SAND2022-12095C

Fallout

A Monitoring Infrastructure Supporting Informed System Acceptance

M. Showerman, E. Roman, J. Greenseid, T. Tucker, and J. Brandt

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc. for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Fallout Objectives and Approach

Goal: Minimize time spent in initial testing phase through quick identification of component “fallout”

Easily deployed end-to-end monitoring solution tailored to easy deployment on **any** minimally configured Linux system

- Factory testing
- Initial standup and acceptance testing

Analysis and visualization enables identification of outlier components e.g.,:

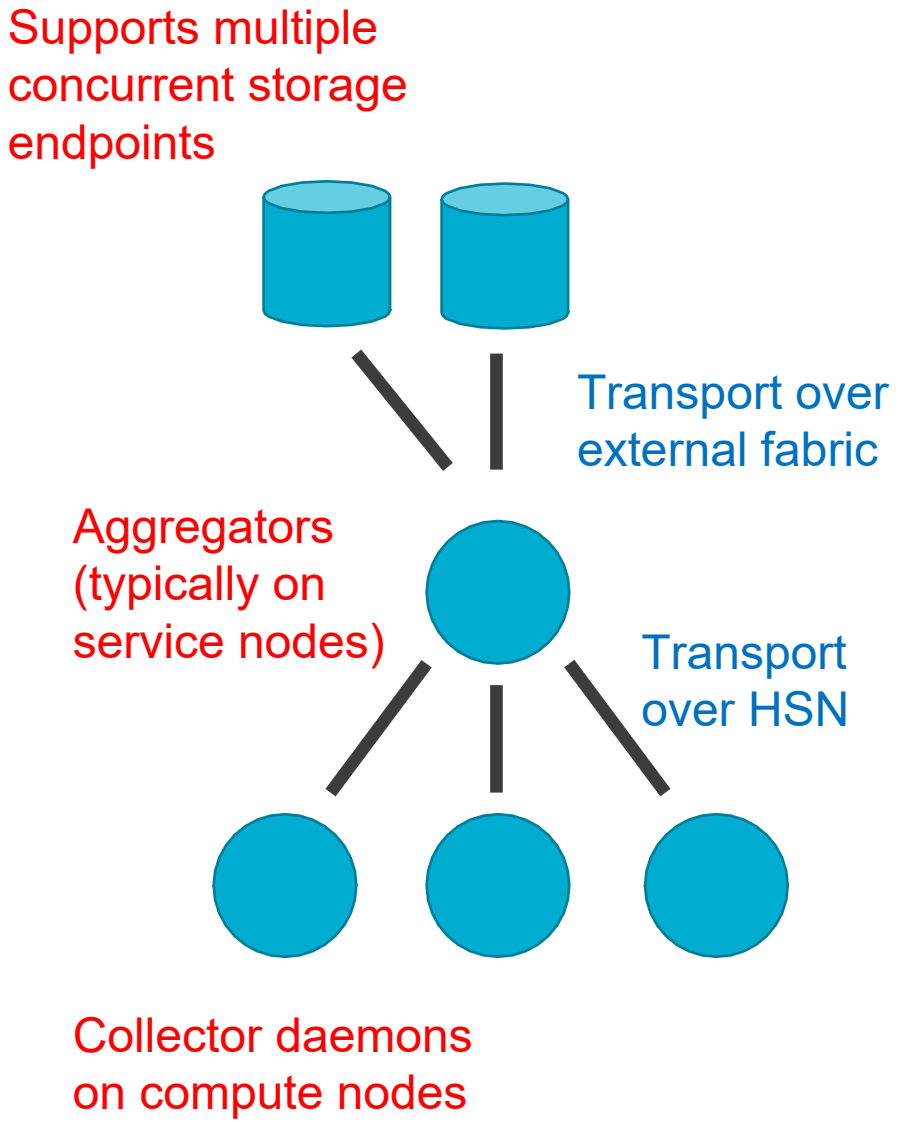
- Network links
- Processor thermals
- Memory Utilization
- CPU Utilization

Constraints during standup/factory testing

- Limited access to repositories satisfying external software dependencies
- Limited ability to modify boot images
- Limited/missing access to shared/remote storage
- Vendor wants minimal external influences to target machine operation
  - Don’t want unknown software possibly impeding acceptance
- Vendor monitoring infrastructure is largely unavailable and/or insufficient

LDMS: Brief background

- LDMS is a extremely lightweight monitoring system:
  - Flexible configuration with same base daemons everywhere with different plugins (e.g., different data samplers or stores)
  - Memory footprint is minimized through optimized data structures
  - Minimal network traffic through sending only the data values (contextual metadata only sent on change)
  - Efficient bi-directional transports (including RMDA) over mixed network architectures
  - Typical pull-based model for regular numerical data reduces the compute-node functionality and hence overhead
  - Push-based model enables sending of on-demand event data (including application data!)
  - Highly-scalable with no statistically significant impact on any large-scale system tested (NCSA Blue Waters, NERSC Cori)



Fallout Approach: Simple, Streamlined Standup and Configuration

Few components with minimal dependencies

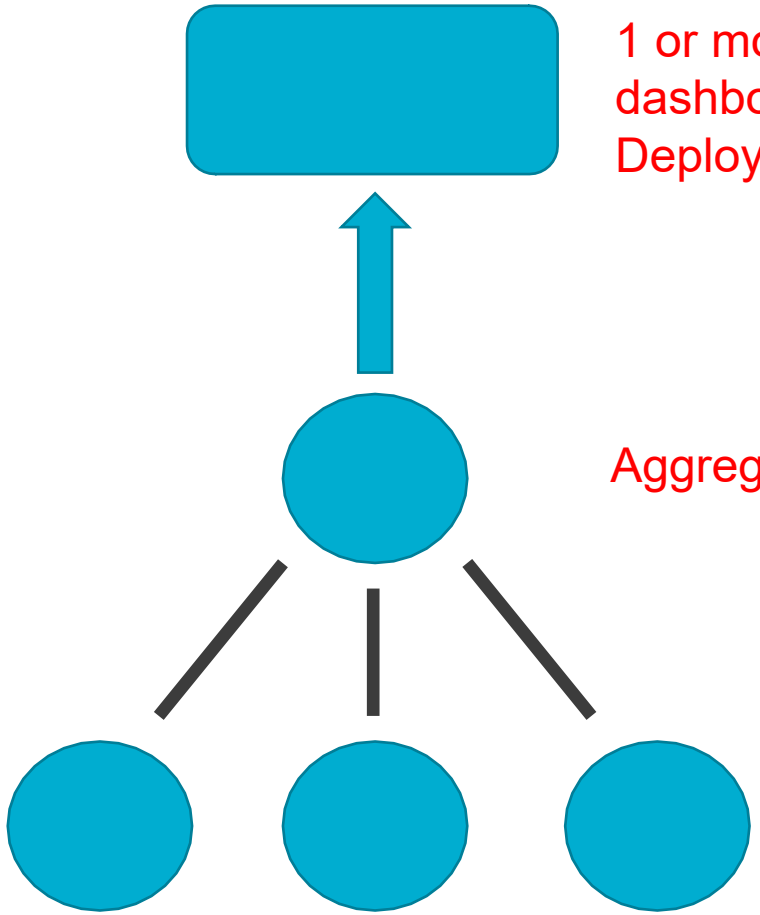
- Very few binaries (3) statically linked
  Ldmsd_standalone, ldms_ls (data query)**, ldmsctl** (dynamic configuration)
- Current dependencies
  - libc, libssl

Ldmsd_standalone sampler binary:

- ldmsd built as a statically linked program
  - Samplers:
    - meminfo, vmstat, loadavg, coretemp
  - Transports:
    - Sock transport
  - Authentication Plugins:
    - “none” authentication method

Initial configuration obtained from sampler.conf located in the same directory as the binary

Simple Topology



1 or more stores including database and visualization dashboards setup for statistical comparison and outlier detection
Deployed as Docker containers

Aggregators deployed as Docker containers

Samplers deployed on compute node bare metal via single scp

Sampler Nodes Assumptions

- Require access to hardware/kernel assets not easily available from a container
- Container support may not be available on compute nodes
- Need a very simple “install” process that places the monitoring software on the node without requiring updating the boot image
- **Minimal configuration beyond installing the application, i.e. no need to configure library paths, plugin paths, etc..**

Fallout Solution:

- Single application that has all dependencies statically linked
  - Configuration file required to be minimally modified to support system specifics
- No external dependencies
- ***Simply copy binary and configuration file to compute nodes and run!***

Keep Sampler Deployment Simple

Configuration = **./sampler.conf**

- Comes with a “standard” configuration but expected to be modified based on user needs
- Available samplers are currently “meminfo”, “vmstat”, “loadavg”, and “coretemps”
  - Assumption that relevant technology-specific samplers will be created prior to deployment and included in compiled-in set of samplers

Standard configuration:

Transport = sock

Listening port = 411

Authentication = none

Sampling period = 1 sec

Running Sampler = meminfo, vmstat, loadavg, coretemp

Set name = <hostname>/<sampler plugin name>

- Example: nid0001/meminfo

Keep Sampler Deployment Simple Cont'd.

- Put 'sampler.conf' in the same directory as '*ldmsd_standalone*'
- *ldmsd_standalone* starts with the standard configuration in sampler.conf:
  ./<path to ldmsd_standalone>/ldmsd_standalone

Two options to customize sampler's configuration:

- Modify sampler.conf and start ldmsd_standalone
- Point ldmsd_standalone to the user's configuration file:
  ./<path to ldmsd_standalone>/ldmsd_standalone -c <user configuration file>

Aggregator Nodes

- Implemented with a Docker container
- Gathers (Aggregates) data from Sampler Nodes
- Stores data locally in the container (SOS or CSV)
- Minimal configuration through docker run parameters
- No external dependencies
- Requires docker swarm + docker network
- ***If it will run docker, it will run Fallout!***

Visualization and Analysis

- Implemented with a Docker container
- No external dependencies
- Uses Grafana or Google Graphs for Visualization
- Aggregator and Viz/Analysis containers map common local storage volumes into the instance
- Suite of canned Analysis modules for outlier detection, rate computations, etc.

Fallout Current Status

Single binary image

- To build, configure with **--enable-standalone** flag

Analyses

- Python analysis methods query distributed database (DSOS, Vitess)

Visualizations

- Grafana
- Google Graphs

Deployment

- Bare metal install
- Container bring up
- Visualization of line plots and heat maps

Current Status Cont.

What the executable file includes:

- **zap_sock**
- **auth_none**
- **meminfo**
- **vmstat**
- **coretemp**
- **loadavg**

The standalone executable files are

- **ldmsd**
- **ldmsctl**
- **ldms_ls**

To build the standalone version, configure with **--enable-standalone** flag.

Simple Sampler Deployment

- Just scp ldmsd, sampler.conf, ldms_ls and ldmsctl to compute nodes
- Or, if there is a shared file system, just cp those files to a shared location

- Execution on compute nodes: simply call
  /path/to/single/ldmsd_standalone

Docker setup

- git clone https://github.com/ovis-hpc/ldms-containers/
- Initialize docker swarm on a barebone machine

  docker swarm init

- If the containers were to spread across multiple barebone machines, the other machines have to join the swarm

  docker swarm join –token KEY HOST:PORT

- Add `ldms` network, do the following on the leader machine
  - Edit **config.sh**, specify **`NET=ldms`** and **SUBNET** of your choice
  - **./network-create.sh**

- **docker pull ovishpc/ldms-agg**
- **docker pull ovishpc/ldms-ui**
- **docker pull ovishpc/ldms-grafana**

Aggregator Container Deployment

On a barebone machine that you want to deploy the container:

./ldms-agg/run.sh --name agg-11 --prdcr "nid{00001..20}" --mem 128M --offset 200000

- This creates a container named `agg-11` with 128 MB memory pool that collects LDMS sets from the ldmsd's on nid00001 – nid00020, with 200ms offset (default 1s interval). Note that `agg-11` does not have a store.

UI container deployment

On a barebone machine that you want to deploy the container:

./ldms-ui/run.sh --name ui --dsosd "agg-{11,12}"

- This creates a container named "ui" that connects to dsosds on agg-11 and agg-12 containers.
- The "ui" container runs the Django UI back-end that serves data points to Grafana.

Grafana Container Deployment

On a barebone machine that you want to deploy the container:

./ldms-grafana/run.sh --name grafana

- This creates a container named "grafana" that by default has the http://ui/grafana/ data source installed.
- The Grafana server inside the container serves the web app over a Unix domain socket to work around a firewall issue. The Unix domain socket is exposed to the bare-metal host at:
  - **"./ldms-grafana/sock/grafana/grafana.sock"**
- In order to get to the Grafana web app, we suggest to use SSH port forward as follows:
  - On laptop:

$ ssh bare-metal-host -L 127.0.0.1:3000:/ldms-containers/ldms-grafana/sock/grafana/grafana.sock

Getting to Grafana Web App

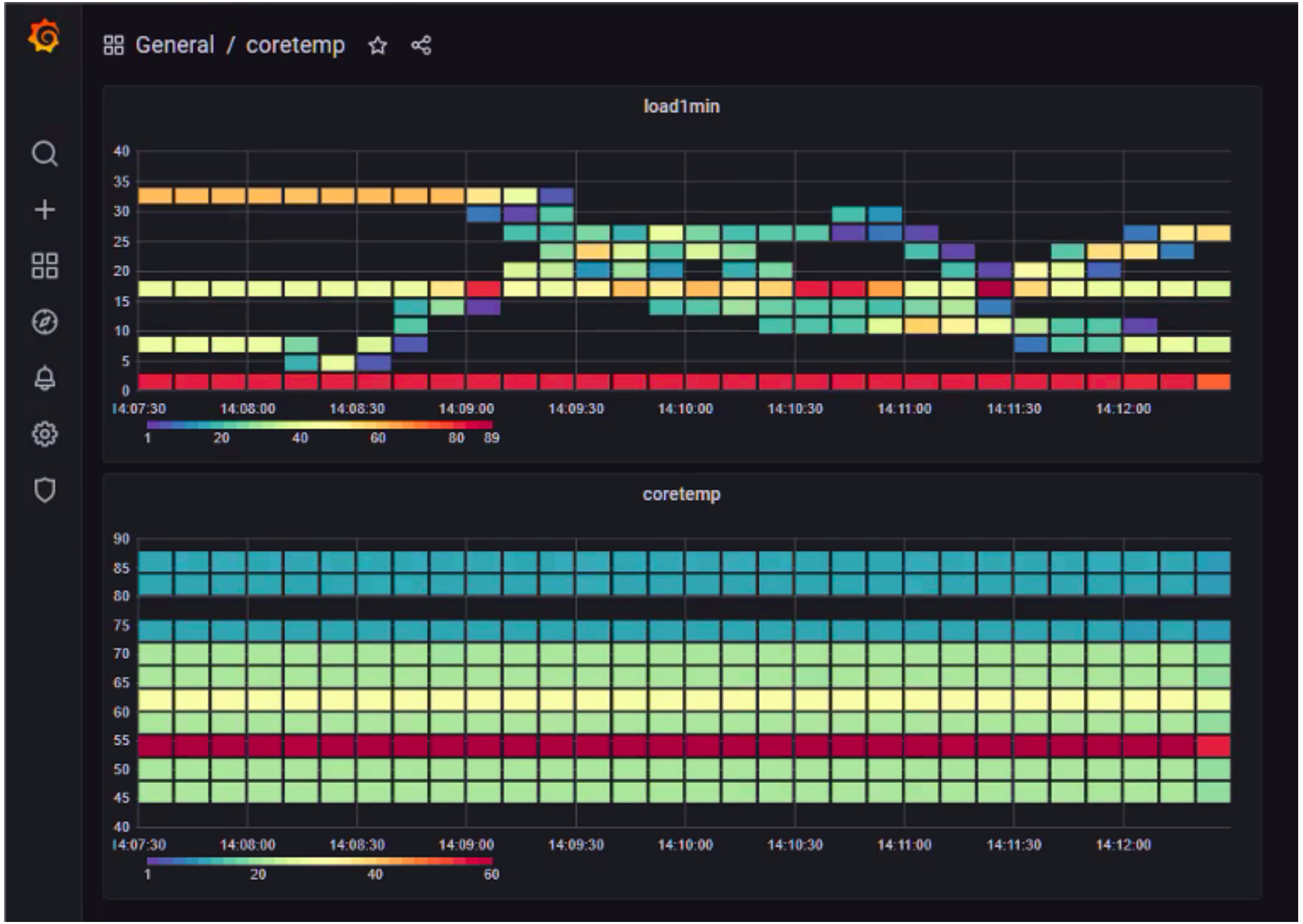
In order to get to the Grafana web app, we suggest using SSH port forwarding as follows:

- Suppose the ldms-containers project is checked out at **/ldms-containers**.
- On laptop:

$ ssh bare-metal-host -L 127.0.0.1:3000:/ldms-containers/ldms-grafana/sock/grafana/grafana.sock

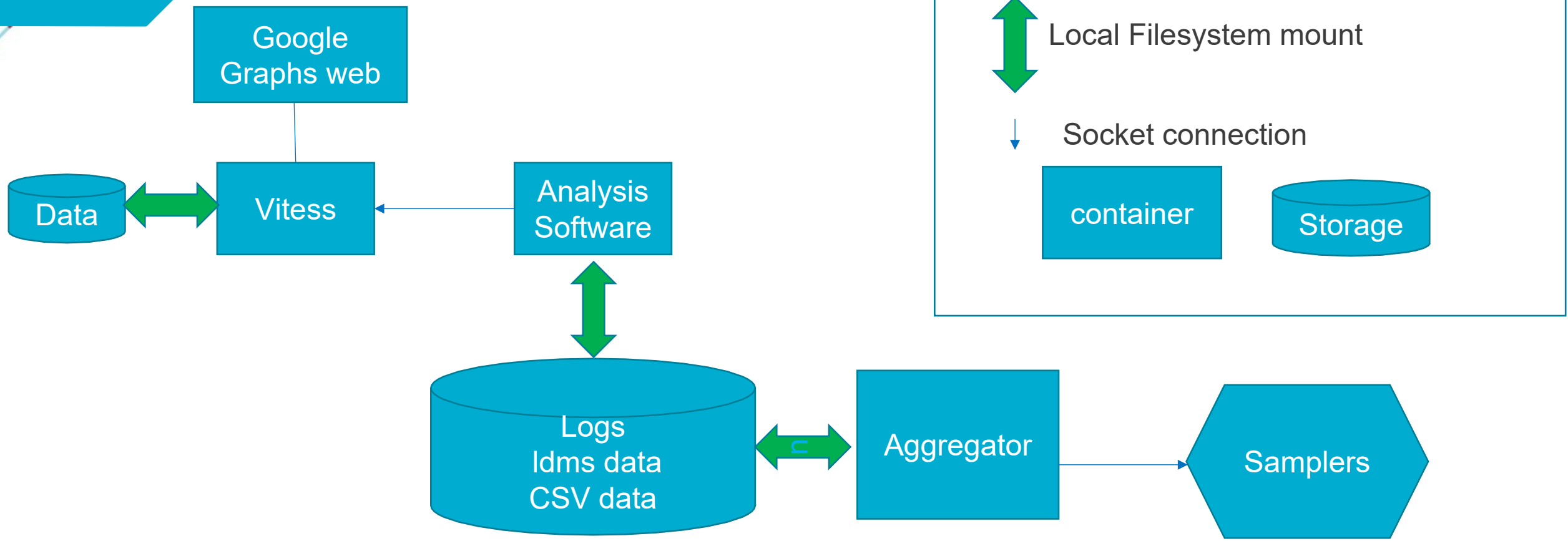
- Then, on a web browser the laptop, go to: **http://127.0.0.1:3000**

Loadavg, and Core Temperature Dashboards



Memory, Loadavg, and Core Temperature Dashboards

4 container Plan Single host



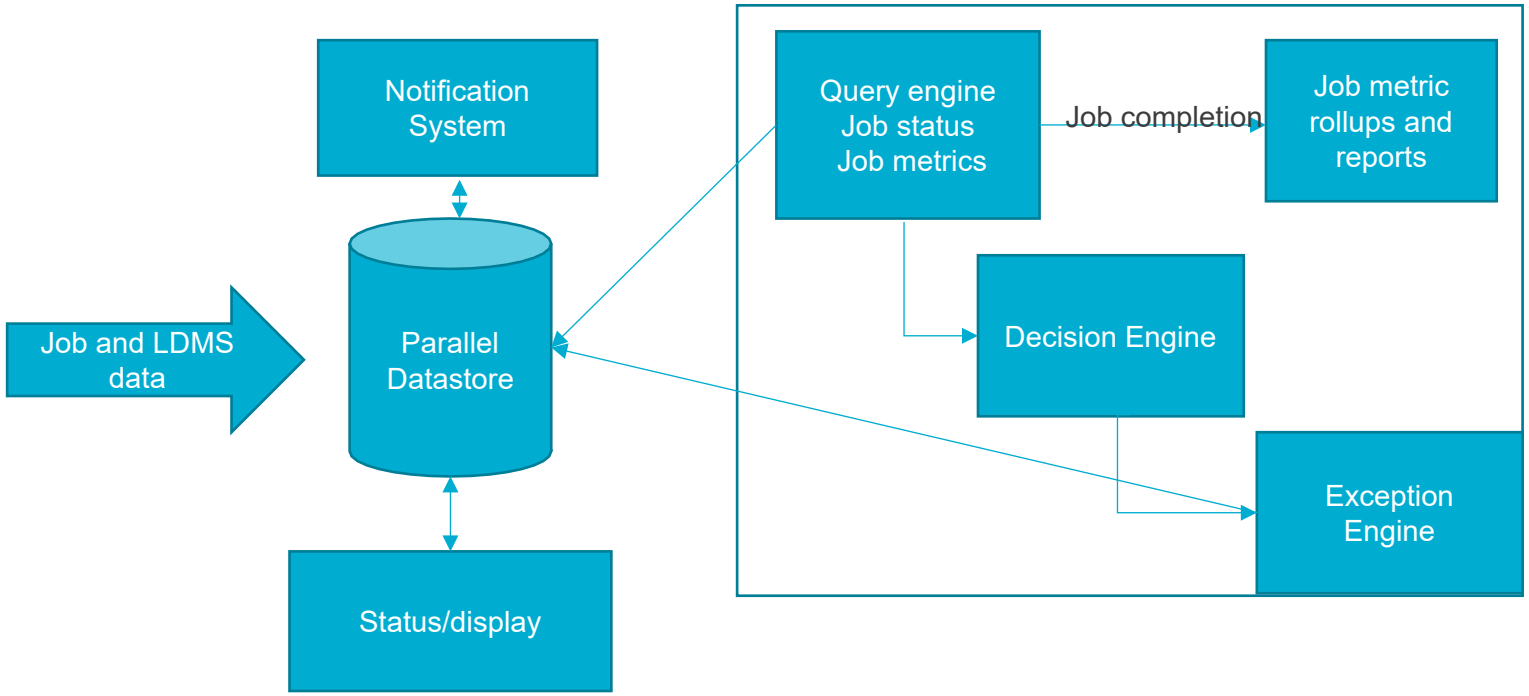
Aggregator Container Deployment w/ store_csv

On a barebone machine that you want to deploy the container:

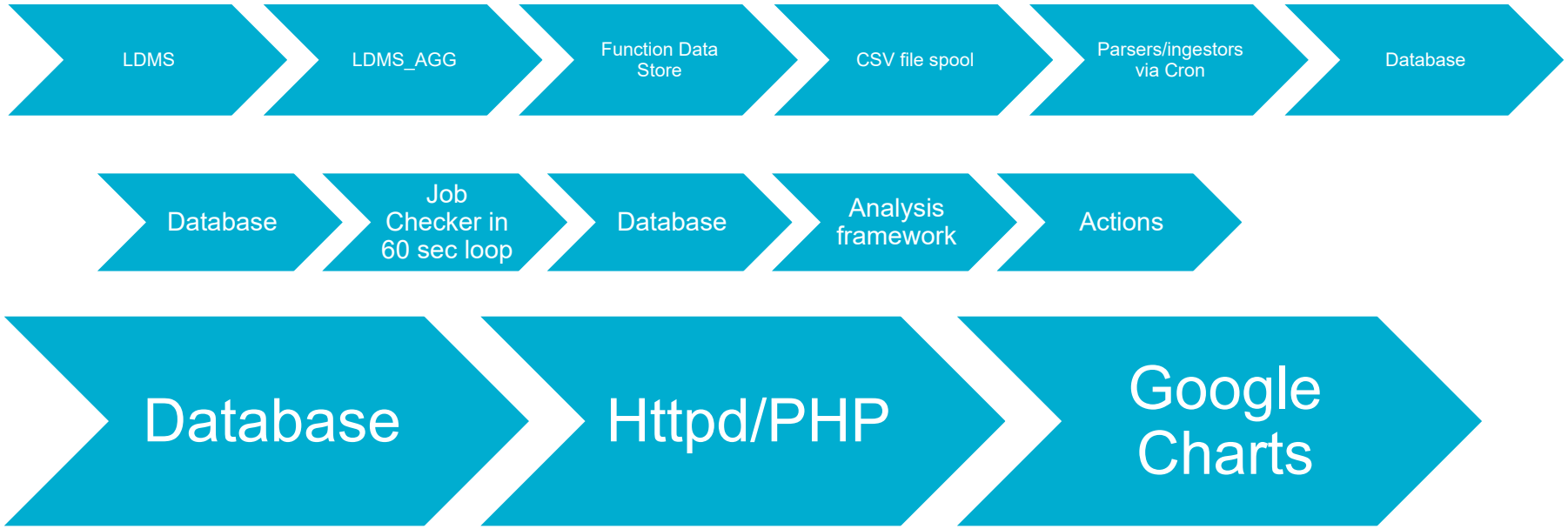
./ldms-agg/run.sh --name csv-11 --prdcr "nid{00001..20}" --mem 256M --offset 400000 --strgp-conf /path/to/csv-strgp.conf

- This creates a container named `csv-11` with 256 MB memory pool that collects LDMS sets from the ldmsd's on nids 1-20, with 200ms offset (default 1s interval).
- "--strgp-conf" can be used instead to provide custom strgp configuration.

Software Structure and Flow



Action pipelines



Test Definitions

Total tests=7

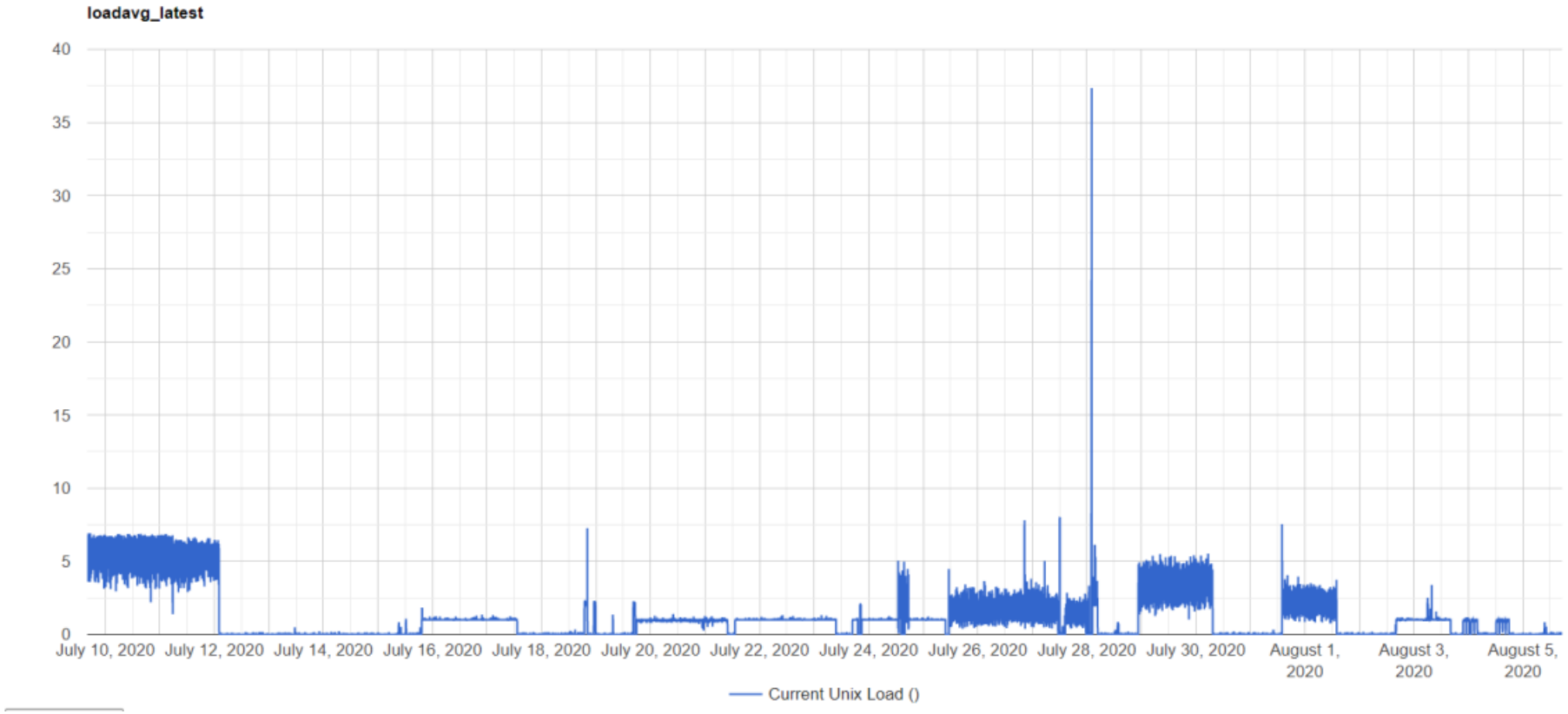
Test Name	Test Type	Test ID	Test Duration	Test Metric	Test Threshold	Test Aggregation	Test Group By
lowJobMem	job	all	300	current_freemem	-100000	min	none
lowMomMem	nodeGroup	mom	300	current_freemem	-10000	min	none
sdbMemory	singleNode	sdb	120	current_freemem	-10000	min	none
noLoad	job	all	300	loadavg_latest	-1	sum	cTime
lowGpu	job	gpu	300	Tesla_K20X_gpu_util_rate	-1	max	none
highload	job	all	300	loadavg_latest	6600	max	none
highOpenScratch	job	all	300	Rate_open_stats_snx11003	1000000	sum	cTime

Criteria fails

2020-08-05 16:26:33	12	job	loadavg_latest	11445057	noLoad	link
2020-08-05 16:23:07	11	job	Tesla_K20X_gpu_util_rate	11445059	lowGpu	link
2020-08-05 16:23:07	11	job	Tesla_K20X_gpu_util_rate	11445057	lowGpu	link
2020-08-05 16:23:07	16	job	Tesla_K20X_gpu_util_rate	11442762	lowGpu	link
2020-08-05 15:47:45	266	job	Tesla_K20X_gpu_util_rate	11443150	lowGpu	link
2020-08-05 15:43:55	50	job	loadavg_latest	11433430	noLoad	link
2020-08-05 15:43:55	33	job	loadavg_latest	11433488	noLoad	link
2020-08-05 15:01:28	54	job	loadavg_latest	11433434	noLoad	link
2020-08-05 14:54:42	14	job	Tesla_K20X_gpu_util_rate	11444969	lowGpu	link
2020-08-05 12:33:22	13	job	loadavg_latest	11444925	noLoad	link
2020-08-05 12:33:22	14	job	Tesla_K20X_gpu_util_rate	11444925	lowGpu	link
2020-08-05 12:09:52	6158	job	loadavg_latest	11339861	noLoad	link
2020-08-05 12:09:52	7116	job	Tesla_K20X_gpu_util_rate	11339861	lowGpu	link
2020-08-05 11:26:32	21	job	loadavg_latest	11444146	highload	link
2020-08-05 11:23:14	20	job	loadavg_latest	11444145	highload	link
2020-08-05 10:53:45	11	job	Tesla_K20X_gpu_util_rate	11444351	lowGpu	link
2020-08-05 10:53:45	202	job	Tesla_K20X_gpu_util_rate	11443727	lowGpu	link
2020-08-05 10:04:32	184	job	Tesla_K20X_gpu_util_rate	11443728	lowGpu	link
2020-08-05 09:47:39	169	job	Tesla_K20X_gpu_util_rate	11442592	lowGpu	link
2020-08-05 07:30:59	43	job	loadavg_latest	11433426	noLoad	link
2020-08-05 07:14:14	113	job	Tesla_K20X_gpu_util_rate	11443729	lowGpu	link
2020-08-05 07:02:12	35	job	loadavg_latest	11433431	noLoad	link
2020-08-05 06:16:44	88	job	Tesla_K20X_gpu_util_rate	11443730	lowGpu	link
2020-08-05 05:19:10	43	job	loadavg_latest	11433435	noLoad	link
2020-08-05 03:52:36	586	job	Tesla_K20X_gpu_util_rate	11436837	lowGpu	link
2020-08-04 21:33:12	34	job	loadavg_latest	11433427	noLoad	link
2020-08-04 19:35:43	145	job	loadavg_latest	11441447	noLoad	link
2020-08-04 18:44:19	101	job	loadavg_latest	11439768	noLoad	link
2020-08-04 16:56:20	31	job	Tesla_K20X_gpu_util_rate	11442060	lowGpu	link
2020-08-04 16:33:59	25	job	Tesla_K20X_gpu_util_rate	11442057	lowGpu	link

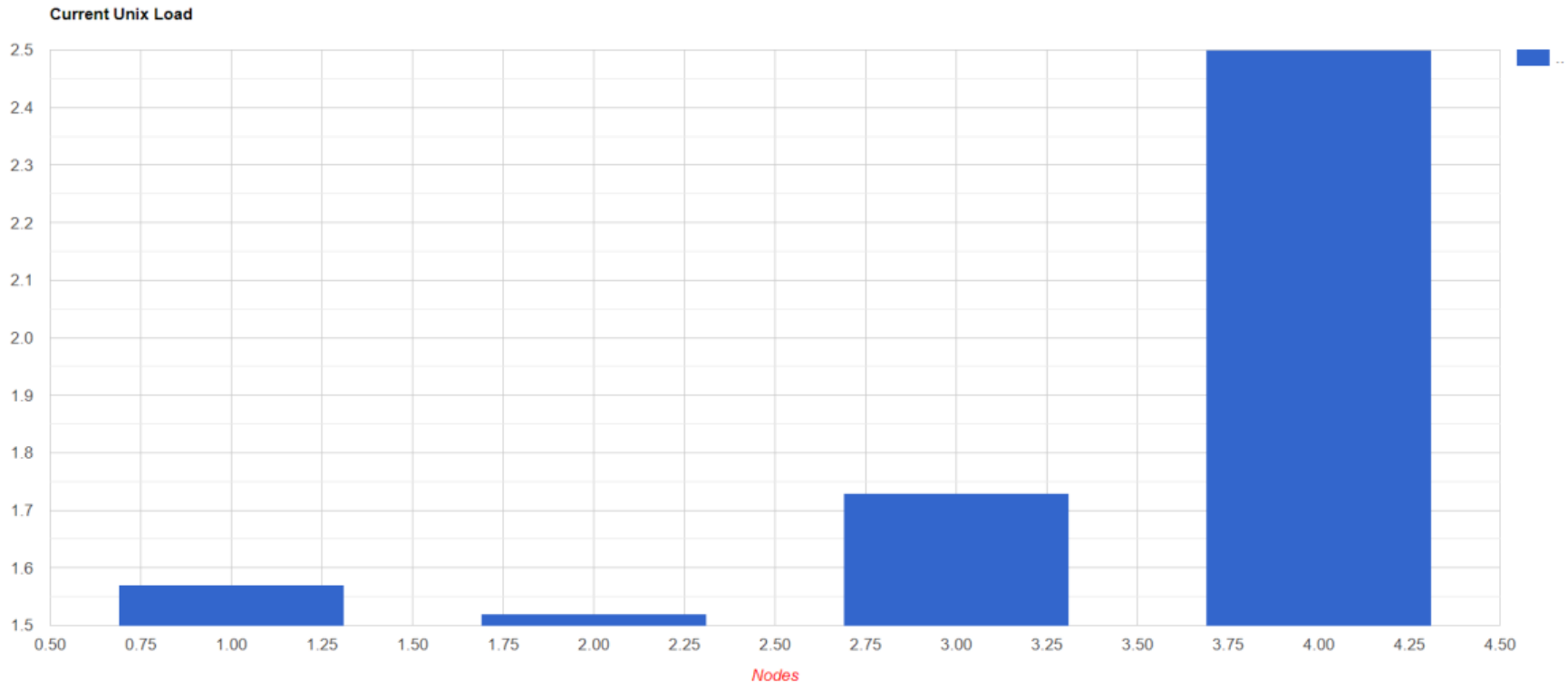
Job Detail for missing load

dutta1 job 11339861 with 1 nodes is still running
Start=Thu, 09 Jul 2020 16:18:04 -0500
End =Wed, 05 Aug 2020 16:59:16 -0500
Data Query took 4 seconds
Average value is 1.2495741488151



Pivot to node based raw data

4 nodes in job 11433434
Mon, 03 Aug 2020 05:54:00 -0500
Data Query took 0 seconds
Job Data



Conclusions

Fallout provides a simple solution to monitoring minimal Linux-based systems

- Four files comprise entire compute node install
- Containerized (Docker) aggregation, analysis, and visualization enables outlier detection and diagnosis on minimally provisioned external host
- Vendor can supply proprietary technology specific samplers as desired

Future Plans

Executable file includes the following compile-time user defined components

- Transport (default sock)
- Authentication configuration (default none)
- Samplers (default meminfo, loadavg, coretemp, technology specific)
- Samplers, transports, and authentication methods added to build using configure line options
- Sampler configurations (default one second sample interval)
- Log level (default QUIET)
- Log location

The standalone executable files will continue to be:

- ldmsd
- ldmsctl
- ldms_ls

To build the standalone version, configure with **--enable-standalone** flag

Questions/Suggestions?