writing directly to a remote Mongo database due to operating over a cell modem network. The Historian was timing out but never reconnecting. By refactoring the connection code, this issue was overcome.

4.0 Testing Scenarios

4.1 Existing Deployments

Although none of the existing deployments were created to serve the needs of scalability testing, they provided valuable real-world feedback. The development team helped debug performance and scalability issues and the experience with real-world deployment led to implementation changes that are quickly deployed in the field. These improvements include:

- Reducing duplicate driver publishes by at least half to decrease traffic on the message bus and improve speed and latency
- Spreading out driver data collection to prevent overloading the underlying devices
- Refactoring the ActuatorAgent to allow for controlling thousands of points at a time
- Extending the ActuatorAgent to allow for parallel multi-point retrieval
- Improving Historian performance by a factor of four by reducing extraneous metadata
- Improving data marshalling for data going into the Historian caching
- Establishing alternative Historian deployment scenarios to meet the needs of commercial deployments
- Greatly simplifying registration process for managing VOLTTRON instances through the management user interface (UI)
- Adding unique indices to the database schema to improve insertion time checks

4.2 Focused Component Scale Testing

The development team investigated specific components of software suspected of being scalability bottlenecks. As the message bus is a critical component to VOLTTRON, initial scale testing focused on message traffic load on the ZeroMQ message bus. Analyzing these results led to design and code changes to reduce duplicate message load in drivers and plans for a refactoring of message routing methods to be more efficient. Detailed message counts and timing results from the testing can be found at https://docs.google.com/spreadsheets/d/1iWc9K8Bgn_fqgmNNjw_F10xF867ucm9vpaFOVQcsnRg/edit#gid=0

The development team also investigated rewriting the message bus in Cython (a combination of C and Python) to improve performance. The initial development performed poorly when compared to the existing solution. The message bus (Pub/Sub over RPC) communication between agents via router results in numerous I/O calls. Since Cython does not help in speeding up I/O communication, the performance gained through Cython router was being overshadowed by the numerous I/O calls. The development then focused on reducing the number of I/O calls (i.e., essentially remove RPC from Pub/Sub and have a more direct send/receive communication between agents and router). This work is discussed in Section 6.0 (message bus router). Some of this work has already been released into VOLTTRON version 4.0. The remainder is expected to be completed in FY17 and is in testing for inclusion into the version 5.0 release.

4.3 Application Scalability in a Distributed Environment

Oak Ridge National Laboratory (ORNL) has done extensive testing in the scalability of applications in a distributed VOLTTRON environment. The ORNL testbed features numerous VOLTTRON nodes communicating with each other to achieve a control objective. The demonstration application chosen is a distributed signal regulation algorithm. The scalability experiment featured two distinct components:

- Distributed implementation of a control algorithm regulation that meets a regulation signal
- A discrete event simulation framework for scalable software-based testing of the control algorithm

The experiment has shown that application partitioning is possible in the VOLTTRON platform. Open questions still remain on how messaging latency and scalability of VOLTTRON central services may affect applications depending on this data.

More detail on the application and experiment methodology can be found in Appendix B.

4.4 Cloud Scaling Tests

To enable larger scale and repeatable testing, VOLTTRON is configured in a private computational cloud environment. Initial testing was performed with 1 to N buildings reporting to a central node and storing the data using a MongoHistorian. To facilitate automated testing, the example FakeDriver was used. As this produces more than is likely to ever occur in the real world, this should introduce scaling issues quickly. Each collector is a virtual machine running the MasterDriver agent, with the FakeDevice example. They are each running 500 devices with 18 points each. Each collector also runs a ForwardingHistorian and communicates with a “central” instance. The MongoDb is installed on its own machine.

In the simplest and most common topology of many collectors forwarding data through a single node to a MongoHistorian, each building introduces a 10-20 second lag in the time to store records into the database. The initial collection goes into the database in the regular 1-second window, as data points stack up, the time to store increases linearly across the number of points. For each synthetic building added to the topology, the total storage time increased. Initial analysis points to limitations on the throughput of a single MongoDB connection. This test data yields an upper bound for data points per sampling time window. This will inform deployments and be a baseline for possible improvements.

Cloud technology is most effective when resources can be stood up and torn down in a scripted automatic factor. Prior to VOLTTRON version 4.0, key management issues prevented straightforward building of collector nodes with full security enabled. Upgrades in version 4.0 allow better key management through the use of automated tools such as Heat or Ansible. This management-at-scale is now something the VOLTTRON team considers routinely when evaluating changes and new features.

Very preliminary data analytics were created to evaluate the health of data pipeline. Further detail on this can be found in Appendix B.

5.0 Scalability Improvements

VOLTTRON performance depends on the interaction of several components of the platform, the underlying hardware, and the operating system. This section addresses specific components in terms of identified issues and resolutions that support scalability. The community is encouraged to capture their own findings as they utilize the platform for their specific deployments. We look forward to working with the community to continue addressing scalability in general and with specific use cases.

5.1 Performance Improvements

Table 1 provides a summary of the scalability improvements, including issues addressed, resolutions implemented, and resulting effects.

Table 1. Summary of Scalability Improvements

Component	Issue	Resolution	Effects
Message Bus	Pub/Sub over RPC implementation imposed a performance penalty to ensure security	Refactor router and Pub/sub subsystem to be more efficient	Message bus performance at least 4x faster for various use cases. Approaches raw ZMQ performance.
Drivers	Published same data point 4 different ways	Option to limit publishes	Typical settings reduce traffic to, at most, ¼ the traffic of the original behavior. A reduction to 1/20 of the original volume is common.
Drivers	All data collected at once	Ability to stagger collection over a set period	Prevents overwhelming BACnet device (especially serial devices connected to router). Smooths out message load on message bus.
ActuatorAgent	Single command for single point each call	Takes a list of commands to issues at once	Allows for commanding 4,000 simulated devices from a single VOLTTRON instance.
Historian	Inefficient handling of metadata	Remove unneeded metadata handling	Improved performance by factor of 4.
Historian	ForwardHistorian only recommended mechanism for storing data remotely	New DataMover Historian designed and additional deployment recommendations	On secure networks, Historians can be configured to write to a network accessible database and increase throughput.
Historian	Backup cache grew unbounded and could run its system out of storage	Configurable maximum cache size	In cases where there is a long-term issue recording data, the backup cache will not grow infinitely.
Historian	Cache could be slow	Change to JSON library and change date formats	Cache performance increased approximately 10 times and date time parsing increased by 30 times.

Platform Registration	Error-prone and time-consuming process to register a platform with a management instance	VOLTTRON configuration utility needs minimal information then the act of registering is automatically handled by the platforms	Dramatically reduced the time to set up a platform from tens of minutes to almost no time.

The experiments and practical experience gained by the team led to a number of performance improvements that will be a part of VOLTTRON version 4.0 and highlighted areas to investigate for version 5.0.

5.1.1 Message Bus

The message bus is the essential component of the platform and its performance is paramount in ensuring an efficient and reliable platform. Numerous experiments and proof-of-concepts were performed to determine the best method of improving performance of the message bus without losing the gains in security accomplished in versions 3.0 and 4.0. These strategies included rewriting portions of the bus in C, separating out the component responsible for routing messages into its own process, and combining the router and Pub/Sub systems.

Ultimately, the router and Pub/Sub integration proved to be the correct approach as others did not show enough improvement for their development cost, introduced unreliability, or sacrificed security. Another deciding factor was that this improvement could be accomplished with no impact to the user base. Developers would not need to modify their agents and existing deployments could interact with new VOLTTRON instances running the improved message bus.

Table 2 provides a summary of performance testing showing the time from when data is published by a driver and received by a listening agent. Factors that impact time are the number of publishing devices, points per device, and the number of listeners.

Table 2. Summary of Performance Testing

	Total mean time at Listener end				
No. of devices	**Original RPC based Pub/Sub**	**With native ZMQ Pub/Sub (In sec)**	**Router and Pub/Sub integrated**	**No. of data points**	**No. of listeners**
50	0.13615	0.0119	0.05199	18	2
100	0.3533	0.01234	0.0404	18	2
500	1.467	0.0398	0.1972	18	2
1000	2.258	0.08439	0.2634	18	2
1500	3.6162	0.13425	0.2301	18	2
50	0.5088	0.0751	0.196	18	10
100	0.97167	0.16813	0.285	18	10
500	3.9861	0.651	1.1968	18	10
1000	10.2173	1.2188	2.4375	18	10

No. of devices	Total mean time at Listener end			No. of data points	No. of listeners
	Original RPC based Pub/Sub	With native ZMQ Pub/Sub (In sec)	Router and Pub/Sub integrated		
1500	16.2269	2.704	3.3629	18	10
2000	19.24	4.7028	5.436		10

5.1.2 Driver

VOLTTRON drivers are the major contributor to traffic on the message bus. Any improvements or refinement of their publishing behavior noticeably impacts the performance of the platform overall.

VOLTTRON drivers, by default, publish the same data four different ways: depth first single point, breadth first single point, depth first all points for a device, and breadth first all points for a device. This provides maximum flexibility for potential subscribers by allowing them to subscribe to single points or all points for a device and by the full path or start with points of a certain type. In actual use, most agents subscribe to the depth first all and pull out the points they need. With this behavior in mind, the types of publishes for drivers is now configurable. Narrowing publishes down to the single depth first all for a device reduces the amount of traffic by half at least and much more in instances of devices with a high number of points (which would no longer be published singly). This allows for the publishing of more individual devices and reduces unneeded messages being published.

Another improvement made to the VOLTTRON drivers to address scalability is the ability to stagger data collection. Instead of collecting from hundreds of points at one time, the collection can be spread across a configured time period. This prevents flooding both the VOLTTRON message bus and the underlying interaction with building controllers or devices. During a deployment where 9,000 points were being collected at once, both these errors appeared. Collection was enabled by spreading the collection of the points over a minute.

5.1.3 Actuator

The ActuatorAgent serves as a gatekeeper for access to devices being controlled by VOLTTRON. It maintains a schedule of device control to ensure that multiple agents cannot send control commands to the same device at the same time.

Scalability issues with the Actuator were identified while supporting an experiment involving agents sending control commands to thousands of simulated devices at the same time. The Actuator issued commands singly in serial that was sufficient for previous use cases where control commands were issued infrequently. The Actuator was refactored to allow multiple commands to be sent at a time and now supports sending more than 4,000 commands at a time. In the future, the MasterDriver and Actuator could be combined to further enhance scalability in this area.

5.1.4 Historian

The Historian subscribes to a set of base topics and records this data to a storage solution as configured. This storage solution could be a file, database, or forwarding to another VOLTTRON instance. In cases where the Historian fails to record to the ultimate destination, data is stored in a backup cache until the recording resumes.

During high-volume testing of the platform, two concerns became apparent with the Historian: 1) inefficient handling of metadata was slowing recording of data to unacceptable levels and 2) the backup cache could be a concern on small boards that experience long-term connection issues.

Removing the unnecessary metadata improved performance by a factor of four and allowed the lower-end boards to keep up with the volume of data in the testing. The other issue has handled by the addition of a configurable maximum size of the cache. Administrators can set this maximum to ensure their boards do not run out of storage space.

The backup cache was also transitioned to a different JSON processing library. This provided significant speed improvements. Retaining the floating point precision in the messages yielded an approximate 10 times increase. As the date time parsing was consuming significant time, switching to a more well-defined format in the database and replacing the parsing function yielded a parsing speed increase of about 30 times.

5.1.5 Profiling

Agents that provide services to the platform (Historians, drivers, platform agent, etc.) are undergoing profiling tests to identify inefficiencies.

In a large-scale installation, the VOLTTRON Central (VC) Platform Agent (VCP) was taking the majority of CPU resources. Profiling revealed inefficiencies in the way the VCP handled status reports for the agents that were operating on its platform. Improvements are being made to handle these reports more efficiently.

5.1.6 Registering Platforms

While not a traditional “scalability” issue, the manual process for registering platforms with the VC Management UI would have hindered a large-scale deployment of VOLTTRON platforms. This is an example of where high security leads to decreased usability as the administrator was required to manually complete several steps to properly configure security on VC and the remote platform and complete an error-prone action in the VC UI.

This was addressed by enabling the auto-registration of platforms with a controlling VC Management instance. The remote instance can be pre-configured to look to a specific VC instance for management. The necessary security setup is handled by a VOLTTRON configuration command that greatly simplifies the process and removes the need for administrators to hand edit security settings and constructing a long URL containing security keys. This decreases the time for the whole process from tens of minutes to a single minute to use the configuration command or no time for a pre-provisioned instance.

6.0 Upcoming Changes

Although some scalability bottlenecks were fixed in the normal development schedule, some are more complex and require greater evaluation before they can be released. They must maintain existing functionality and security while improving their performance issue.

6.1 Message Bus Router

The VOLTTRON development team is exploring multiple ways to speed up the message bus without sacrificing security. The message bus router is the primary candidate for performance improvement and the team is looking at the effects of rewriting this component, separating it out as another process, and replacing it with another option.

Two targets under current development include:

- RPC calls directly between VOLTTRON platforms
  - In the current VOLTTRON setup, if an agent on one platform wants to make a RPC method call to an agent on remote platform, it has to make an explicit connection to the target platform to make the RPC call. Instead, this change will allow the VIP routers of each platform to make the connection and manage the RPC communication internally. This will reduce the burden on the agents and enable a more seamless RPC communication between agents on different platforms.
- Pub/Sub between remote platforms without using a ForwardHistorian.
  - The goal of this task is to improve the current setup of having a forward Historian to forward local Pub/Sub messages to remote platforms. Instead, routers in each platform will be responsible for maintaining inter-platform connections, inter-platform message subscriptions, and routing of messages across platforms.

These improvements will reduce the burden of routing messages between remote platforms. It will also reduce the complexity to the deployer. Work is still ongoing to address scaling considerations such as platform discovery, key distribution, and maintaining topic subscriptions.

The results of these investigations are informing performance enhancement work in FY17.

6.2 DataMover

The ForwardHistorian was originally created to forward data from one VOLTTRON platform to another and have the data appear as quickly as possible. Over time it has been installed in most deployments to move data from one machine to another. In many cases, no use case exists for the data to appear on the destination as if it was locally collected. Using this agent causes each message to be processed multiple times through the Historian/message bus processing. This is significant overhead when live data is not required.

The agent that will address this issue is the upcoming DataMover. The DataMover agent will process data from the collection platform directly to the remote Historian, skipping the local message bus and local Historian processing. Instead of forwarding individual messages immediately, data can be collected into batches for more efficient transmission. These batches can then be sent to a corresponding agent on the receiving platform, which immediately writes to the database instead of republishing on the message bus. This approach has several benefits including instant feedback on the success or failure of the attempted write. This increases data assurance over the Forwarder, which cannot verify that the ultimate write to the database succeeded. In cases where data still needs to appear live on the receiving platform, the batch can be broken up and republished, although there will be some delay over live publishes from a ForwardHistorian.

7.0 Conclusion and Deployment Recommendations

VOLTTRON scalability investigations will continue throughout the lifetime of the project. In FY16 many practical lessons were learned and applied to the platform. In FY17 these will be enhanced with more formal testing of tens, hundreds, and thousands of platforms working together to implement a distributed energy efficiency application. Also strategies to address the manageability of such a large deployment will be explored in order to make recommendations for platform improvements. In addition to scalability of the core VOLTTRON, we look forward to additional explorations by the community of new storage, analytic and dashboard technology.

Table 3 is an overview of the interplay before hardware, platform roles, services, and applications. This table will be updated as changes to the platform and further investigation provide new information.

Table 3. Interplay of Hardware, Platform Roles, Services, and Applications

Example Hardware	Supported Platform Roles	VOLTTRON Services	Data Collection	Example Applications
Raspberry Pi, BeagleBone Black, ODROID, etc.	Edge Data Collection and small applications	ActuatorAgent, Drivers, ForwardHistorian, SQLite Historian, WeatherAgent, VCP	~1000 points/minute	Fault Detection, Whole Building Energy
Intel NUC, mini PCs[1]	Data Collection and application support Local Data Store	ActuatorAgent, Drivers, ForwardHistorian, SQLite Historian, MySQL Historian, WeatherAgent, VCP	~2000 points/minute (dependent on device communications capabilities)	Intelligent Load Control
High-capability virtual machine, high-end desktop	Management UI Centralized Data Store	VC Management UI, MongoDB Historian, VCP, Weather, Connection to external signals	Not applicable for this role	Optimizers, Analysis, Modeling

[1] http://www.tomshardware.com/reviews/mini-pc-round-up,3697.html

Appendix A
ORNL Distributed Application Scalability

Appendix A
ORNL Distributed Application Scalability

A.1 Distributed signal regulation agent

Buildings are currently large passive participants in the electric grid. They comprise a large thermal storage that can be used as an ancillary service to enhance the reliability of the power grid, which will be subject to more volatility and unpredictability from the integration of renewable energy resources. Two control strategies were explored that allow HVAC loads in commercial and residential buildings to provide frequency regulation services to the grid while maintaining occupants' comfort. The first control strategy is based on model predictive control acting on a variable air volume HVAC system that is available in large commercial buildings. The second strategy is a rule-based control that acts on numerous HVAC units, switching them on or off. These units are typically available in residential buildings and in many small to medium size commercial buildings.

The simulation experiments were implemented in Matlab and considered a set of 50 buildings that illustrated the proposed control coordination and control of HVAC systems for commercial and residential buildings. In this setup, a master control sent on/off signals to control HVAC units to meet a signal regulation objective. Ordinary differential equations were used to model the HVAC units and the master control would determine the number of units that must be switched on or off. The individual units were selected in a manner to minimize occupant discomfort.

The developed algorithm was ported to VOLTTRON and implanted with additional features to enable true multi-node control coordination. The implementation consisted of a MasterNode Agent and multiple ModelNode Agent instances. Upon booting up, a ModelNode instance would communicate with the MasterNode periodically updating the master of its temperature and HVAC state. The MasterNode maintains a list of active ModelNodes (a list of buildings being controlled). The first time a ModelNode communicates with the MasterNode is considered a new 'registration' and the ModelNode is added to the list of ModelNodes to be controlled. The MasterNode senses the regulation need every 10 minutes (to avoid short cycling equipment) and sends control signals to the ModelNodes.

The developed algorithm was tested on multiple instances of VOLTTRON running on Oracle VirtualBox as well as more extensively using the ADEVS (A Discrete Event Simulation Framework), as described later in this section.

While this demonstrated the scalability of typical algorithms in VOLTTRON, true scalability is demonstrated through an implementation of a distributed version of the signal regulation algorithm. The approach taken relies on accomplishing the three fundamental principles of distributed algorithms:

1. Distributed election of a leader
2. Arriving at a consensus
3. Achieving robustness

In a typical real-world control implementation, there is a high likelihood of some form of master node in a building. Grid-level needs should be met robustly across an integrated set of buildings, microgrids, and possibly across campuses. It is important to design distributed transactive algorithms that allow for flexibility within a customer's premises as well as integrate seamlessly across multiple customers. Figure A.1 illustrates the scenario. A campus may include multiple buildings, each controlled by a VOLTTRON node. The campus itself elects a main node that communicates with other peer instances in other campuses and participates in distributed decision making.

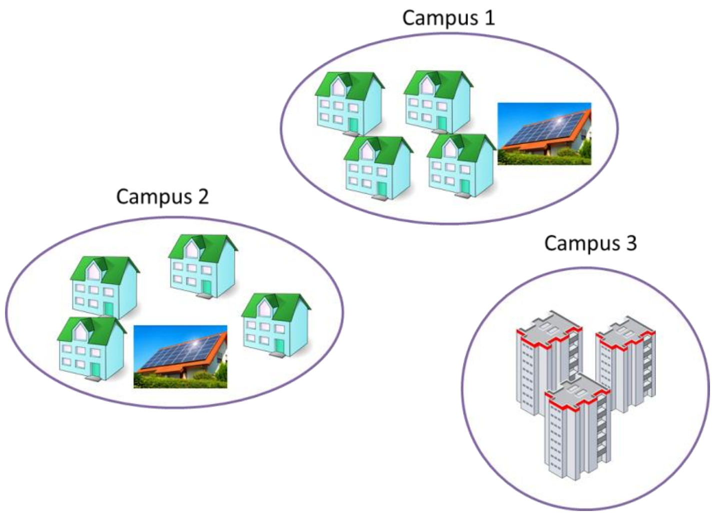

Figure A.1. VOLTTRON Campus Scenario.

As a result, the distributed agent is being developed to accomplish the following:

Within a campus:

i. Establish a virtual network (could use the existing wired or wireless network). In a simple implementation, a wireless ad-hoc network would suffice.
ii. Individual VOLTTRON nodes, upon boot up, will register in the network.
iii. Nodes determine (elect) a leader. Straight forward implementation uses the node with the smallest IP.
iv. Determine rules/circumstances that result in “leader is down; elect new leader.”
v. Each node communicates its regulation capacity/flexibility.
vi. Nodes communicate periodically with the leader.
vii. Leader runs optimization algorithms and communicates back decisions.
viii. Optionally, other nodes may have local logic and may send an acknowledgment of decision or an error code if it cannot comply.

Across campuses:

i. The leader nodes across campuses communicate.
ii. Each campus communicates load flexibility capacity.
iii. Regulation signal need is met in the ratios that each campus is able to contribute.

A.2 Testbed for scalability testing

Relatively slow evolution of “smart” energy and building technologies is due in part to the considerable gap between building affordable, highly reliable software, and the continuing reliance of the energy and buildings industry on prototype-based engineering. Indeed, all industries that now use software to perform major tasks abandoned prototype-based engineering in favor of model-based engineering. Model-based engineering is practiced because it is impossible to adequately test complex software by examining its operation in a physical prototype of the system being monitored and controlled. Adequate test coverage can require millions of test cases. As in the case of the distributed signal regulation agent, realistic testing poses a major challenge.

Model-based engineering addresses these problems by allowing for relatively inexpensive “virtual” prototypes that operate in simulation. Our simulation technology supports two parts to address this need:

- The continuous dynamics of the physical systems being monitored and controlled
- The discrete event dynamics of the computing and communications elements responsible for the monitoring and control.

Physical system models developed in MODELICA are transformed into Functional Mock-up Units (FMU) and these units are encapsulated in ADEVS. The QEMU computer system emulators are leveraged as components in ADEVS by wrapping QEMU system images running VOLTTRON, which incorporate the control system as it is expected to be deployed in the field. Modifications to QEMU enable event scheduling of the emulators’ execution in an order dictated by their current position in virtual time (simulation time). ADEVS handles events and communications between QEMU and FMU instances and evolves FMU systems over time by user-defined numerical solvers.

A key technical achievement in creating the simulation testbed was creating a very small footprint system image running VOLTTRON. Significant effort went into its creation, including dynamic repartitioning of a tinycore Linux installation, compiling required compilers in tinycore to create Python modules, and modifying the system’s library loading routines. These efforts resulted in a system image that less than 200MB and can run in as little as 256 MB RAM, while enabling full networking, serial ports, and additional peripherals for true controls emulation.

Our demonstration is illustrated in Figure A.2 orchestrating multiple buildings to meet signal regulation needs using VOLTTRON. Figure A.3 shows how the temperature in each building is controlled by aforementioned VOLTTRON agents. Five computers (incorporating the master node and four model nodes) and four building models are simulated allocating 500 MB memory for each building. The real running time of a 12-hour simulated test was approximately 60 minutes. This demonstration shows how a testbed for more complex multi-building (community level) and multi-layer control algorithms can be realized with the new simulation technology.

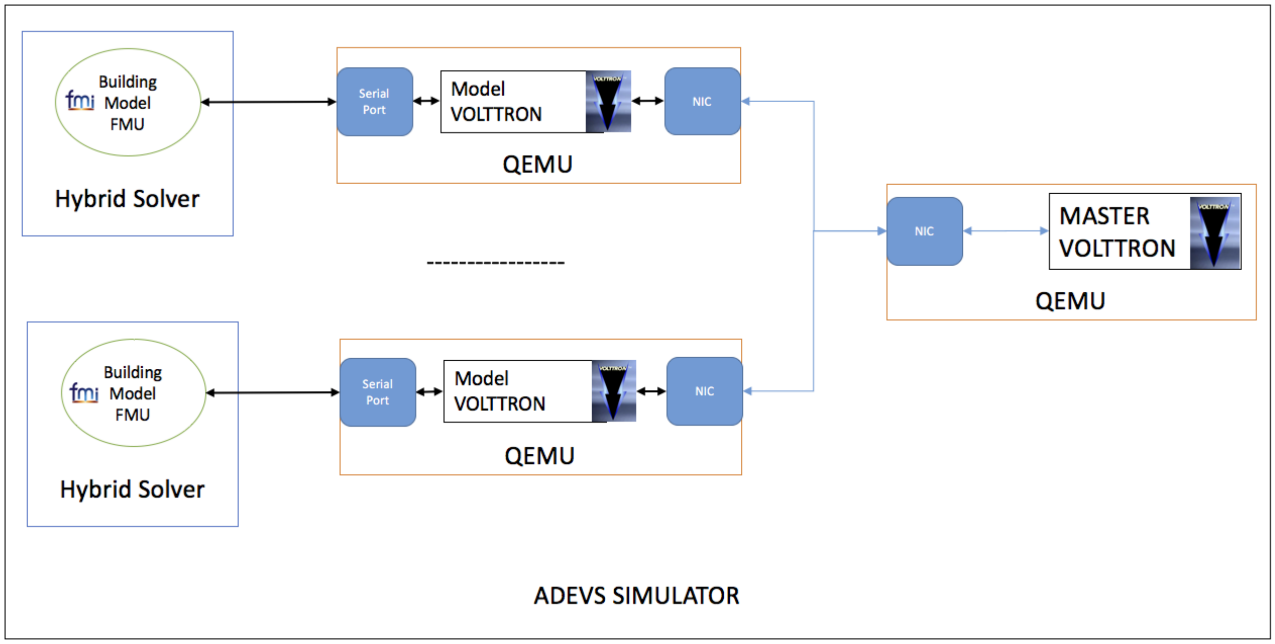

Figure A.2. Couplings of the components. Each model node represents a building and the master node monitors states of all model nodes and evaluates decisions. NIC is network card.

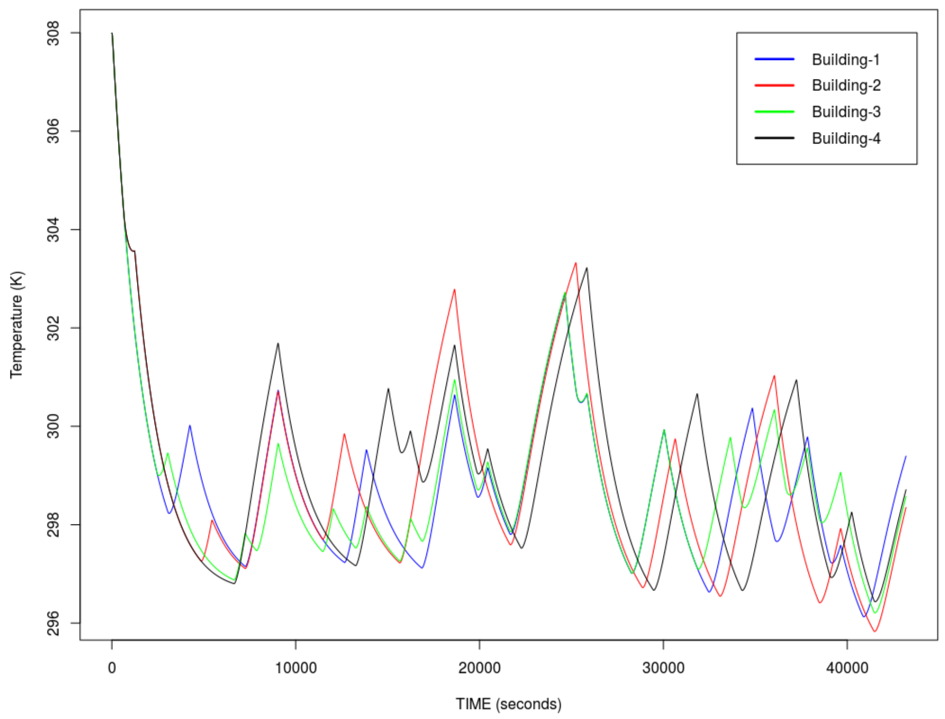

Figure A.3. Room temperatures of different buildings evolve around the set point temperature over time based on regulation signal.

Appendix B
Cloud Scaling Testing

Appendix B
Cloud Scaling Testing

The goal of using a cloud environment to perform scale testing for VOLTTRON include the ability to:

- Establish multiple networks and create different installation topologies
- Create and destroy installations in an automated and reproducible manner
- Provide re-creatable installations to test installation and configuration using technology such as Docker and Ansible
- Test new releases or client topologies

Testing priorities focused on key areas and elements not covered by practical deployments or developer focused testing. The most common topology, using data collectors and ForwardHistorian agents with MongoDB, was tested in the cloud using automated creation techniques.

The cloud infrastructure used for this experiment is the PNNL Institutional Research Cloud running OpenStack (version Mitaka). It is backed by a 40G network and shared cloud-attached storage.

Initial experiments yielded the following insights:

- Unnecessary agents installed on a VOLTTRON node can impact performance.
  - We had a platform agent on a machine that did not need it. The agent was constantly performing work it had no need to do. Once the agent was removed, the CPU and memory load on the machine flattened out.
- In using automated tools to deploy our machines, we did get the benefit of the interactive VOLTTRON configuration tool. This led to our collector machines publishing significantly more data than was necessary. This, in turn, led to the Historian being unable to process all the data without falling behind. We saw this performance with one building with 500 fake devices and it was magnified significantly as each building was added. Turning off the extra data publishes reduced the load and the Historian was able to keep up.
- Key management must be considered in an automated deployment. It is not difficult to pass server keys down to new machines, but providing server keys from one collector to another, especially if they are being created in the same script, can be challenging. Improvements were made in version 4.0 to allow look ups of some keys, but this must be considered in a large automated deployment.
- Running the VC agent on the central processing machine has a non-zero but manageable footprint. Experimental results put each 500-device building at about a 200-300 MB footprint.
- Poor naming choices of devices and buildings will not affect VOLTTRON's ability to function, but does impact the ease of analytics.
- Lack of closely synchronized time settings can make performing health analytics significantly more challenging. In a large-scale deployment, it is not practical to log into each box to update its time and tools like Ansible should be considered. Analytics producers must also be sensitive to this phenomena.

Measurements of the processing pipeline included tracking the average load average and memory use of the central node and time to storage metrics as each building was added. Due to the slight time setting configurations, anything within 1-3 seconds from “collection” to storage in the database was considered to be not impacted by new scale.

As each building was added to the topology, the time between generate and database storage was tracked. Each synthetic building running the FakeDriver added an average of an extra 10 seconds of maximum time to store. Each batch of records began taking 1 second to store and progressed in a linear fashion to taking 9 seconds initially to almost a minute with four buildings.

Initial analysis suggests this is related to the throughput performance of the single Python MongoDB connection. This yields a recommended upper boundary for number of points stored per time unit for any single concrete MongoHistorian. Estimated storage times are listed in Table B.1.

Table B.1. Storage Times

Number of Buildings	Devices	Points	Starting Storage Time (sec)	Ending Storage Time (sec)
1	500	18	1	9
2	500	18	1	25
3	500	18	1	45
4	500	18	1	60