EDGEXFOUNDRY™

EdgeX Foundry Performance Report

Fuji Release

01 March 2020

Fuji (version 1.1) marks the 5th community release of EdgeX Foundry. It was formally released in November 2019. In addition to a number of new features (cataloged here), Fuji also represented efforts to help improve the overall performance of the platform (improving the performance quality of the code base, providing more asynchronous behavior, etc.).

This report is intended to be a regular part of each release going forward; providing EdgeX users with information they can use to guide their solution development and deployment while also assisting the EdgeX development community to target future performance improvements (and testing needs).

Raw data underlying this report is provided here.

www.edgexfoundry.org

LF EDGE

THE LINUX FOUNDATION

Deployment Options

The EdgeX Foundry platform is comprised of a collection of microservices. While a number of services are used in almost all deployments of the platform, the specific collection of services used are dependent on the use case, solution architecture, and user discretion.

EdgeX provides flexibility in its architecture that offers many options for deployment and use. A user of the EdgeX platform could deploy using the central logging service or have each service log to a local file (bypassing the need for the logging service). A user could deploy with Redis or Mongo databases. In very austere environments, a deployment many only include a device service and a few core services.

Therefore, this report provides resource usage based on the "general" deployment and "minimal deployment". The general deployment includes all those services used together as envisioned by the original architects of the platform and inclusive of one example device service. Specifically, the general deployment includes the following services and infrastructure:

Service	EdgeX Service Name	EdgeX Required or Optional
Consul	edgex-core-consul	Optional – configuration can be obtained from local file
Database	edgex-redis	Redis in this configuration. Mongo is the alternative.
Core Data	edgex-core-data	Required
Core Metadata	edgex-core-metadata	Required
Core Command	edgex-core-command	Optional – only required in actuation use cases
Support Logging	edgex-support-logging	Optional – logging can also be done to local file
Support Notifications	edgex-support-notifications	Optional – services do not have to notify via central service
Support Scheduler	edgex-support-scheduler	Optional – services do not have to use a central scheduler (Device Services embed their own)
Export Client	edgex-export-client	Optional – clients can register for data of interest
Export Distro	edgex-export-distro	Optional – delivers data to registered client
Device Virtual	edgex-device-virtual	Optional – any number of device services can be connected. Device Virtual serves as the representative example

***Note**: The current report does not include the rules engine as part of the performance metrics exploration. The Fuji release uses Drools as the rules engine. Drools is a Java enterprise rules engine. As such, it is quite large and slow and not entirely suited for edge use cases. Its inclusion in any metrics would have severely skewed the statistics reported here.The Geneva release will include a Go based replacement and provide a better indication of command actuation and other related statistics.*

The minimal deployment includes a subset of services that support a basic deployment:

Service	EdgeX Service Name	EdgeX Required or Optional
Database	edgex-redis	A database is required and Redis is much smaller in its typical resource usage
Core Data	edgex-core-data	Required
Core Metadata	edgex-core-metadata	Required
Device Virtual	edgex-device-virtual	A deployment would include at least one device service and the virtual device service serves as the representative example

Service Executable and Image Size (Footprint)

EdgeX microservices are compiled into an executable (typically from either a Go or C code base). That executable has a size – otherwise known as footprint – as it sits on disk. Additionally, the executable, along with a base operating system, any needed configuration and supporting infrastructure, is “containerized” for use. Specifically, EdgeX microservices are built into Docker containers for ease of deployment and orchestration with other EdgeX services. These containers also have a footprint size (typically much larger as it incorporates a base OS, infrastructure, configuration, etc.) as it sits on disk.

Depending on the use case, a user of EdgeX may choose to use EdgeX in either containerized or non-containerized form (that is use the executable without using Docker). The raw executables could also be used to deploy/orchestrate EdgeX using an alternative mechanism.

Both the non-containerized (“Executable footprint”) and containerized footprint ("Image footprint") are reported here for platform user consideration. Because the compile and containerization differ slightly on Intel and Arm based-platforms, statistics for both are provide.

In general, the **footprint of EdgeX is around 330MB containerized** and just under 300MB non-containerized but inclusive of the infrastructure elements (database and configuration/registration service). EdgeX services, by themselves without infrastructure are around 180MB in size.

General Deployment Footprint

Footprint in MB	Intel/AMD	Intel/AMD	ARM	ARM
Micro service	Container	Executable	Container	Executable
edgex-core-consul	106.99	98.3	111.71	
edgex-redis	29.33	9.9	29.09	
edgex-core-data	28.46	20.61	27.37	19.81
edgex-core-metadata	21.74	21.73	21.11	21.10
edgex-core-command	20.47	20.46	19.84	19.84
edgex-support-logging	19.45	18.70	19.00	18.20
edgex-support-notifications	22.21	20.62	21.63	19.99
edgex-support-scheduler	20.66	20.65	20.00	19.99
edgex-export-client	20.39	20.38	19.83	19.82
edgex-export-distro	26.18	18.33	25.24	17.67
edgex-device-virtual	20.19	20.16	19.12	19.09
TOTAL	**336.07**	**181.64**	**333.94**	**175.51**

__Note__: Executable sizes (shown in blue background) for Consul and Redis obtained from inspection of respective Linux containers for binary images (consul in /bin and redis-server in /usr/local/bin). Executable sizes of Consul and Redis are not included in the TOTAL for executables at the bottom of the column.

Minimal Deployment Footprint

Footprint in MB	Intel/AMD	Intel/AMD	ARM	ARM
Micro service	Container	Executable	Container	Executable
edgex-redis	29.33	9.9	29.09	
edgex-core-data	28.46	20.61	27.37	19.81
edgex-core-metadata	21.74	21.73	21.11	21.10
edgex-device-virtual	20.19	20.16	19.12	19.09
TOTAL	**99.72**	**62.50**	**96.69**	**60.00**

CPU Usage

The CPU usage of each container was measured on startup of the service. It is measured as a percentage of CPU available as reported by the Docker Engine. Because the CPU characteristics of the Intel and ARM platforms vary, the percentage of CPU usage can differ widely. **CPU usage on an Intel Atom Processor** (E3805 1.33GHz 1MB L2 cache) **is around 25%** - inclusive of infrastructure elements (database, configuration/registry, etc.). On an ARM Cortex A processor (1.4GHz 64-bit quad-core Broadcom Arm Cortex A53), CPU usage sits near 60% - again inclusive of infrastructure elements.

General Deployment CPU Usage

CPU Usage % at Startup Micro service	Intel/AMD Container	ARM Container
edgex-core-consul	13.49	46.00
edgex-core-redis	0.47	0.63
edgex-core-data	1.98	1.66
edgex-core-metadata	1.87	1.49
edgex-core-command	1.72	1.59
edgex-support-logging	1.71	1.77
edgex-support-notifications	1.64	1.98
edgex-support-scheduler	1.87	1.76
edgex-support-client	1.59	1.53
edgex-support-distro	0.06	0.20
edgex-support-virtual	0.00	0.34
TOTAL	**26.4**	**58.95**

Memory Usage

Again, the memory usage of each container was measured on startup of the service. It is measured in MB used as reported by the Docker Engine. **Memory usage**, in a containerized environment, **sits around 60MB for Intel and 75MB for ARM platforms** (inclusive of infrastructure elements).

Minimal Deployment Memory Usage

Memory Usage at Startup in MB Micro service	Intel/AMD Container	ARM Container
edgex-redis	19.99	27.44
edgex-core-data	4.94	5.30
edgex-core-metadata	5.55	6.74
edgex-device-virtual	6.56	7.14
TOTAL	**37.04**	**46.62**

Minimal Deployment CPU Usage

CPU Usage % at Startup Micro service	Intel/AMD Container	ARM Container
edgex-redis	0.47	0.63
edgex-core-data	1.98	1.66
edgex-core-metadata	1.87	1.49
edgex-device-virtual	0	0.34
TOTAL	**4.32**	**4.12**

***Future Consideration**: In general, the services consume a lot of CPU as they startup and so the measure of usage at startup can be a good upper bound for many services. However, future performance tests will also test the CPU usage periodically throughout a period of time and at peak sensor data ingestion and device actuation times.*

General Deployment Memory Usage

Memory Usage at Startup in MB Micro service	Intel/AMD Container	ARM Container
edgex-core-consul	19.99	27.44
edgex-core-redis	2.13	2.62
edgex-core-data	4.94	5.30
edgex-core-metadata	5.55	6.74
edgex-core-command	3.02	3.88
edgex-support-logging	3.80	4.83
edgex-support-notifications	3.62	4.69
edgex-support-scheduler	3.39	4.16
edgex-support-client	3.10	3.96
edgex-support-distro	3.65	4.24
edgex-support-virtual	6.56	7.14
TOTAL	**59.75**	**75.00**

***Future Consideration**: In general, the services consume various amounts of memory during operation. Future performance tests will also explore the memory usage at peak usage times such as during sensor data ingestion and device actuation.*

Operational Latency

Startup and Ping Operations

Currently, the startup (a.k.a. bootstrap) time and response to the HTTP ping request are the only two operational performance measures collected across all services. Additionally, the time to export the data to an endpoint is also measured.

The time to start all EdgeX services is less than 4 seconds on an Intel platform – including the time it takes to create the container (although not the time it takes to pull the container from Docker Hub). Startup time on ARM platforms takes a bit longer; often a second more. The ping of any service typically takes less than 15 milliseconds, regardless of platform.

The time to pull the Docker container image, create the container and start all of the EdgeX containers is significantly higher – especially on more resource constrained platforms. The time to pull, create and start EdgeX containers on the 2GB (RAM) Intel platform is just under a minute. The time to pull, create and start EdgeX containers on a 1GB (RAM) ARM platform is a just over a minute and a half.

Startup Time: Micro service	Intel/AMD Executable Startup (ms)	Intel/AMD Container Startup Time (s)	Intel/AMD Ping (ms)	ARM Executable Startup (ms)	ARM Container Startup time (ms)	ARM Ping (ms)
edgex-core-data	838.18	41.74067712	19.43	678.75068	73.91606617	23.32
edgex-core-metadata	383.33	44.56999302	11.58	2699.670399	71.84158325	19.105
edgex-core-command	256.94	51.48968911	10.97	297.965289	81.10887623	9.449
edgex-support-logging	118.61	32.67362809	13.68	2091.646797	56.17503428	10.106
edgex-support-notifications	177.50	43.3496232	15.12	414.655227	73.98805928	13.629
edgex-support-scheduler	244.80	50.37671804	11.31	1957.121666	83.89384127	11.338
edgex-export-client	242.41	47.67028117	9.85	633.752687	84.20551419	12.188
edgex-export-distro	199.73	54.43341112	12.97	689.506076	97.24861026	10.656
edgex-export-virtual	1340.80	59.39391804	12.89	4287.060543	100.3819423	11.76
TOTAL	**3802.28**	**59.39391804**	**117.80**	**13750.12936**	**100.3819423**	121.56

***Note**: additional measures were taken to pull and to start the micro service containers in alternative scenarios and modes. These measures are highly dependent on Docker Hub and Docker Engine and are not summarized in this report but are available in the raw data sets provide here.*

Future considerations: beyond startup up and general response, understanding the performance of data as it moves through EdgeX services is a critical factor in understanding the platform going forward. In future releases, the community will begin to look at performance involving many services – that is exploring performance associated to the interoperability of services (ex: time to collect sensor data and cause actuation to another device).

Data Export

The time it takes to send EdgeX data from the "south side" to the "north side" has improved for Edinburgh. This is inclusive of the time it takes the virtual device service to create the Event/Reading, send it to (and through) core data, and have the export service read and prepare the data for a north bound system. The Edinburgh release uses Export Services. Future releases will be based on the new Application Services.

Export includes extracting the event from the Core Data supplied event message topic, performing the export operations (including filtering by device name), sending the data to its designated endpoint, and marking the record as "pushed" in core data.

Intel / AMD	ARM
16.69 ms	18.00 ms

User Guidance

The deployment of an edge solution is dependent on many variables including: device connectivity and volume of data processed from the sensors/devices, how often data is sensed, how quickly actuation must be achieved, amount of data sent to north side systems. As the EdgeX community continues to test its product, it will continue to make recommendations on required resources to execute EdgeX based solutions under various loads and to refine any system requirements. At this time, the following minimal requirements are defined based on existing testing.

Hardware Recommendations

Memory: At least 1 GB of RAM

CPU: at least 30% free CPU on Intel platforms and 50% free on ARM platforms

Disk space: dependent on the number of EdgeX services running, the amount of data captured and the length that the data is retained (including sensor data, service logs, etc.), it is advised that the platform have at least 500MB of free space available.

Methodology

The following notes describe how the performance data found in this report was captured. The services were run with default configuration. The Device Virtual was allowed to run and produce sensor data in its default manner (generating data every 30 seconds for a variety of mimicked sensors).

The CPU and memory usage data was captured from Docker Engine via docker stats after successful start up of EdgeX using the release Docker Compose files. Export testing was run 15 times – that is generating 15 event/readings through core data and into export services via the device service.

Tests were run on two hardware platforms:

- Intel – Dell Gateway 3002 (Atom E3805 processor @1.33GHz with 2GB RAM, 8GB disk, running Ubuntu 18.04 LTS)
- ARM – Raspberry Pi 3 B+ (Broadcom BCM2837B0 Cortex A53 ARMv8 processor @1.4GHz with 1GB RAM, 16GB disk, and running Ubuntu 18.04 LTS)

EDGEX FOUNDRY™

EdgeX Fuji Release

The EdgeX v1.x series of release include important new features as well as improvements and hardening of the existing EdgeX functionality.

Highlights of the v1.1 Fuji release include:

- New and improved security services – fully integrated with existing micro services (API Gateway, secure storage)
- Application services and application functions SDK as full replacements for older export services (we expect to deprecate the export services with the next release)
- System management improvements include ability to set EdgeX configuration
- Improved testing and quality assurance procedures and tools
- Addition of an many more device services (to be released independently in mid-December with the device service SDK release).

What makes this release unique

- Over two years of combined intense development and customer testing
- Stable API baseline for the standardization of an Open IoT edge applications
- Cloud-agnostic Northbound communications with full SDK and support for AWS, Google Cloud and Microsoft Azure
- Multi-vendor Southbound connectivity options across many popular protocols plus SDKs for ease of development and integration of new protocols
- Improved Security and Management services
- Comprehensive testing, including performance and scalability testing
- High-quality user documentation
- Cleaner and more flexible code base
- A huge Global partner ecosystem offering a range of complementary products and services

Contact Us

For general information about Edge X, EdgeX Foundry, or membership inquiries, please email **info@lfedge.org**

Visit our website at

www.edgexfoundry.org

Have you got any questions?

This document was produced by IOTech Systems - a global IoT/edge software products company offering Edge Xpert, which is a commercially supported implementation of EdgeX Foundry.

If you have questions about this report or if you need more information on IOTech or Edge Xpert, email **info@iotechsys.com** or visit the website **www.iotechsys.com**

LF EDGE

THE LINUX FOUNDATION