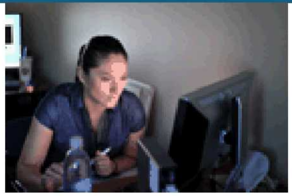
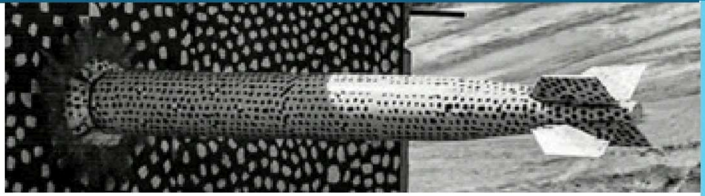
SAND2019-9734PE

HPC Monitoring & Analysis + Power 9 Specifics





Jim Brandt (brandt@sandia.gov)
ORNL August 19, 2019

Unclassified Unlimited Release
SAND 2019 XXXX

Outline

HPC Monitoring Background

Lightweight Distributed Metric Service (LDMS)

- Major features
- Large scale testing and deployments
  - NCSA, Trinity, NERSC
- Typical topology
- Analytics

Deployment on IBM Power 9 (Demo Vortex)

- Standard LDMS sampler plugins
- Power 9 Specific – need design document
  - Will need more work to be made useful
- GPU – reference email – need to develop plugin (funding, develop from template, SNL look into)

Future plans

- HMDSA
  - LDMS
  - Analytics
  - Presentation
  - Feedback

Key Questions Identified Across Operations Community (SC & CUG BoFs)

- System Managers and Users: How can I know if the system is having problems?
- Users: Is **my** application performance variation due to system conditions or code changes?
- Architects, System Managers, and Support Staff: How can a system provide more effective and efficient services?
- Architects and Acquisition Teams: What are the architectural requirements given the site’s workload?

HPC Monitoring Goals and Concerns

Goals:

- Real-time troubleshooting (e.g., nodes down, out of memory, resource congestion)
- Automated recovery from recoverable problems (e.g., congestion avoidance)

Concerns:

- Impact on running applications
- How to aggregate data from different sources for analysis
  - Network, filesystem, CPU utilization, memory utilization
- What analyses and presentations would be meaningful
  - e.g., What raw and derived data would indicate performance-impacting network congestion.
- How to process large amounts of data in real-time

Typical system monitoring:

- Typically performed at intervals of minutes
- Analyses largely consists of detecting monitoring values exceeding pre-defined thresholds
- Data is unsuitable for gaining significant insights into application performance problems

Resource-Aware Computing

Lightweight high-frequency continuous run-time monitoring, analysis, and feedback with synchronized data sampling could enable:

- **Faster problem detection**, including component-specific issues based on a particular component's known behaviors and environment (e.g., thermal variations)
- **Insight into a large-scale application's use of resources** under *production* conditions, including resource contention with other applications
- **Dynamic application-to-resource mapping** based on application needs and system state
- **Better co-scheduling of applications** based on contention for shared resources
- **Dynamic tailoring of system operations** to a data center's changing power demands, temperature etc.

Lightweight Distributed Metric Service (LDMS)

Synchronized system wide data sampling provides resource utilization "snapshots"

- Memory, Memory Bandwidth, CPU, Power, Hardware performance counters
- Network utilization and congestion parameters, I/O

No significant impact on applications at collection rates (1Hz or higher) necessary for resolving resource utilization features

- **Optimized data structures**
- **Optimized Transport** over RDMA, IB, socket
  - The RDMA transport enables aggregation and storage of sampled data at no additional compute node CPU cost
- Testing at scale on Blue Waters (27,648 nodes) and Cori (12,000 nodes)
  - Blue waters ~200GB/day, Cori ~8TB/day
- On-node CPU overhead limited to the cost for data exposure and a single memory copy

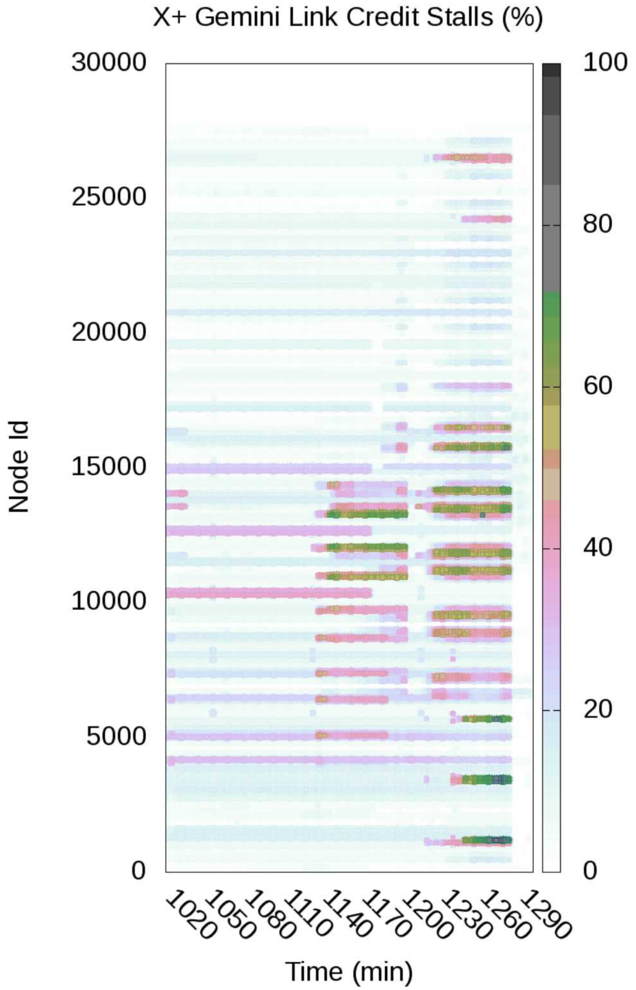
Scalable and low overhead acquisition, transport, and storage of information

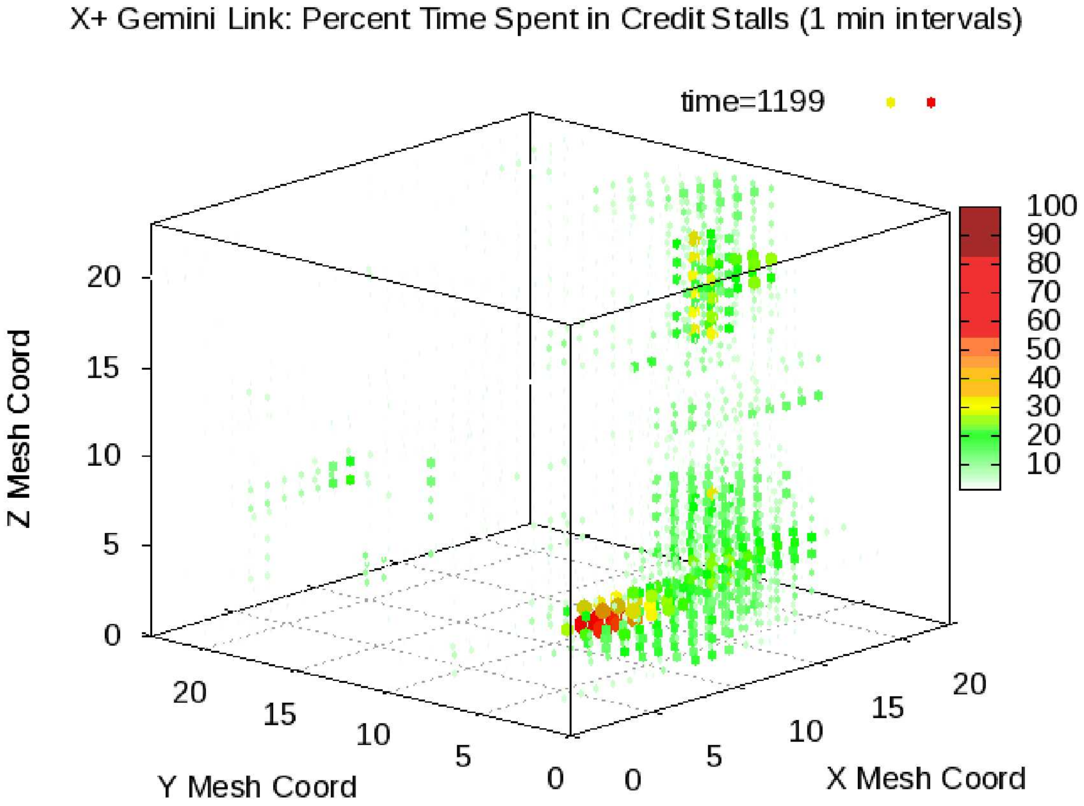
Use Case Examples of System Wide Information Analysis

SNL Current Analytics Endeavors

- ML based anomaly detection and root cause analysis
- Statistical comparison anomaly detection
- Network congestion characterization and correlation with application performance
- Job/application comparisons for regression detection
- Resource utilization profiling
  - Memory, memory bandwidth, CPU, cache, speculative execution, network, file I/O
- Log file analysis using SNL's Baler software

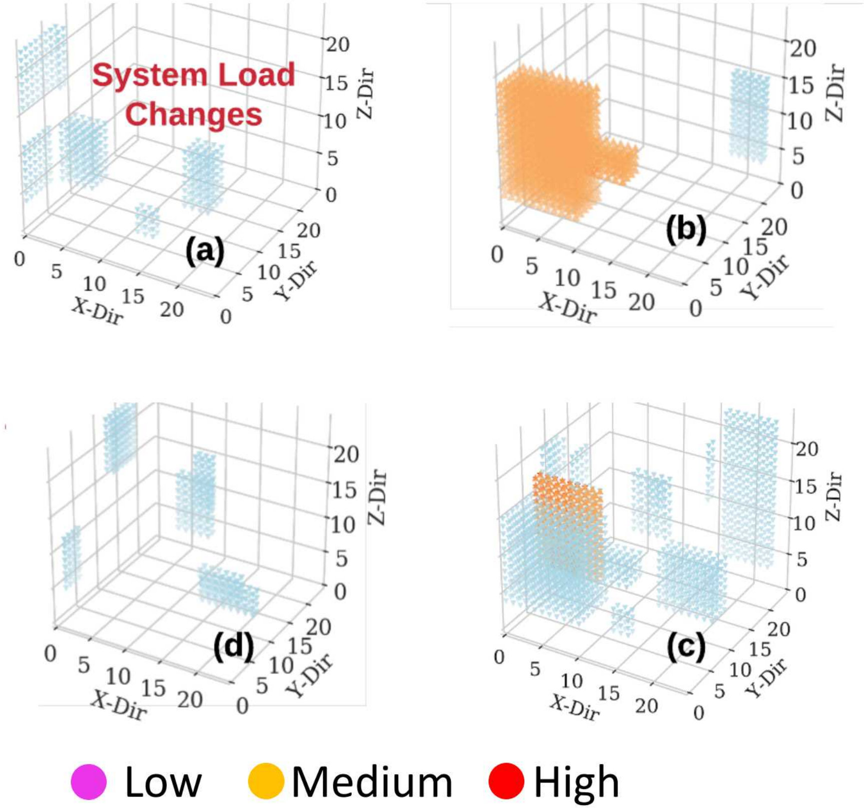
Visualization of Congestion in a Gemini Network



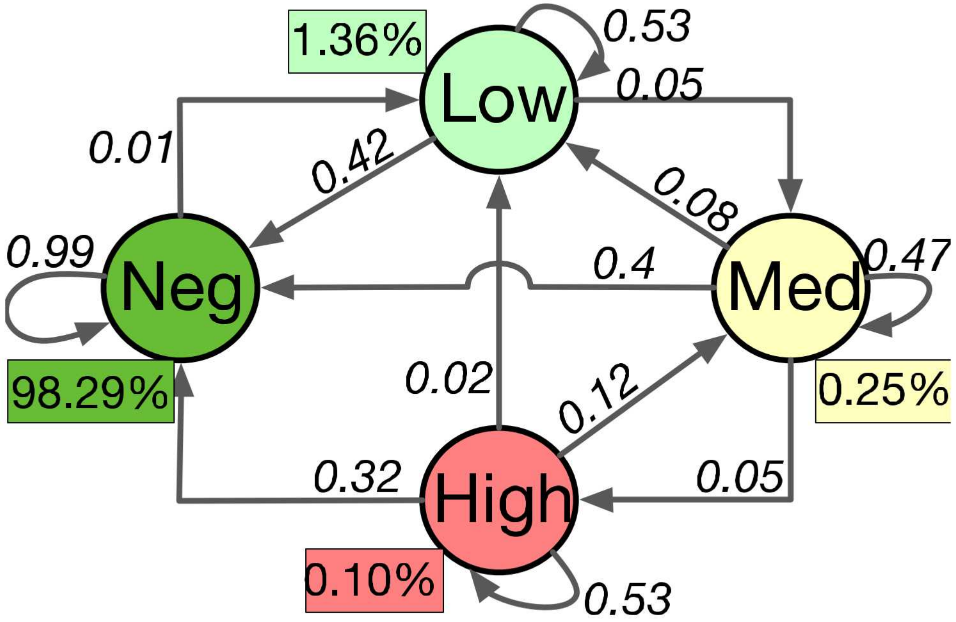


Analysis from NCSA's Blue Waters, *Lightweight Distributed Metric Service: A Scalable Infrastructure for Continuous Monitoring of Large Scale Computing Systems and Applications, Agelastos et al. SC14*

Characterizing Congestion On Toroidal Networks



Congestion Cloud (NCSA Blue Waters)



Continuous presence of highly congested links

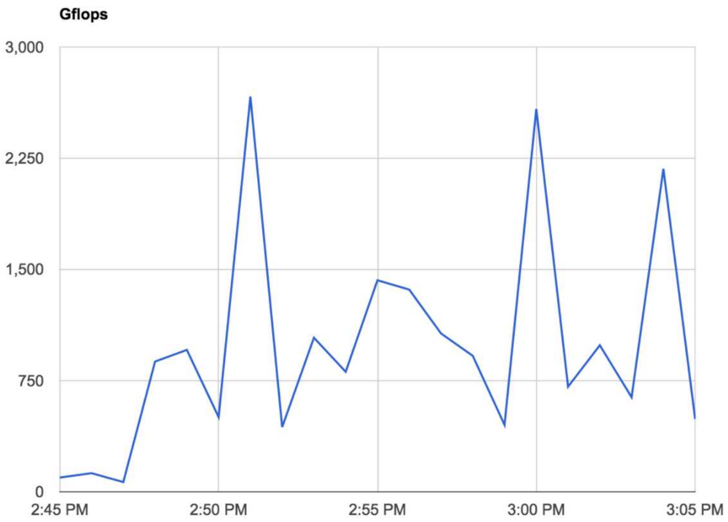
95% of the operation time contained a region with a min size of 20 links.

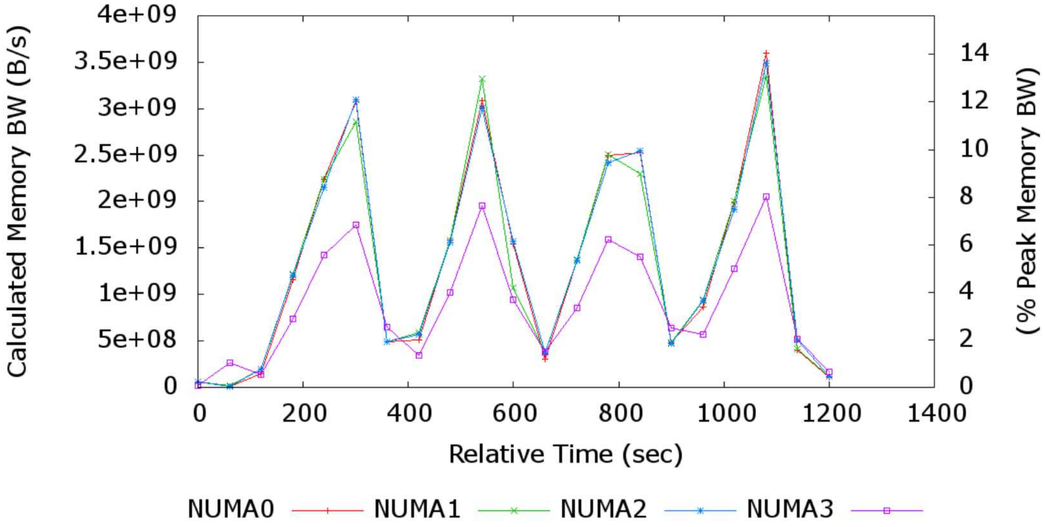
Max **size** of 6904 (17%) links

Average congestion duration: 16 minutes, 95th percentile: 16 hours

System-level Monitoring Captures Application Characteristics

NWCHEM

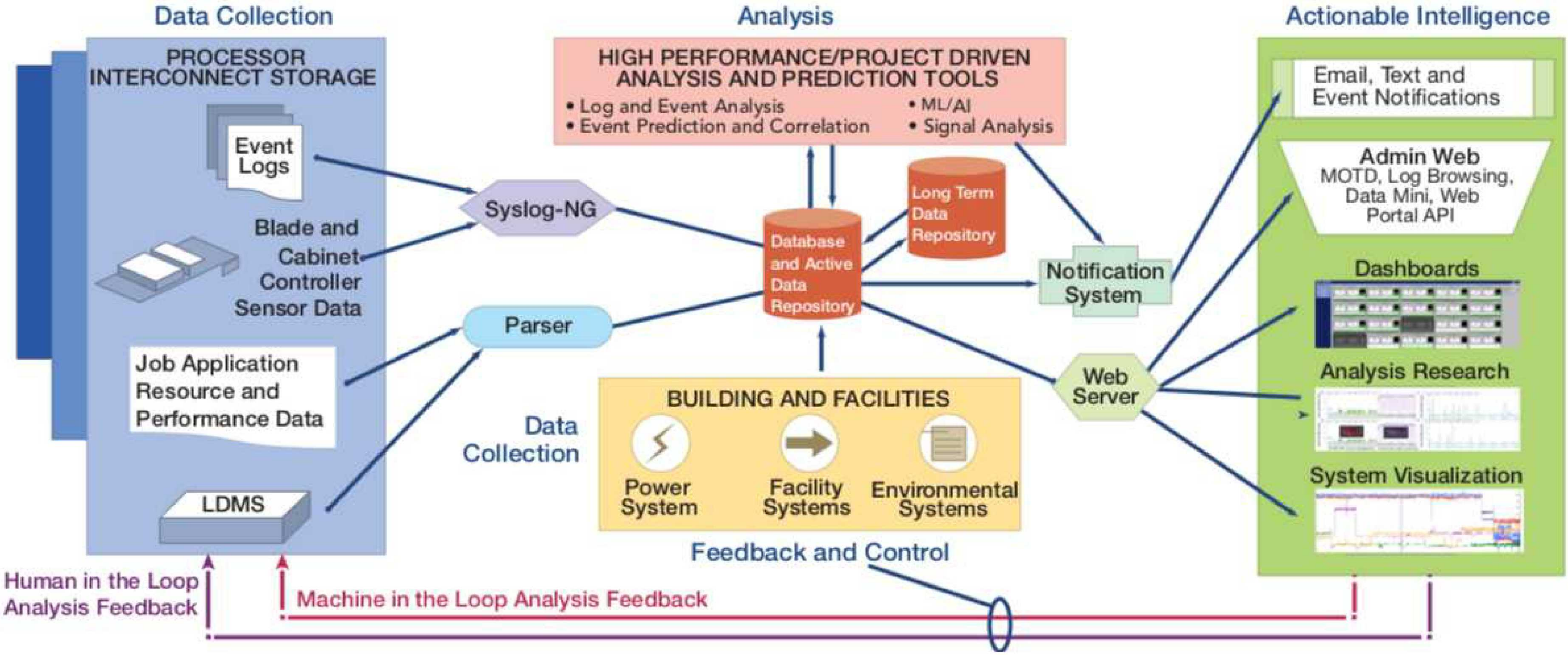




- Mixed QM and molecular mechanics suite
- Run with CCSD as opposed to T(CCSD) - more mem BW limited as opposed to floating pt intensive
- Run characteristics
  - 100 nodes
  - 8 MPI tasks/node
  - 4 OMP threads/task
  - 30 GB mem/node
- 9.7 GF/s/node ave
- 14% max of peak mem BW
  - Mem BW 1 lower NUMA domain - investigate further
- Sawtooth
  - CCS vs CCD phases
  - Instructions/cycle - Same thing but higher frequency

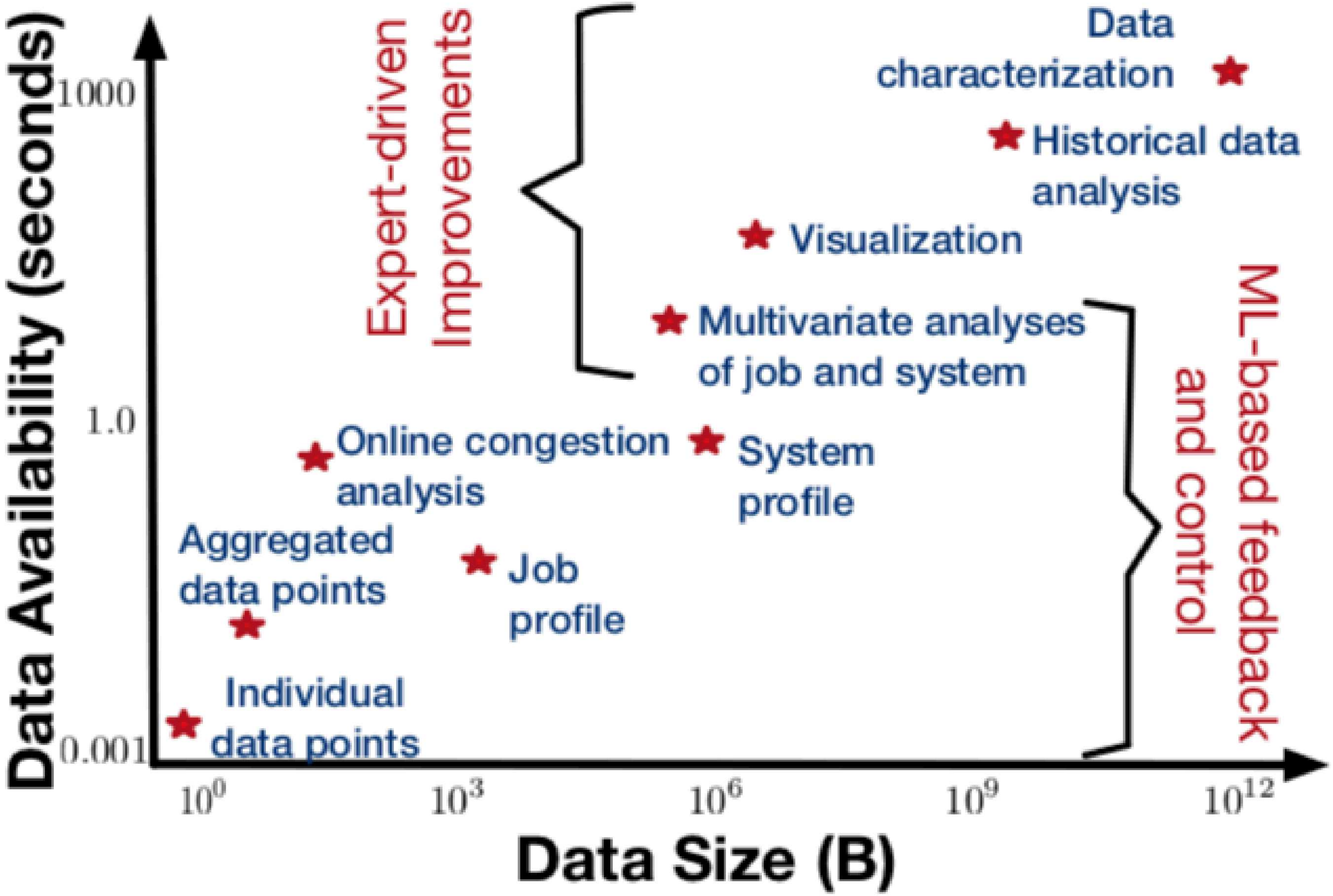
Analysis from NCSA's Blue Waters, *Dynamic Model Specific Register (MSR) Data Collection as a System Service, Bauer et al, CUG 2016*

Holistic Measurement Driven System Assessment (HMDSA) Scalable System Architecture



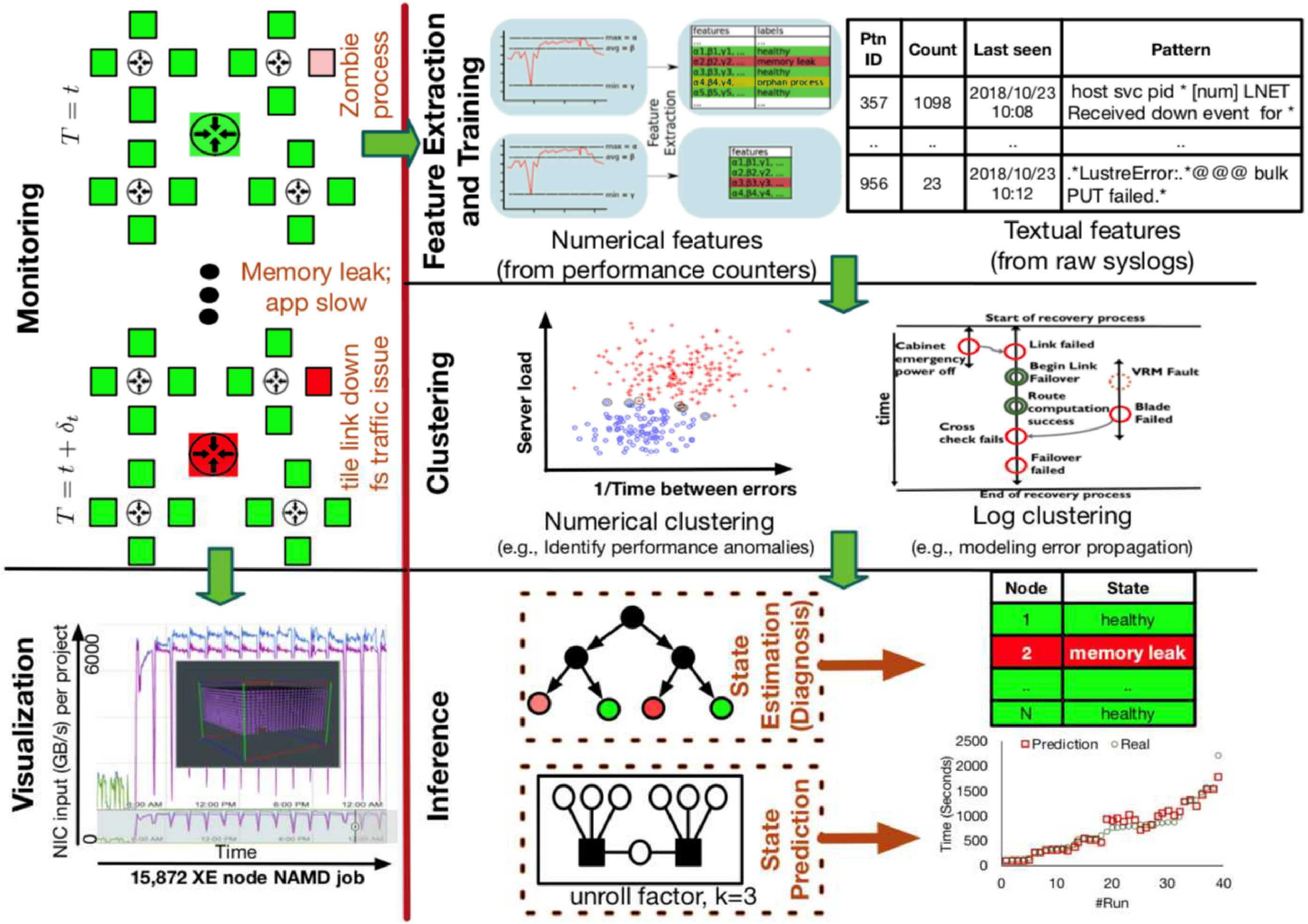
From: Holistic Measurement Driven System Assessment – *ECP Annual Meeting 2019*

Actionable Time Scales



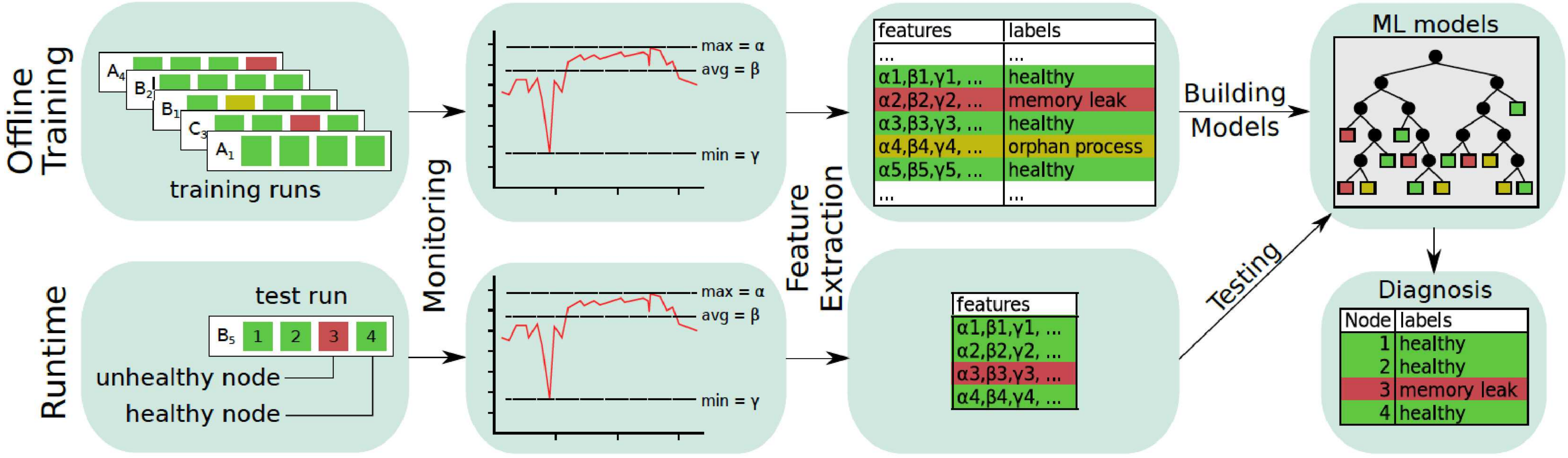
From: Holistic Measurement Driven System Assessment – *ECP Annual Meeting 2019*

Actionable Expert- and ML-Driven Analysis



From: Holistic Measurement Driven System Assessment – *ECP Annual Meeting 2019*

Anomaly Detection and Problem Diagnosis



Detection and diagnosis of performance problems

- Machine learning models built offline are used for classifying observations at runtime
- Detect and diagnose behavioral differences due to: memory leaks, errant processes, contention, etc...

From: Diagnosing Performance Variations in HPC Applications Using Machine Learning ISC 2017 -- Gauss Award Winner

Hardware Resource Utilization

- Hardware performance counters reveal performance-affecting resource-utilization: Cycles per instruction, branching mispredictions, cache misses, etc.
- Analysis: Determine and collect hardware-specific performance counters and assess contextually-specific quantities
- Figure of merit metric: Per rank variation in Cycles per Instruction (compute performance)

```
test/spapi1505734446162: consistent, last update: Mon Sep 18 08:41:11 2017 [183611us]
M u64      component_id                          162
M u64      job_id                                0
D char[]   Appname                               "lulesh"
D u64      Jobid                                 1186151
D char[]   Username                              "oaaziz"
D u8       NumNodes                              4
D u8       PPN                                   16
D u8       NumThreads                            1
D u64[]    Pid                                   17749,17750,17751,17752,17753,17754,17755,17756,17757,17758,17759,17760,17761,17762,
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
D u64[]    BR_INST_RETIRED:COND:precise=1:u=1    5481289172,5476439288,5479319975,5479187901,5476609604,5484115830,5486034361,5471000
3680,5478787138,5478780103,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
D u64[]    BR_MISP_RETIRED:COND:precise=1:u=1    25062,28508,25330,28528,31715,24893,25337,41362,27098,25894,27211,30637,26173,40281,
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
D u64[]    BR_INST_RETIRED:ALL_BRANCHES:precise=1:u=1 5539782057,5530226180,5535554031,5534985031,5531577864,5544667106,5546656936,5487327
7217,5533621662,5534690788,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
D u64[]    BR_MISP_RETIRED:ALL_BRANCHES:u=1      47739,52183,47813,52661,56491,47695,48067,70879,50684,48385,50565,54872,48977,69469,
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
```

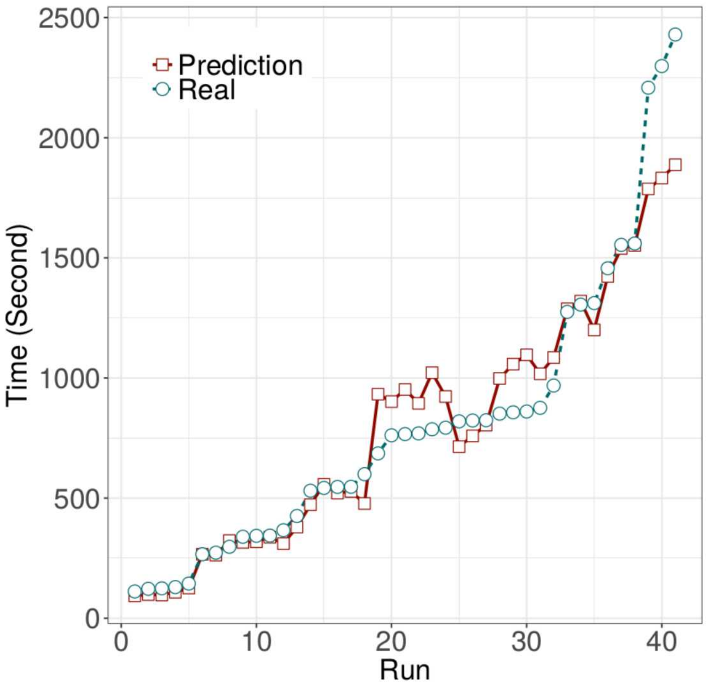
LDMS presents hardware-independent interface to meaningful contextual counters: Branch, Cache, Instructions, General

Domain-specific sensor data collection from Trinity testbed

Application Progress Assessment and Performance Prediction

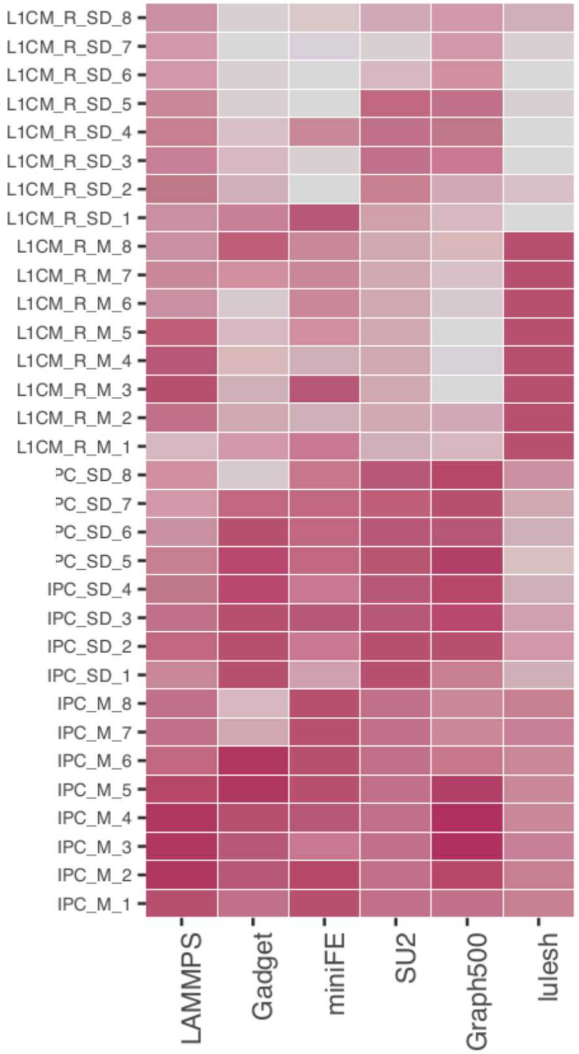
- User-determined data collection and analysis based on specific application details
- Analysis: Use application data to assess application progress and sensitivity
- Figure of merit metric: Avg. heartbeat1 rate

Detect and predict performance problems based on variation of run time progress in sensitive code sections

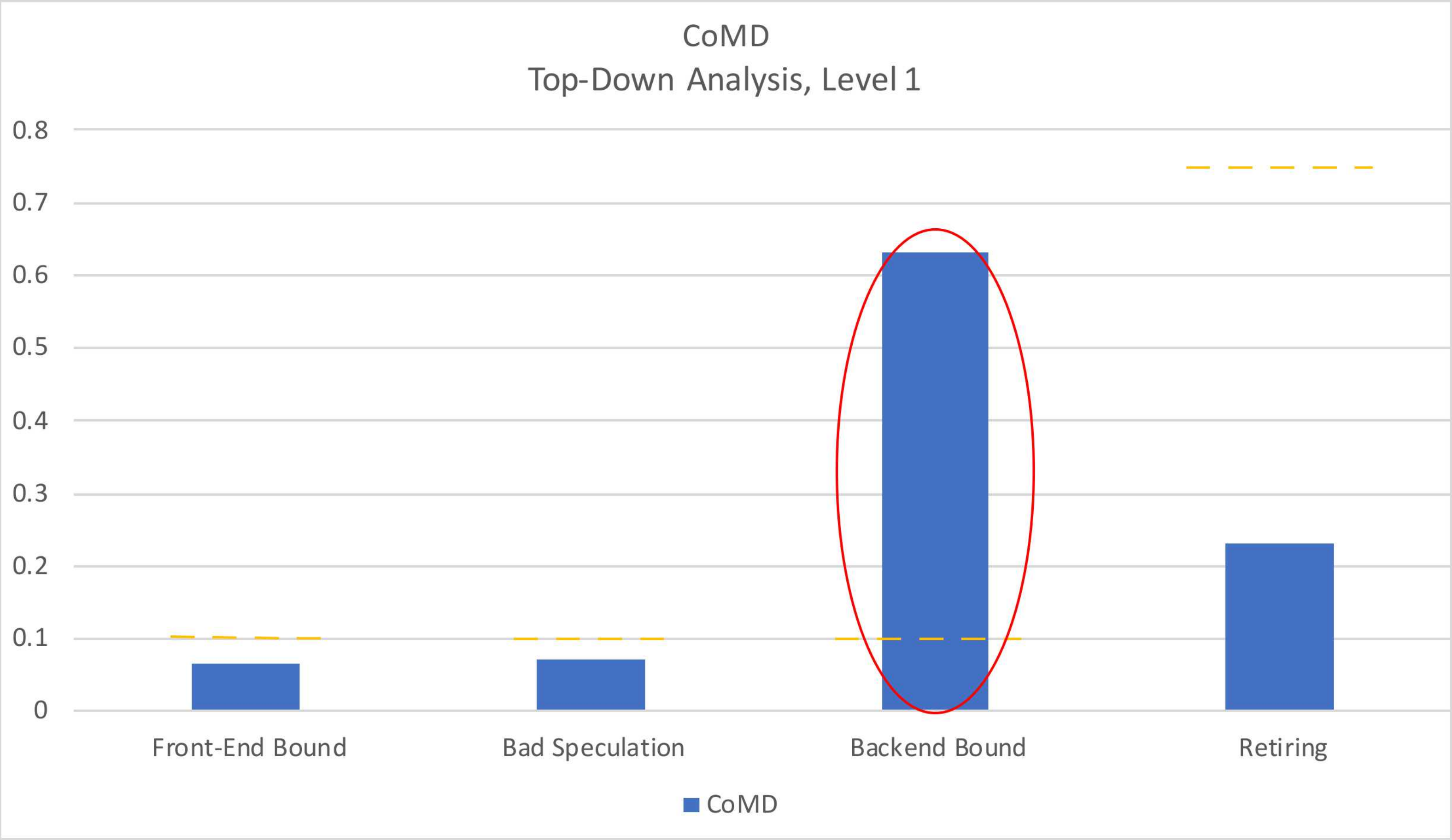


LAMMPS runtime prediction

Interval h/w counter importance heatmap



Top Down Analysis (TDA)
Based on Intel Top Down Analysis Method



From: Enabling HPC Performance Insights via End-to-End Monitoring and Analysis – *in submission to IEEE Cluster 2019*

E2EMon Interface Example Use Case
Same Inputs Different Behaviors

Job ID	App ID	Node ID	Runtime (s)	Back Pressure	Mem Score	Anomalies	PAPI Perf	App Perf
42093	miniAMR	nid000[52-55]	439	0.0	2	None	Back	1.45
42092	miniGhost	nid000[21;29-31]	1043	49.07	2	Cache	Back	-1.93
42091	miniMD	nid000[57-60]	617	5.24	3	Cache	Back	No data
42089	CoMD	nid000[52-55]	742	91.68	1	Cache	Back	1.52
42088	miniAMR	nid000[21;29-31]	447	0.0	2	Cache	Back	1.45
42087	miniGhost	nid000[57-60]	1043	73.88	2	Cache	Back	-0.27
42086	miniMD	nid000[21;29-31]	619	13.33	3	Cache	Back	No data
42084	CoMD	nid000[52-55]	742	90.88	1	Cache	Back	1.81
42034	miniGhost	nid000[52-55]	1022	98.59	1	None	Back	No data
42028	kripke	nid000[57-58	748	0.0	1	Mem	No data	No data
42027	kripke	nid000[21;29-31]	751	0.0	1	Mem	Back	No data
42019	kripke	nid000[52-55]	1092	0.0	1	Mem	Front, Back	No data

From: Enabling HPC Performance Insights via End-to-End Monitoring and Analysis – *in submission to IEEE Cluster 2019*

Job Based Drill Down

E2EMon: Job Detail

back to job listing

Job ID	App ID	Node ID	Runtime (s)
42093	miniAMR	nid000[52-55]	439

Back Pressure	Mem Score	Anomalies	PAPI Perf	App Perf
0.0	2	None	Back	1.45

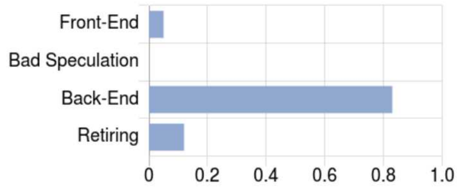
Application Heartbeat Analysis:

Application heartbeat data for job 42093, [0, 1, 2, 3, 4] (graph)

HBeat #	Avg	Dev	Min	Max	MinPID	MaxPID	Graph
0	4.41	1.45	4.28	4.64	55442	55856	graph
1	88.26	28.97	85.69	92.88	55442	55856	graph
2	0.0	0.0	0.0	0.0	24050	24050	graph
3	3530.3	1158.65	3427.57	3715.3	55442	55856	graph
4	0.0	0.0	0.0	0.0	24050	24050	graph

TopDown Analysis:

First Level TopDown Analysis for job 42093



Threshold for reporting Retiring is .75, threshold for all other bottlenecks is .1

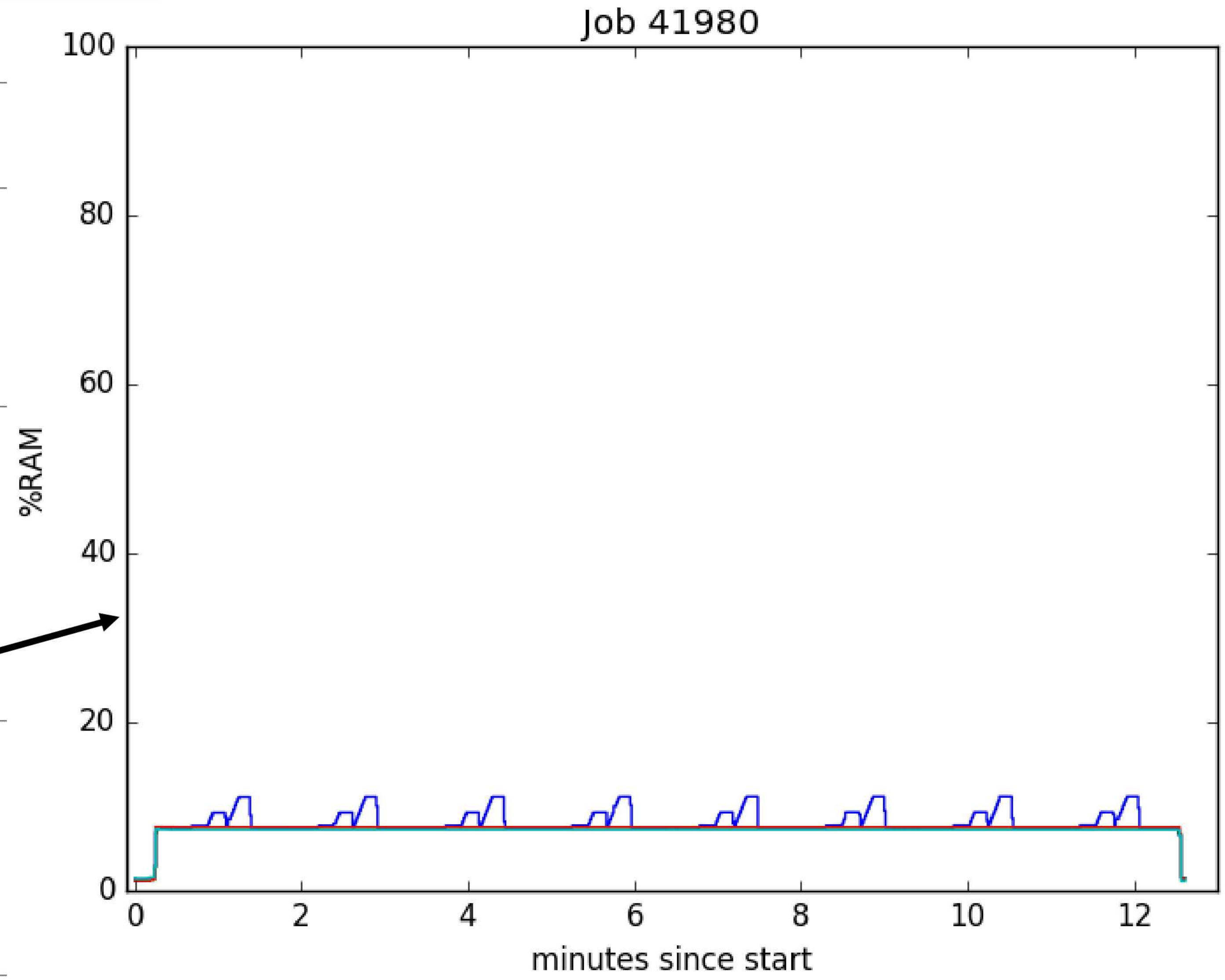
Memory Analysis for 42093:

Avg %RAM	Stdev %RAM	Max %RAM	Min %RAM peak
9.8	1.6	10.4	10.3
Sample interval	Balance	max %RAM host	Min %RAM peak host
1	1	52	55

VIEW	Filesystem location
histogram	/lscratch/e2emon/data/memplot/4209303000000000.png
line plot	/lscratch/e2emon/data/memplot/4209302000000000.png
job end UNIX UTC	1558735473

%RAM is computed as 100*(MemTotal-MemFree)/MemTotal

No detailed network data available



From: Enabling HPC Performance Insights via End-to-End Monitoring and Analysis – *in submission to IEEE Cluster 2019*

E2EMon Interface Example Use Case
Same Inputs Different Behaviors

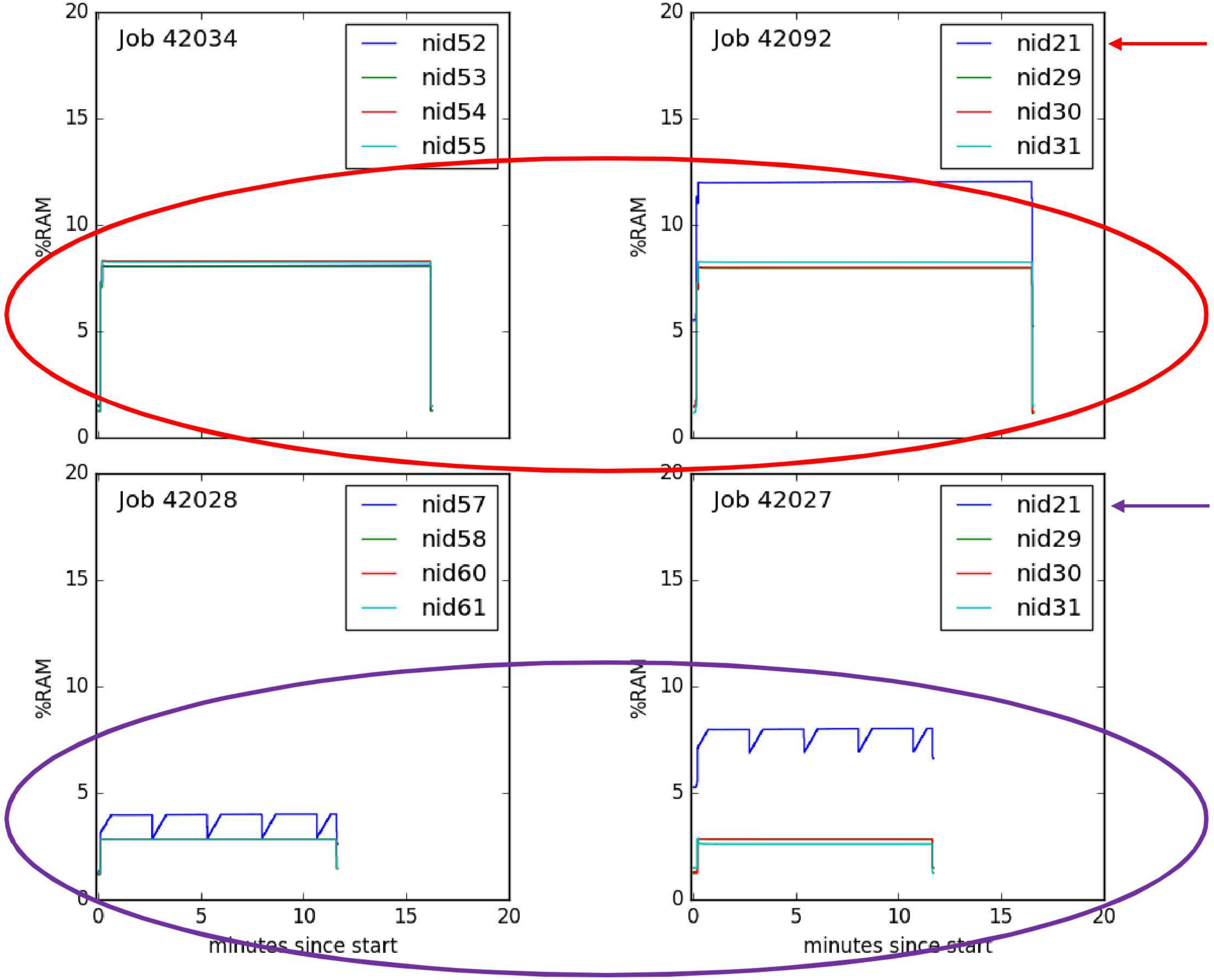
Job ID	App ID	Node ID	Runtime (s)	Back Pressure	Mem Score	Anomalies	PAPI Perf	App Perf
42093	miniAMR	nid000[52-55]	439	0.0	2	None	Back	1.45
42092	miniGhost	nid000[21;29-31]	1043	49.07	2	Cache	Back	-1.93
42091	miniMD	nid000[57-60]	617	5.24	3	Cache	Back	No data
42089	CoMD	nid000[52-55]	742	91.68	1	Cache	Back	1.52
42088	miniAMR	nid000[21;29-31]	447	0.0	2	Cache	Back	1.45
42087	miniGhost	nid000[57-60]	1043	73.88	2	Cache	Back	-0.27
42086	miniMD	nid000[21;29-31]	619	13.33	3	Cache	Back	No data
42084	CoMD	nid000[52-55]	742	90.88	1	Cache	Back	1.81
42034	miniGhost	nid000[52-55]	1022	98.59	1	None	Back	No data
42028	kripke	nid000[57-58	748	0.0	1	Mem	No data	No data
42027	kripke	nid000[21;29-31]	751	0.0	1	Mem	Back	No data
42019	kripke	nid000[52-55]	1092	0.0	1	Mem	Front, Back	No data

From: Enabling HPC Performance Insights via End-to-End Monitoring and Analysis – *in submission to IEEE Cluster 2019*

Same Inputs Different Behaviors Drill Down

Two identical runs of miniGhost

Two identical runs of kripke



From: Enabling HPC Performance Insights via End-to-End Monitoring and Analysis – *in submission to IEEE Cluster 2019*

Lightweight Distributed Metric Service (LDMS)

LDMS Overview

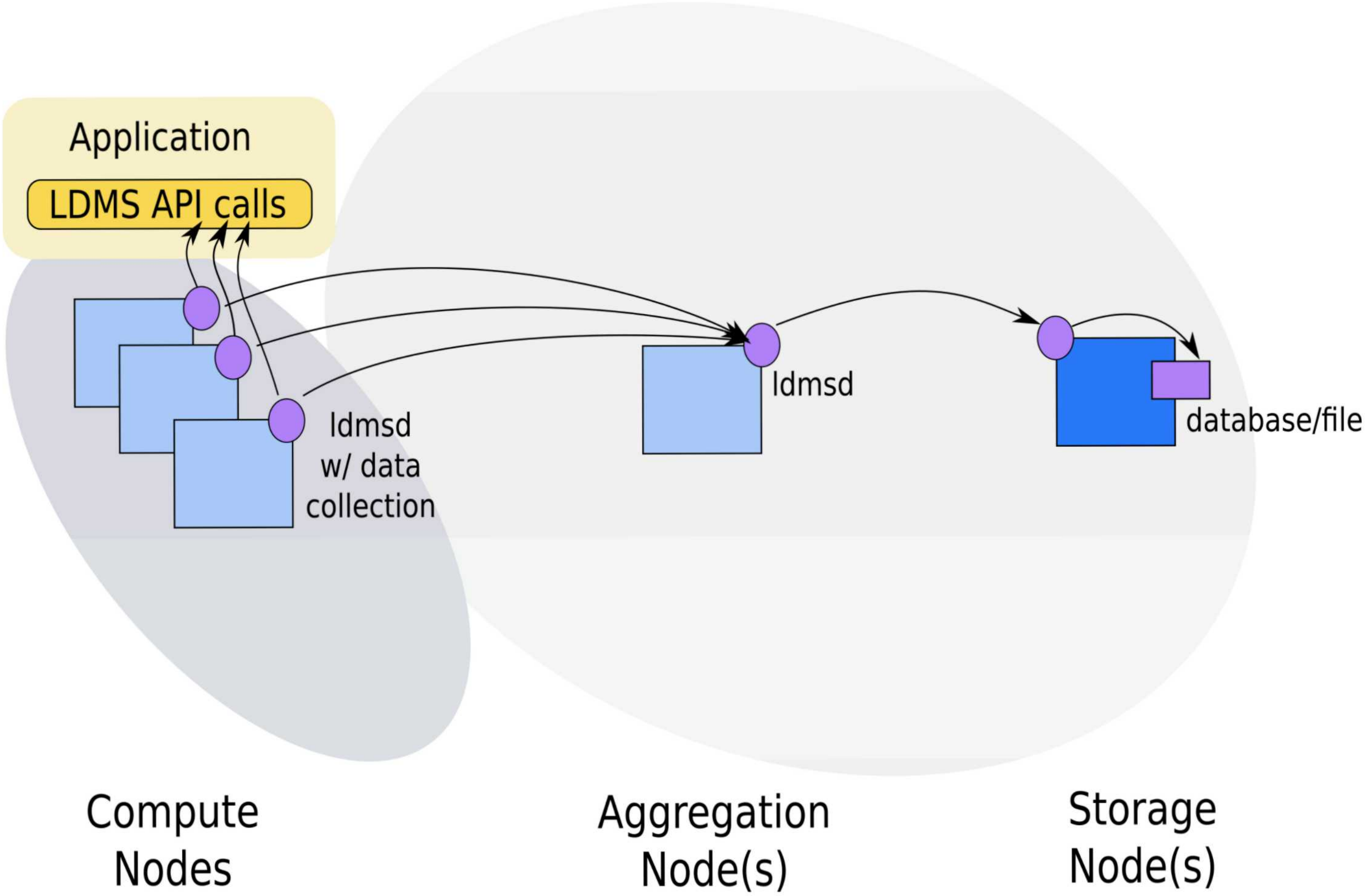
What is the Lightweight Distributed Metric Service (LDMS)?

- Daemon based
- Collect numeric data
- Move, aggregate, and possibly analyze data
- Store data, raw or processed, for use in:
  - Troubleshooting
  - Comparison
  - Optimization
  - Inform future designs

Typical use cases

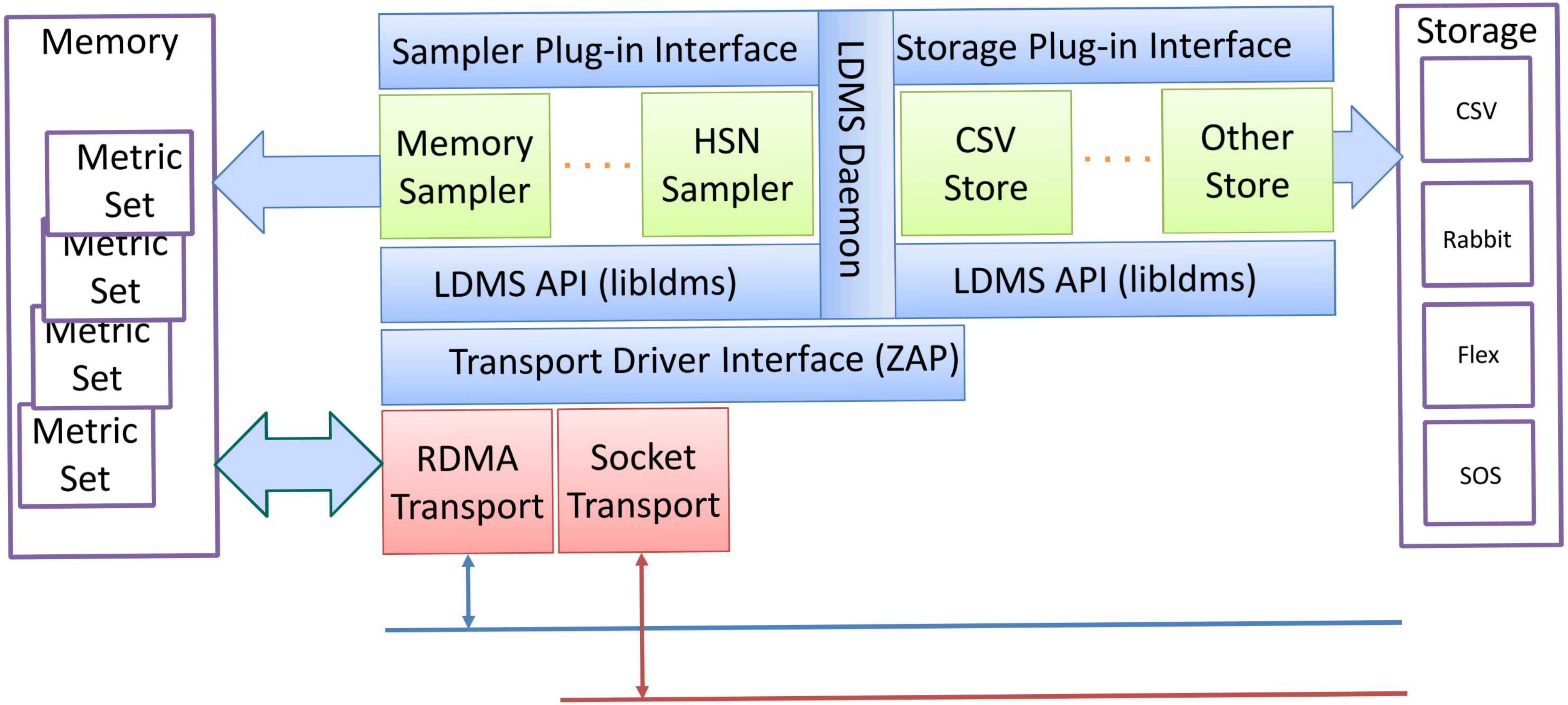
- Identify applications memory (and other resource) utilization behaviors
- Identify network congestion
- Determine over-provisioned resources
- Identify heavy Lustre users

Lightweight Distributed Metric Service (LDMS) High Level Overview

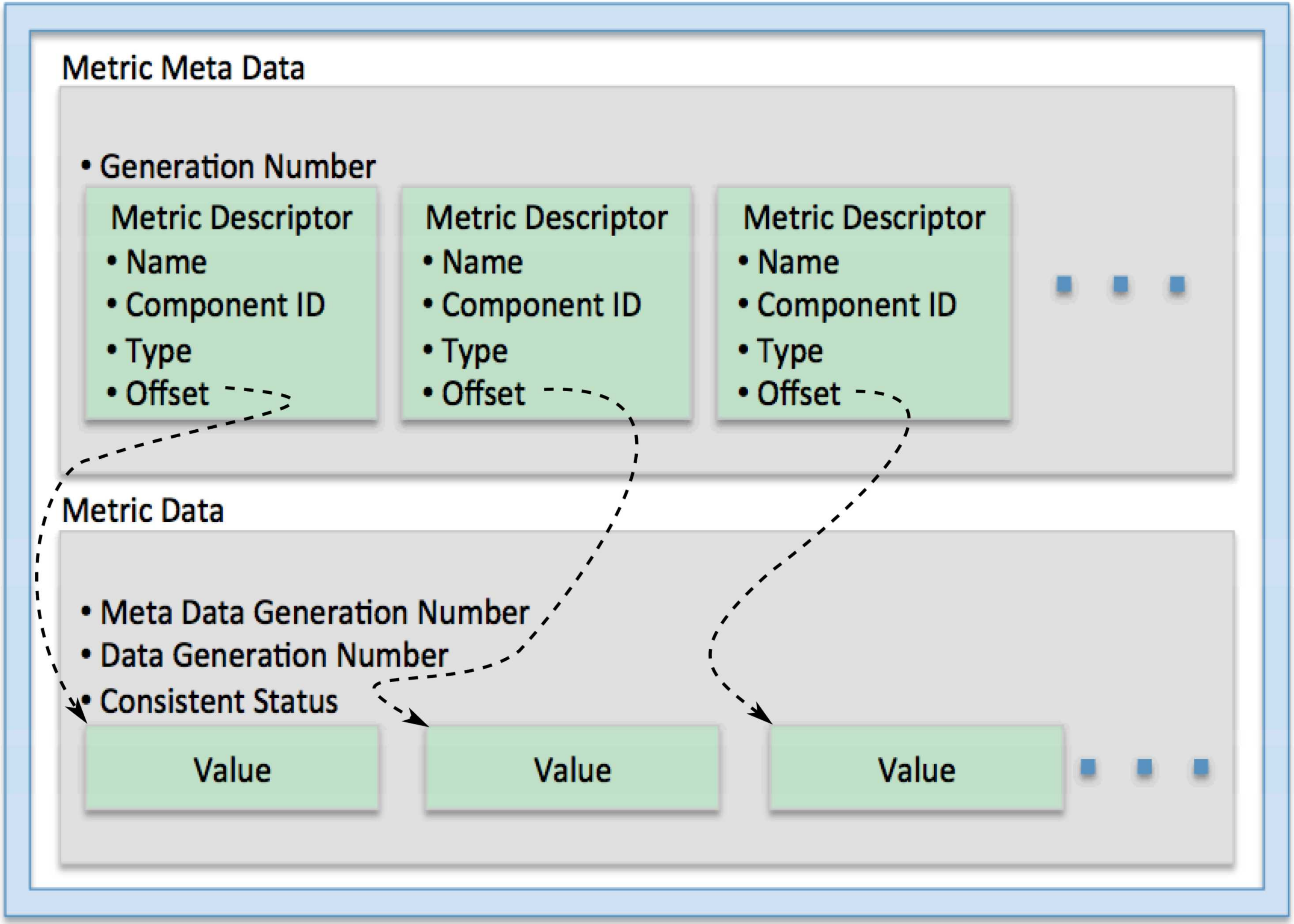


* Only the current data is retained on-node

LDMS Plugin Architecture



Metric Set Memory



LDMS Data Representation

In-memory data object. Only the most recent* data sample is stored-on node

Tools to query a data set in human readable format

timestamp

Metadata and data sizes, sample duration, source affiliation, etc

Data type, name, value

```
localhost1/meminfo: consistent, last update: Tue Jan 17 14:22:45 2017 [2222us]
  METADATA --------
    Producer Name : localhost1
    Instance Name : localhost1/meminfo
      Schema Name : meminfo
             Size : 1816
     Metric Count : 43
               GN : 2
  DATA ------------
        Timestamp : Tue Jan 17 14:22:45 2017 [2222us]
         Duration : [0.000074s]
       Consistent : TRUE
             Size : 384
               GN : 83
  -----------------
M u64        component_id                              0
D u64        job_id                                    0
D u64        MemTotal                                  33083764
D u64        MemFree                                   32255448
D u64        Buffers                                   0
D u64        Cached                                    64700
D u64        SwapCached                                0
D u64        Active                                    10856
D u64        Inactive                                  10928
D u64        Active(anon)                              5888

                ...

D u64        VmallocUsed                               3619068
D u64        VmallocChunk                              34339332092
D u64        HugePages_Total                           0
D u64        HugePages_Free                            0
D u64        HugePages_Rsvd                            0
D u64        HugePages_Surp                            0
D u64        Hugepagesize                              2048
D u64        DirectMap4k                               804864
D u64        DirectMap2M                               9680896
D u64        DirectMap1G                               23068672
```

Lightweight Distributed Metric Service: A Scalable Infrastructure for Continuous Monitoring of Large Scale Systems and Applications – SC14

Security and Ease of Use

Plugin based authentication now supports "none", shared secret, and munge based authentication for access to ldmsd data and set information

Sample update frequency hints provide auto-configuration of pull based aggregation

```
voltrino:~ # ldms_ls -h localhost -x sock -p 411 -a munge
nid00006/vmstat
nid00006/var/opt/cray/imps/image_roots/login-large_cle_6.0up05_sles_12sp3_x86-64_ari-created20180529.SNL
nid00006/var/opt/cray/imps
nid00006/procstat
nid00006/meminfo
nid00006/jobinfo
nid00006/cray_aries_r_sampler
nid00006/aries_rtr_mmr
nid00006/aries_nic_mmr
nid00006/aries_linkstatus
```

```
gentile@voltrino:~> ldms_ls -h localhost -x sock -p 411 -a munge
nid00006/vmstat
nid00006/jobinfo
```

Metric sets seen based on permissions via munge: Root (top) sees more than user (bottom)

```
voltrino:~ # ldms_ls -h localhost -x sock -p 411 -a munge -v nid00006/meminfo
nid00006/meminfo: consistent, last update: Mon Jul 02 11:07:09 2018 [2540us]
  APPLICATION SET INFORMATION ------
           updt_hint_us : 1000000:0
  METADATA --------
    Producer Name : nid00006
    Instance Name : nid00006/meminfo
      Schema Name : meminfo_x86_ven0000fam0006mod002D
             Size : 1976
     Metric Count : 46
               GN : 2
             User : root(0)
            Group : 44476
      Permissions : -rwxrwx---
  DATA ------------
        Timestamp : Mon Jul 02 11:07:09 2018 [2540us]
         Duration : [0.000035s]
       Consistent : TRUE
             Size : 416
               GN : 16923
  ----------------
```

Update hint

Permissions

Features

- Synchronized data sampling provides system state "snapshots"
- Binary formatted sets can contain a mixture of text, scalars, and vectors
- Supports both pull (on a defined interval) and push (on update) modes over both socket and RDMA transports
- Update intervals are independent across samplers and metric sets and can be modified on-the-fly
- Update interval hints propagated from Sampler through aggregation hierarchy and are propagated upon modification
- Job ID is carried in set meta-data and can be used as a search key
- Support for json based pub-sub using LDMS transport (are data and meta-data separated?)
- Plugin based authentication (none, shared secret, munge)
- Failover aggregator pairs can be defined and operate automatically through automated information exchange and co-monitoring
- Plugin based storage (CSV, SOS, Rabbit, Kafka, function, flex) supports a broad eco-system of storage, data exchange, and analysis
- Vector of sets - A sampler can collect, and locally store, multiple time instances of the same metric sets (e.g., for *high frequency collection*).

Attributes

Minimal CPU footprint

- Support for kernel sampler to minimize contention with applications for CPU
- RDMA transport doesn’t involve compute node CPU for pull and minimal for push

Small memory footprint on compute nodes ~3MB (depends on set sizes)

Minimal network footprint

- Meta-data only transmitted upon change
- Data transmitted upon change or pull

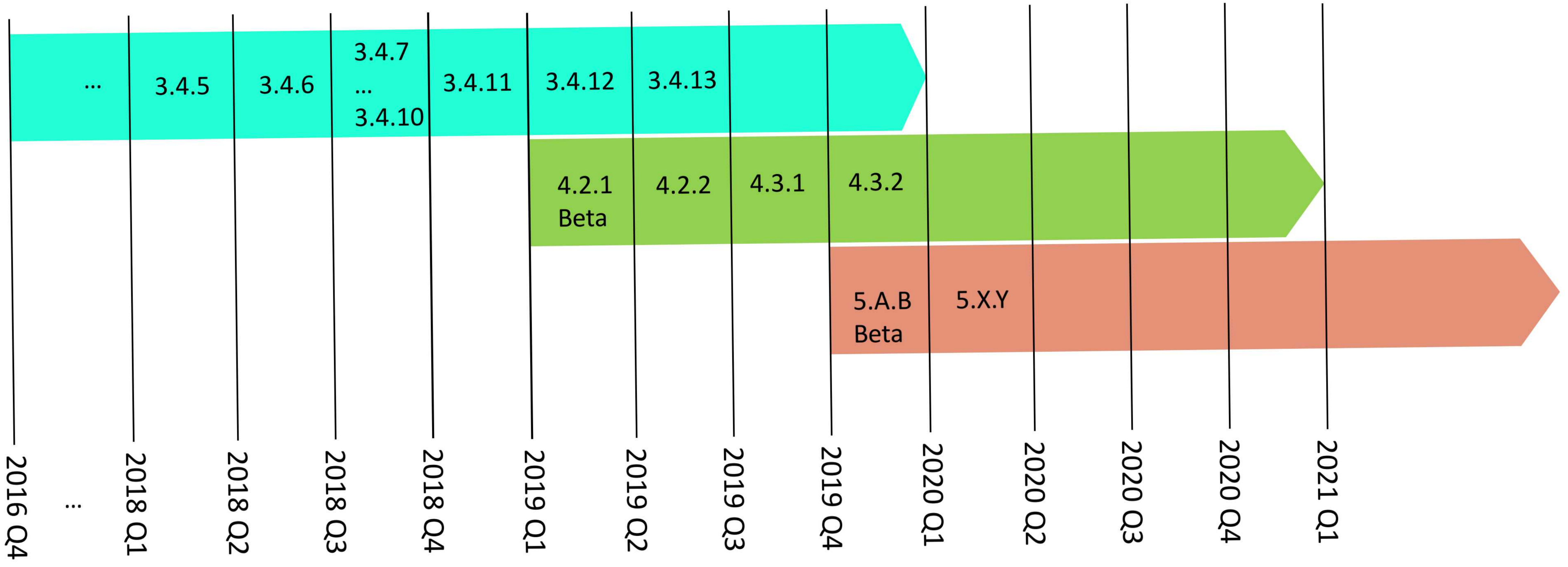
Large fan-in ratios of 1000s:1 (e.g., Blue Waters 7500:1, Cori 4000:1) minimizes required infrastructure

- Dependent on number of number of sets and aggregation interval

No statistically significant adverse impact on applications at collection rates (1Hz or higher) necessary for resolving resource utilization features

- Testing at scale on Blue Waters (27,648 nodes), Trinity (2 * 10,000 nodes), and Cori (12,000 nodes)

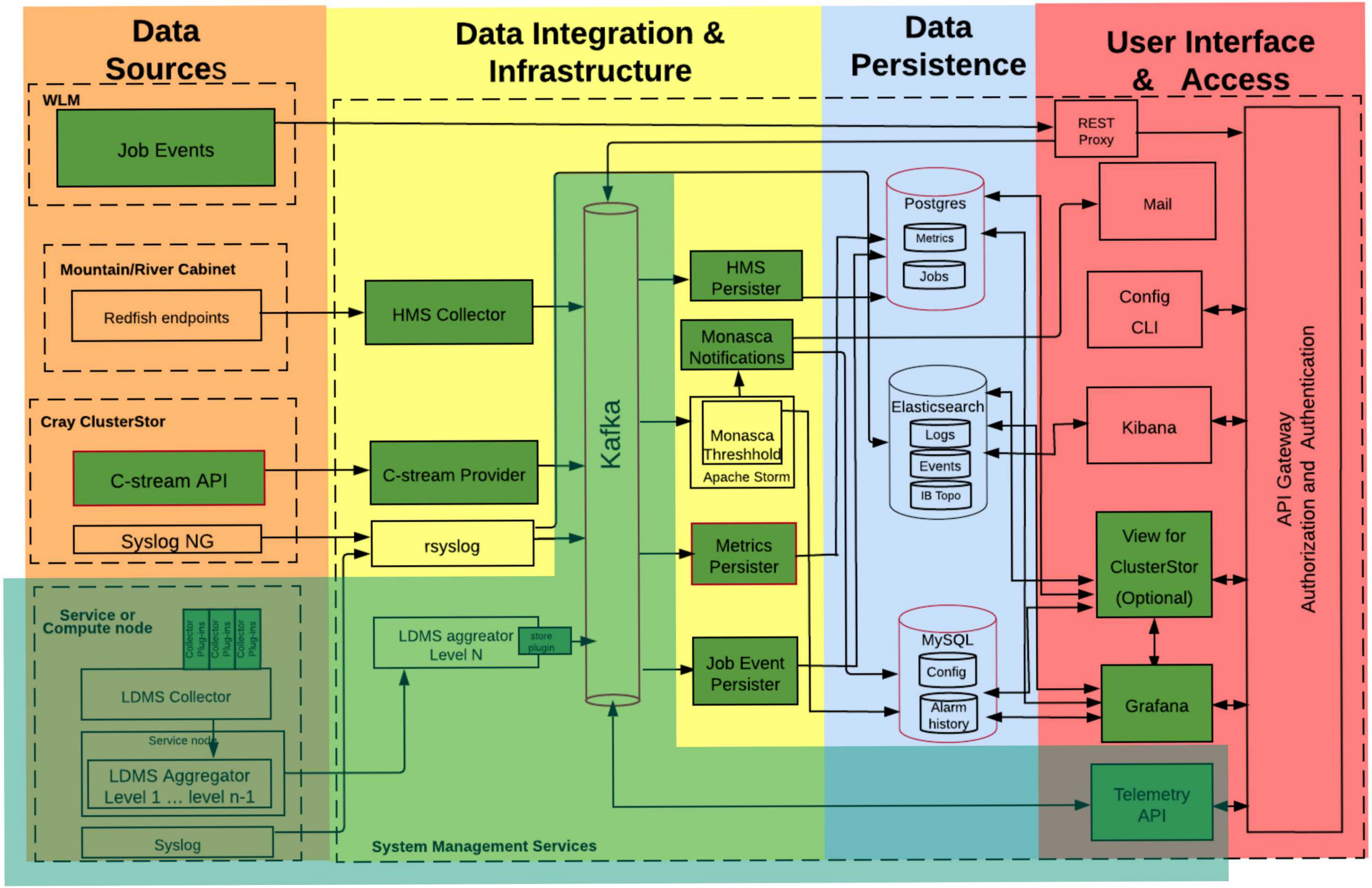
Planned LDMS Release Timeline



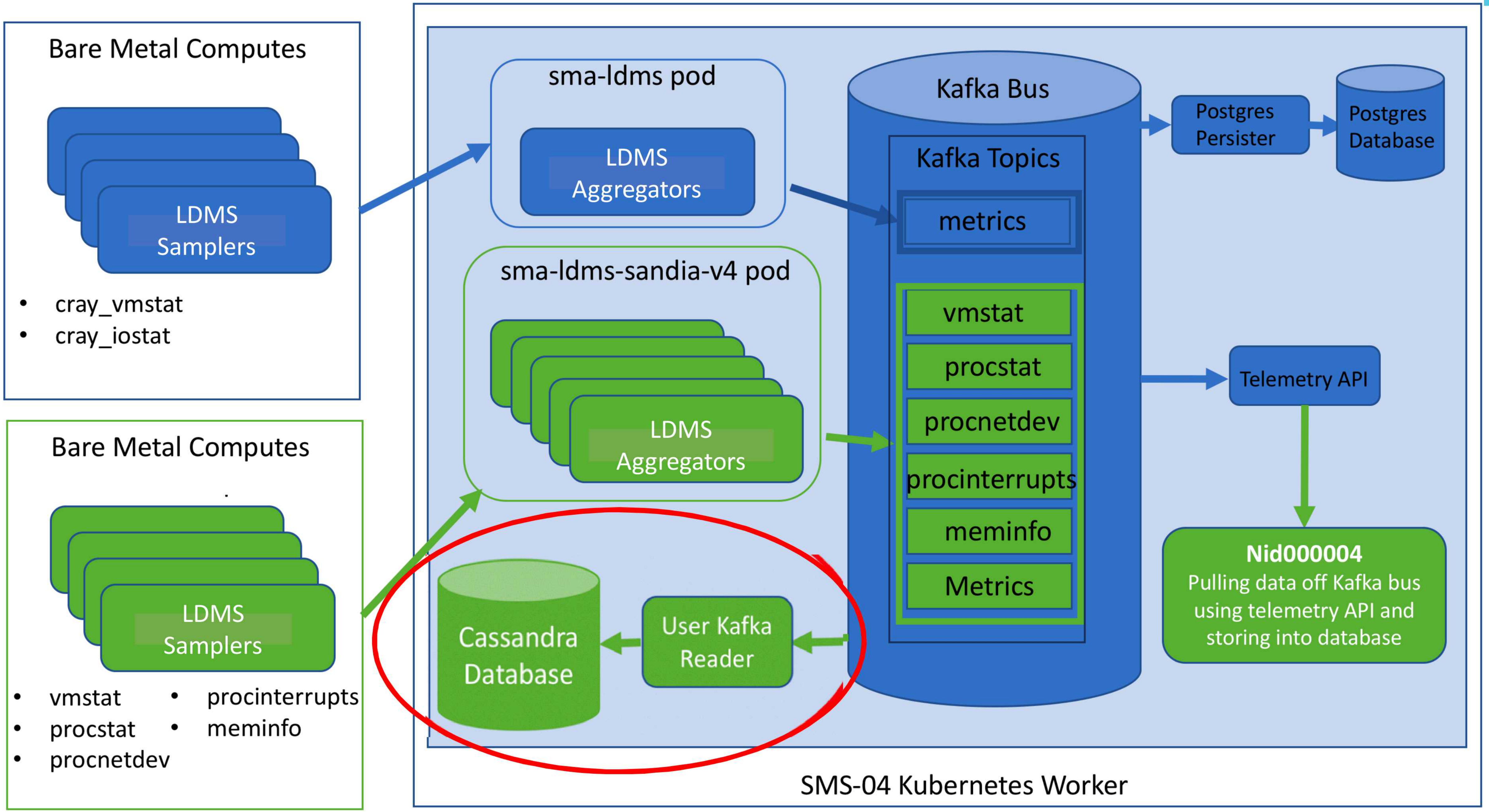
Platform Specific Deployments of Interest

The Focus of Our Investigation

- LDMS
- Kafka
- Kubernetes
- Docker



Implementation of User-Driven Metric Collection



IBM Power 9 LDMS Deployment Options

Normal Linux OS and PAPI sampler plugins function as on any other Linux based platform

Additional instrumentation is available via the IBM Power 9 via the "/sys/firmware/opal/exports/occ_inband_sensors"

interface

- This is the interface is read by IBM CMS
- There is an LDMS sampler plugin to read the same binary file

IBM xxx interface could provide more flexibility and additional information but would require additional programming

- With the current programming a node requires a BIOS reflash and a reboot to modify which metrics are exposed
- There is additional CPU overhead for in-band sampling of data via the sysfs interface – doesn't really matter due to 4 cores lying idle
- Would like to program xxx processor to:
  - Accept configuration parameters for choosing what data to expose
  - Bundle data and meta-data in LDMS formatted data structures that are mmapped into user space
    - If LDMS used for transport, a kernel module would enable DMA of data from nodes to aggregators without need for host CPU
    - CMS plugin that reads LDMS formatted data would enable CMS with no loss of functionality

GPU sampler still needs to be written

Power 9 LDMS Data Set

CURVDD
CURVDN
VOLTVDD
VOLTVDDSENSE
VOLTVDN
VOLTVDNSENSE
VOLTDROOPCNTC – all zero
VOLTDROOPCNTQ – all zeros
TEMPNEST
TEMPVDD – numeric value
TEMPPROCTHRMC
TEMPDIMM
TEMPGPU
TEMPGPUMEM
UTILC
UTIL
NUTILC

FREQA
FREQAC
PWRSYS
PWRGPU
PWRAPSSCH
PWRPROC
PWRVDD
PWRVDN
PWRMEM
IPS
STOPDEEPACTC
STOPDEEPREQC
IPSC
NOTBZEC
NOTFINC
PROCPWRTHROT
PROCOTTHROT
MEMPWRTHROT
MEMOTTHROT

OCC Data Availability

11.3.2.1 Performance Sensors

Sensor Name	SMF Mode?	Master only?	#	Unit	Type	Loc	Sample time	Scale Factor	Description
IPS	N	N	1	MIP	PERF	PROC	8MS	1	Vector sensor that takes the average of all the cores in Processor x
STOPDEEPACTCy	Y	N	24	ss	PERF	CORE	8MS	1	Deepest actual stop state that was fully entered during sample time for core y
STOPDEEPREQCy	Y	N	24	ss	PERF	CORE	8MS	1	Deepest stop state that has been requested during sample time for core y
IPSCy	N	N	24	MIP	PERF	CORE	8MS	1	Instructions per second for core y on this Processor
NOTBZECy	N	N	24	cyc	PERF	CORE	8MS	1	Not Busy (stall) cycles counter for core y on this Processor
NOTFINCy	N	N	24	cyc	PERF	CORE	8MS	1	Not Finished (stall) cycles counter for core y on this Processor
MSTLCy	N	N	24	cpi	PERF	CORE	8MS	1	Memory (L1) stalled cycles per instruction for this Processor, Core y
MRDWRMxPy	N	N	8	GBs	PERF	MEM	8MS	0.00128	Sum of Memory read and write requests per sec for Memory Controller x port pair y
MEMSPSTATMxPy	Y	N	8	%	PERF	MEM	8MS	1	Static Memory throttle level setting for MC x (01/23) port pair y (0,1) when not in a memory throttle condition
MEMSPMxPy	N	N	8	%	PERF	MEM	8MS	1	Current Memory throttle level setting for MC x (01/23) port pair y (0,1)
PROCPWRTHROT	N	N	1	#	PERF	PROC	500US	1	Count of processor throttled due to power
PROCOTTHROT	N	N	1	#	PERF	PROC	32MS	1	Count of processor throttled for temperature
MEMPWRTHROT	N	N	1	#	PERF	MEM	500US	1	Count of memory throttled due to power
MEMOTTHROT	N	N	1	#	PERF	MEM	32MS	1	Count of memory throttled due to memory Over temperature

11.3.2.2 Power Sensors

Sensor Name	SMF Mode?	Master only?	#	Unit	Type	Loc	Sample time	Scale Factor	Description
PWRSYS	Y	Y	1	W	POWER	SYS	500US	1	Bulk power of the system
PWRGPU	Y	N	1	W	POWER	GPU	500US	1	Power consumption for GPUs per socket (OCC) read from APSS
PWRAPSSCHx	Y	Y	16	W	POWER	SYS	500US	1	Power Provided by APSS channel x (where x=0…15)
PWRPROC	Y	N	1	W	POWER	PROC	500US	1	Power consumption for this Processor
PWRVDD	Y	N	1	W	POWER	PROC	1MS	1	Power consumption for this Processor's Vdd (calculated from AVSBus readings)
PWRVDN	Y	N	1	W	POWER	PROC	1MS	1	Power consumption for this Processor's Vdn (nest) (calculated from AVSBus readings)
PWRMEM	Y	N	1	W	POWER	MEM	500US	1	Power consumption for Memory for this Processor read from APSS

Sensor Name	SMF Mode?	Master only?	#	Unit	Type	Loc	Sample time	Scale Factor	Description
FREQA	Y	N	1	MHz	FREQ	PROC	8MS	1	Average of all core frequencies for Processor
FREQACy	Y	N	24	MHz	FREQ	CORE	8MS	1	Average/actual frequency for this processor, Core y based on OCA data

11.3.2.4 Utilization Sensors

Sensor Name	SMF Mode?	Master only?	#	Unit	Type	Loc	Sample time	Scale Factor	Description
UTILCy	N	N	24	%	UTIL	CORE	8MS	0.01	Utilization of this Processor's Core y (where 100% = fully utilized): NOTE: per thread HW counters are combined as appropriate to give this core level utilization sensor
UTIL	N	N	1	%	UTIL	PROC	8MS	0.01	Average of all Cores UTILCy sensor
NUTILCy	N	N	24	%	UTIL	CORE	3S	0.01	Normalized average utilization, rolling average of this Processor's Core y

11.3.2.5 Temperature Sensors

Sensor Name	SMF Mode?	Master only?	#	Unit	Type	Loc	Sample time	Scale Factor	Description
TEMPNEST	N	N	1	C	TEMP	PROC	8MS	1	Average temperature of nest DTS sensors
TEMPPROCTHRMCy	N	N	24	C	TEMP	CORE	8MS	1	The combined weighted core/quad temperature for processor core y
TEMPDIMMx	N	N	16	C	TEMP	MEM	64MS	1	DIMM temperature for DIMM x
TEMPGPUx	?	N	3	C	TEMP	GPU	1S	1	GPU x (0..2) board temperature
TEMPGPUxMEM	?	N	3	C	TEMP	GPU	1S	1	GPU x hottest HBM temperature (individual memory temperatures are not available)

11.3.2.6 Voltage Sensors

Sensor Name	SMF Mode?	Master only?	#	Unit	Type	Loc	Sample time	Scale Factor	Description
VOLTVDD	Y	N	1	mV	VOLTAGE	VRM	1MS	0.1	Processor Vdd Voltage (read from AVSBus)
VOLTVDDSENSE	Y	N	1	mV	VOLTAGE	PROC	1MS	0.1	Vdd Voltage at the remote sense. (AVS reading adjusted for loadline)
VOLTVDN	Y	N	1	mV	VOLTAGE	VRM	1MS	0.1	Processor Vdn Voltage (read from AVSBus)
VOLTVDNSENSE	Y	N	1	mV	VOLTAGE	PROC	1MS	0.1	Vdn Voltage at the remote sense. (AVS reading adjusted for loadline)

11.3.2.7 Current Sensors

Sensor Name	SMF Mode?	Master only?	#	Unit	Type	Loc	Sample time	Scale Factor	Description
CURVDD	Y	N	1	A	CURRENT	VRM	1MS	0.01	Processor Vdd Current (read from AVSBus)
CURVDN	Y	N	1	A	CURRENT	VRM	1MS	0.01	Processor Vdn Current (read from AVSBus)

11.3.3 Other OCC Sensors for AMESTER

This is a list of sensors that are NOT written to main memory but are available via AMESTER.

Sensor Name	#	Unit	Type	Loc	Sample time	Scale Factor	Description
PWRFAN	1	W	POWER	SYS	500US	1	Power consumption of the system fans
PWRIO	1	W	POWER	SYS	500US	1	Power consumption of the IO subsystem (including storage digital 5Volt or less rail)
PWRSTORE	1	W	POWER	SYS	500US	1	Power consumption of the storage subsystem (storage 12Volt rail)
FREQREQCy	24	MHz	FREQ	CORE	8MS	1	Requested frequency from OCC voting box for this processor, Core y
TEMPCy	24	C	TEMP	CORE	8MS	1	Average temperature of core DTS sensors for Processor's Core y
TEMPQy	6	C	TEMP	QUAD	8MS	1	Average temperature of quad (in cache) DTS sensors for Processor's Quad y
TEMPPROCTHRM	1	C	TEMP	PROC	8MS	1	Maximum of all TEMPPROCTHRMCy core temperatures
TEMPPROCAVG	1	C	TEMP	PROC	8MS	1	Average of all TEMPPROCTHRMCy core temperatures
VRMPROCOT	1	#	TEMP	VRM	2MS	1	Vdd/Vdn VRM over temperature status. Read from AVS bus The accumulator is a count of number of times this was asserted (updated every sample time) '1' = one of the procs VRM OT is asserted '0' = no Processor VRM OT asserted
TODclock0	1	us	TIME	SYS	8MS	16	TOD clock Low 16 bits in 16us resolution
TODclock1	1	s	TIME	SYS	8MS	1.048576	TOD clock mid 16 bits in 1.05s resolution
TODclock2	1	day	TIME	SYS	8MS	0.795364	TOD clock high 3 bits in 0.796 day resolution
CUR12VSTBY	1	A	CURRENT	SYS	500US	.01	12V standby current read from APSS channel (if channel assigned for 12V standby current)
VRHOTMEMPRCCNT	1	#	GENERIC	SYS	500US	1	Memory VR HOT read from APSS GPIO(s). This is info only not used in any control loops. The accumulator is a count of number of times this was asserted (updated every sample time) '1' = one of the procs memory VR_HOT is asserted '0' = no Memory VR_HOT asserted
MRDMy	8	GBs	PERF	MEM	4MS	0.00128	Memory read requests per second for memory controller y (0..7)
MWRMy	8	GBs	PERF	MEM	4MS	0.00128	Memory write requests per second for memory controller y (0..7)
MIRCMy	8	eps	PERF	MEM	4MS	1	0.01 Events/second. Memory Inter-request arrival idle interval longer than programmed threshold for memory controller y (0..7)

Partnerships & Resources

Partnerships

- R&D
  - Boston University – machine learning, anomaly detection, performance analysis
  - University of Central Florida – MPI stats, streaming analysis
  - UIUC – (New for FY19) network congestion characterizations, network congestion impact
  - NMSU – application monitoring, end-to-end architecture, performance metrics
  - Northeastern (New for FY20) – network contention measurement, overhead, and feedback
  - SNL (1400) – application monitoring: kokkos, PAPI
  - LANL – configuration flexibility and analysis usability
  - NCSA – holistic system and application production monitoring and performance analysis
  - NERSC – production requirements, SLURM features
- Cray
  - LDMS integrated into Shasta stack

Installations

- NNSA
  - LANL – ATS (Trinity), CTS, and other
  - LLNL – CTS
  - SNL – ART, CTS
  - TOSS stack
- Office of Science:
  - NERSC – Cori
- NSF
  - NCSA – Blue Waters
- Industry
  - Exxon Mobile
- Non-US
  - AWE
  - HLRS
- Misc Universities

Getting Involved

- LDMS **Users Group Conference** October 22 – 24, 2019

  University of Central Florida campus in Orlando, Florida

  https://sites.google.com/view/LDMSCON2019

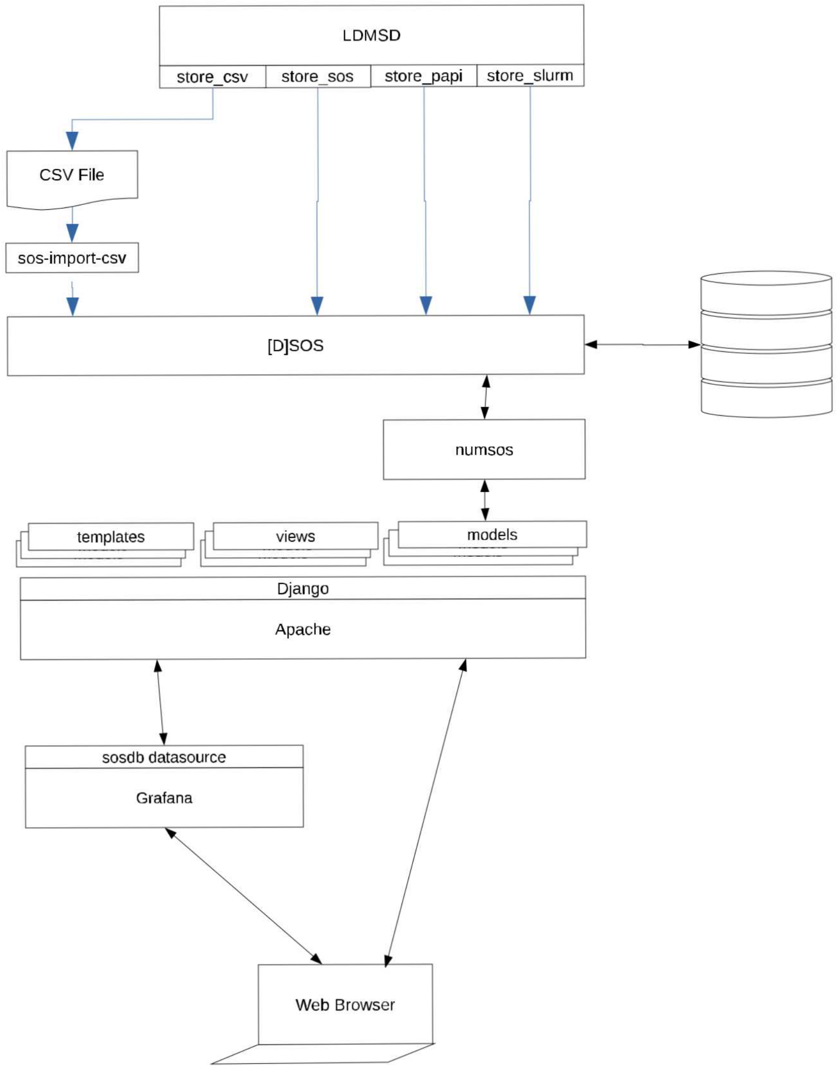
- Source code: https://github.com/ovis-hpc/ovis
- Discussions at: https://github.com/ovis-hpc/ovis/issues
- Participate in LDMS Users Group bi-weekly zoom meeting
  - Telecon info at: https://github.com/ovis-hpc/ovis/wiki/User-Group-Meeting-Notes
- Get help
  - Quick start build and configuration information at: https://github.com/ovis-hpc/ovis/wiki under "LDMS v4 Documentation" sidebar
  - Post problems to: https://github.com/ovis-hpc/ovis/issues
  - Publications: https://ovis.sandia.gov
  - Support services: contact tom@ogc.us

Extras

Abstract

The Lightweight Distributed Metric Service (LDMS) is a scalable lightweight monitoring framework that has been designed to meet the HPC monitoring goals of low CPU, memory, and network overhead. This talk will provide an in-depth description of LDMS including its low overhead mechanisms, basic configuration options with respect to data sampling, topology and storage, and analytics and visualization tools that Sandia has been developing. It will also present collection of IBM Power 9 specific information. LDMS provides the HPC community with an effective monitoring framework to enable users and staff to observe patterns of failures and performance anomalies, improve operational efficiency, achieve higher application performance, and inform the design of future systems. LDMS used in conjunction with appropriate analysis tools can enable effective feedback (human-in-the-loop or machine-in-the-loop) for improved performance.

Analytics Pipeline



Analysis Framework

- Scalable Object Store (SOS) optimized for **scalable storage and performant analysis** of HPC system and application information in flexible formats:
  - Efficient computations: SciPy & Numpy interfaces to access SOS object data as zero copy ndarrays suitable for arrays of data across components and time
- Continuous Analysis loop and/or post-processing supports multiple derived figures of merit

Continuous analysis and visualization of integrated system and application data, in numeric and log formats. Enables run time understanding and response.

Resources

Documentation (Building, Configuration)

- https://github.com/ovis-hpc/ovis/wiki

Source Code

- https://github.com/ovis-hpc/ovis
- git clone https://github.com/ovis-hpc/ovis.git

Publications:

- https://ovis.ca.sandia.gov

How you can contribute

- Write to ovis-help@sandia.gov

Support

- Bug reporting and questions: send email to ovis-help@sandia.gov
- Development services: contact tom@ogc.us
- Support services: contact tom@ogc.us

Supported platforms and networks

Linux support

- Rhel 6 and 7
- SLES 11 & 12
- Ubuntu

Vendor hardware platforms running supported software

- Cray XE6, XK and XC
- Generic Linux clusters
- IBM P8 & P9 (both big and little endian)

Transports

- Socket
- Cray ugni
  - Aries
  - Gemini
- RDMA
  - Infiniband
  - iWarp
  - libfabric

Build dependencies

Typical compute node environment

- Autoconf >=2.63, automake, libtool (collectively called autotools)
- OpenSSH-devel

End use hosts (monitor cluster, special aggregation hosts, etc.)

- Python
  - 2.6 with the argparse module
  - 2.7
- Swig
- Doxygen for documentation

LDMS Installation methods

Manually install using autoconf and automake

Typical deployments use RPMs