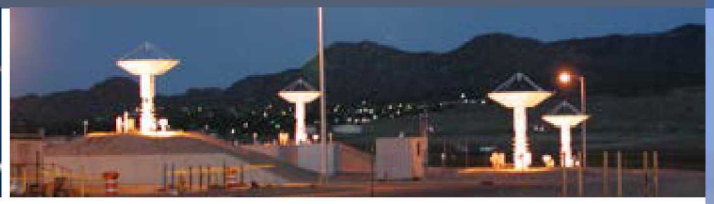
SAND2018-7101PE

Sandia National Laboratories

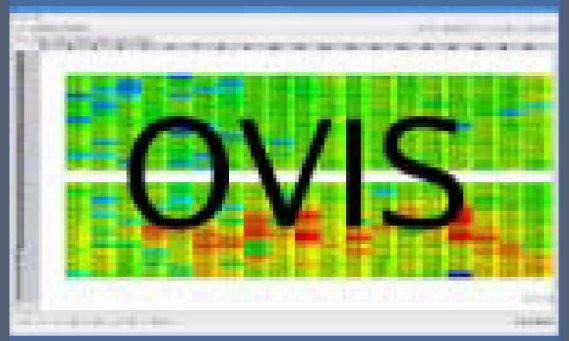
Capacity Systems Monitoring with LDMS and friends





PRESENTED BY

Benjamin Allan



U.S. DEPARTMENT OF ENERGY NNSA

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.

Outline

- What is LDMS and how does a deployment look like?
- Where is it in use in our community?
  - Production
  - Development
- What's new in data?
  - Beta ready: IB fabric counters
  - Still in development: App counters
- Can LDMS system profiling suggest change for apps?
  - Black box example
- Recent community contributions
- How we work

What LDMS is not

- All seeing/all knowing/all notifying
  - Focused on in-band hardware performance metrics collection to date
    - Dashboard capabilities in development by OVIS project
    - Pipelines to other analysis platforms by design
  - Log correlation development in collaboration with university partners
- Replacement for existing notification mechanisms to Net Ops Center
- Replacement for proprietary infrastructure data collection (e.g. H_2O)

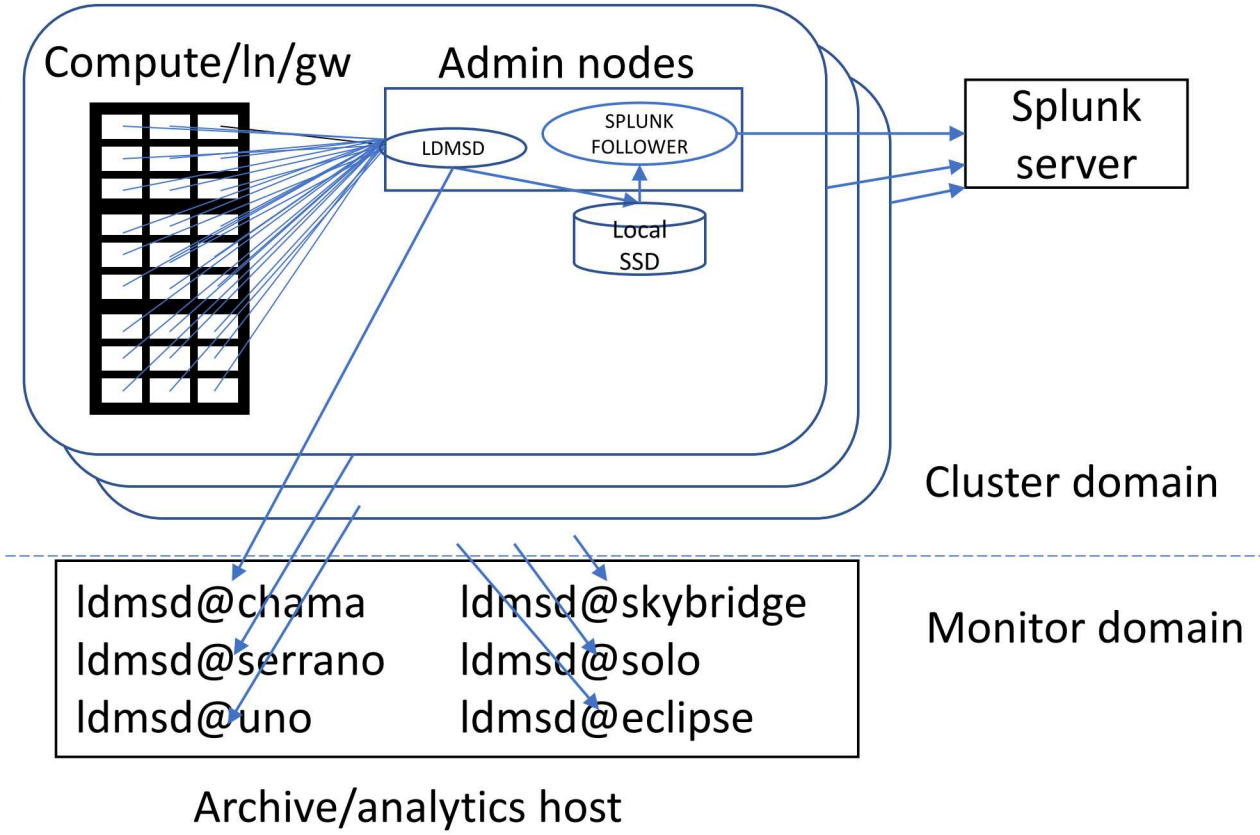
Lightweight distributed metric service

Low overhead, high rate system performance data collection

- Plug-ins for collecting, transporting, storing metrics

How:

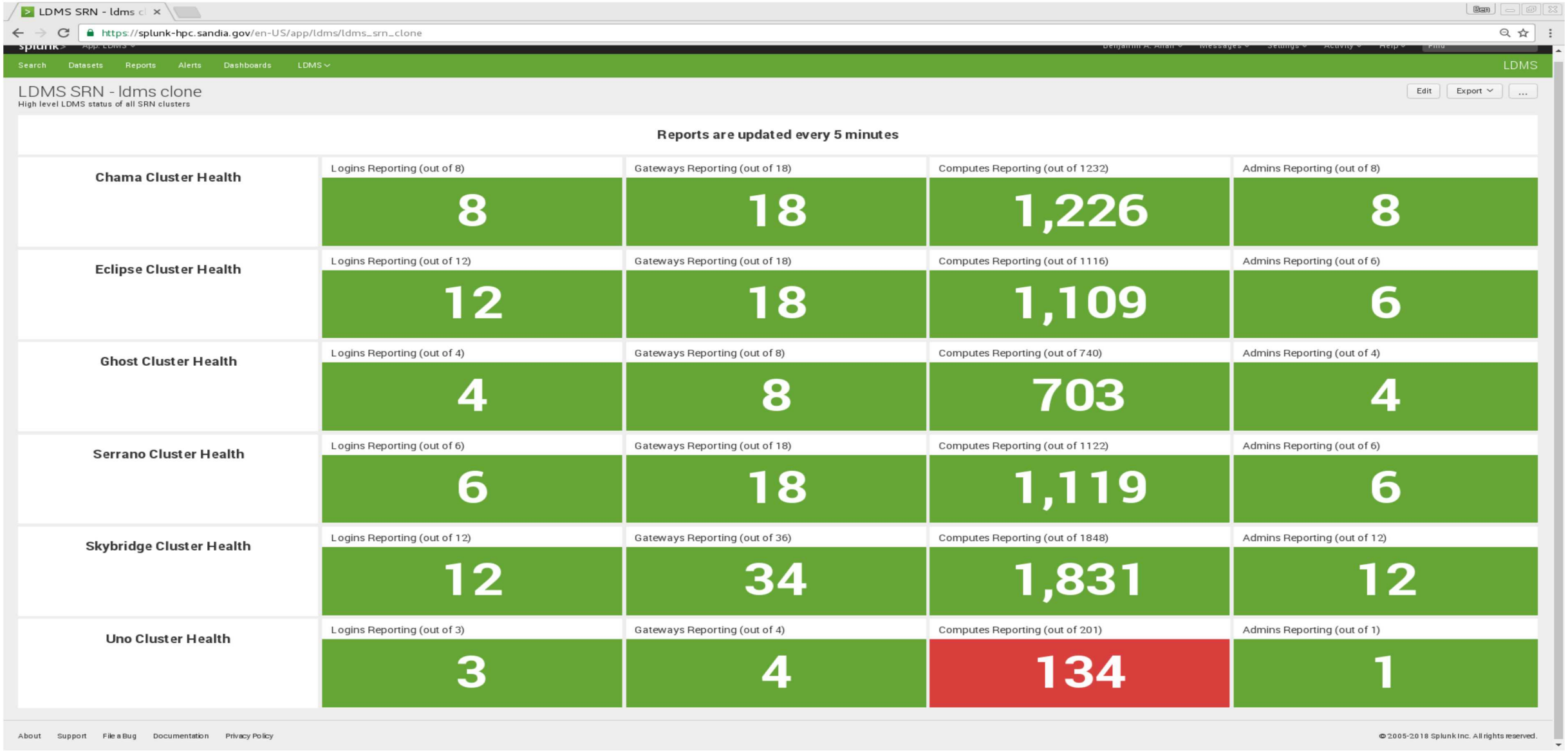
- LDMSD collection on all nodes (L0)
- L1 aggregation @ admin nodes
  - In cluster
  - SSD buffer, splunk store (optional)
- L2 aggregation
  - Outside clusters admin domain
  - Archive store
  - Web interfaces
- Robust to L2, store outages



Production deployments

- SNL
  - LDMS -> splunk & CSV archive
  - 6 unclassified production clusters
  - Full systemd support (per-cluster daemon instances at L2)
  - Libgenders managed configurations
- LANL
  - LDMS -> rabbitMQ & CSV archive
- LLNL
  - LDMS -> CSV

SNL dashboard example (status)



Data sets monitored on SNL TOSS clusters

- **Kernel**: /proc/interrupts, proc/meminfo, /proc/stat, /proc/vmstat,
- **Filesystems**: Lnet stats <pending>, Lustre client stats, NFSv3
- **Network**: InfiniBand HCA, OmniPath HFI, /proc/net/dev
- **Queue**: Job info (fed by SLURM)
- **Hardware**: EDAC (RAM errors) <pending>
- **Monitoring**: LDMSD (daemon self metrics)

Developmental SNL deployments

- Mayer (HPE, ARM64 (IB) testbed for Astra)
- Cts1x (CTS1 production testbed)
- Mutrino, Voltrino (Cray XC, like Trinity)
- Hazel (capacity monitoring analytics using TLCC2-like hardware)
- Shaun (open community monitoring analytics)

What’s new in data (still in development)

- *InfiniBand switch port counters*
- *Periodic application counters (MPI demo) (UCF)*
- PAPI (UNM)
- Sensors (temps, volts, fan speeds)
- Power
- ATA s.m.a.r.t.

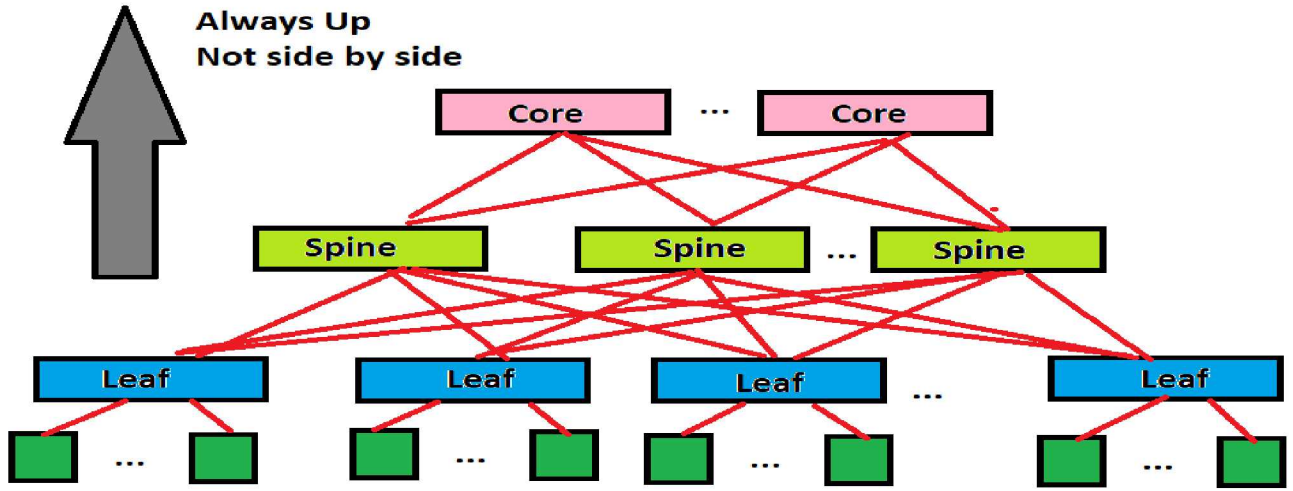
Taking your requests and contributions!

InfiniBand switch port monitoring (beta exp.)

Scale tested on SNL skybridge

268 Switches, 9648 Switch Ports

- Single collector rate possible:
  - once per 20 sec
- 10 collector rate possible:
  - 60Hz



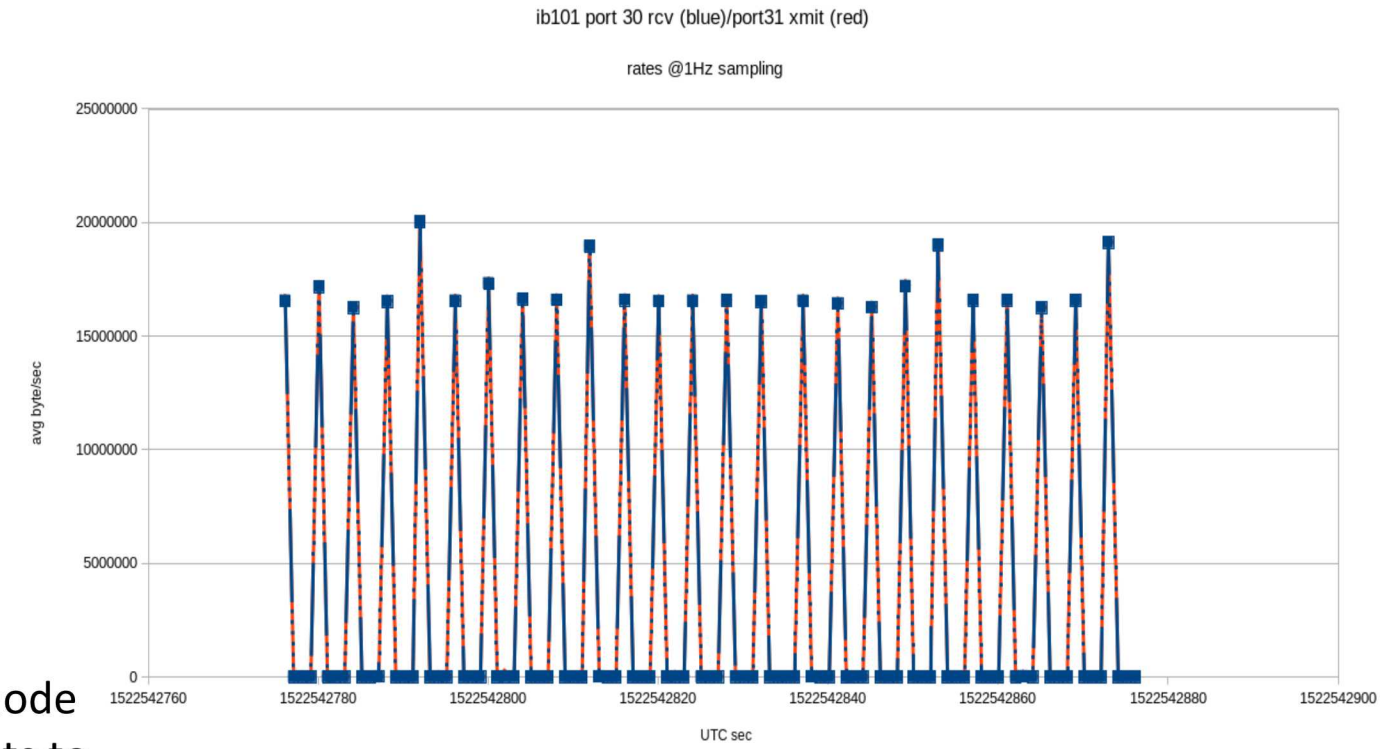
Comprehensive, Synchronous, High Frequency Measurement of InfiniBand Networks in Production HPC Systems, (SNL) Aguilar et al, Open Fabrics Alliance Workshop 2018.

Measuring Minimum Switch Port Metric Retrieval Time and Impact for Multi-layer InfiniBand Fabrics, Aguilar et al, IEEE Cluster, 2017.

IB switch results

We can match up data on paired ports

No interference with application at right rates



Preliminary recommendations for deployment:

- 1Hz: Use set of admin or IO nodes
- 10Hz: Collect 1 switch's ports per compute node
- To minimize duration, assign collection targets to minimize downward hops.

Things we learned about IB systems along the way

- Expect counter resets from subnet manager
- Expect dropped counter reset requests

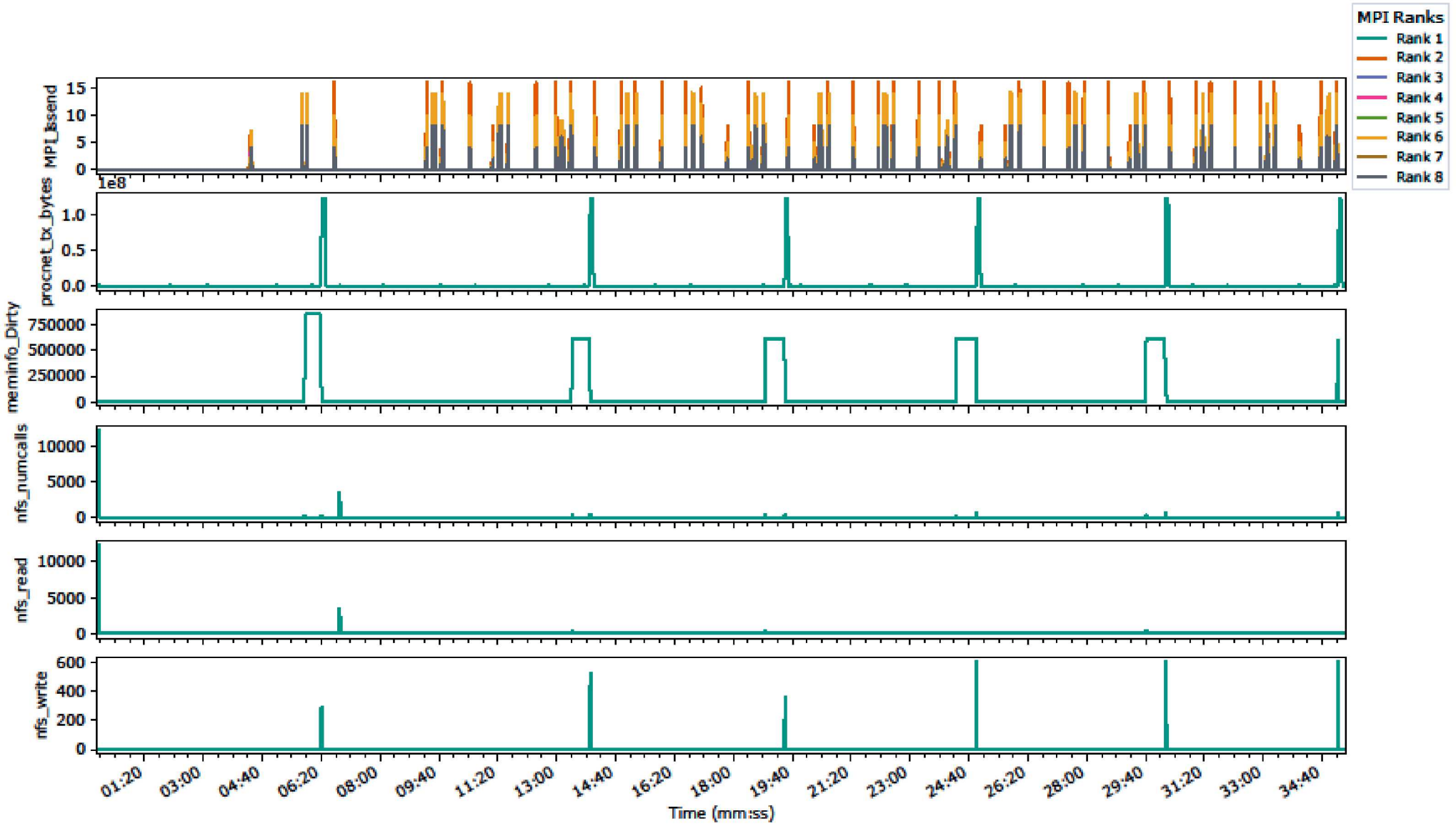
LDMS and Infiniband @ Sandia, Sipolev et al, Open Fabrics Alliance Workshop 2018.

Periodic application monitoring (alpha exp.)

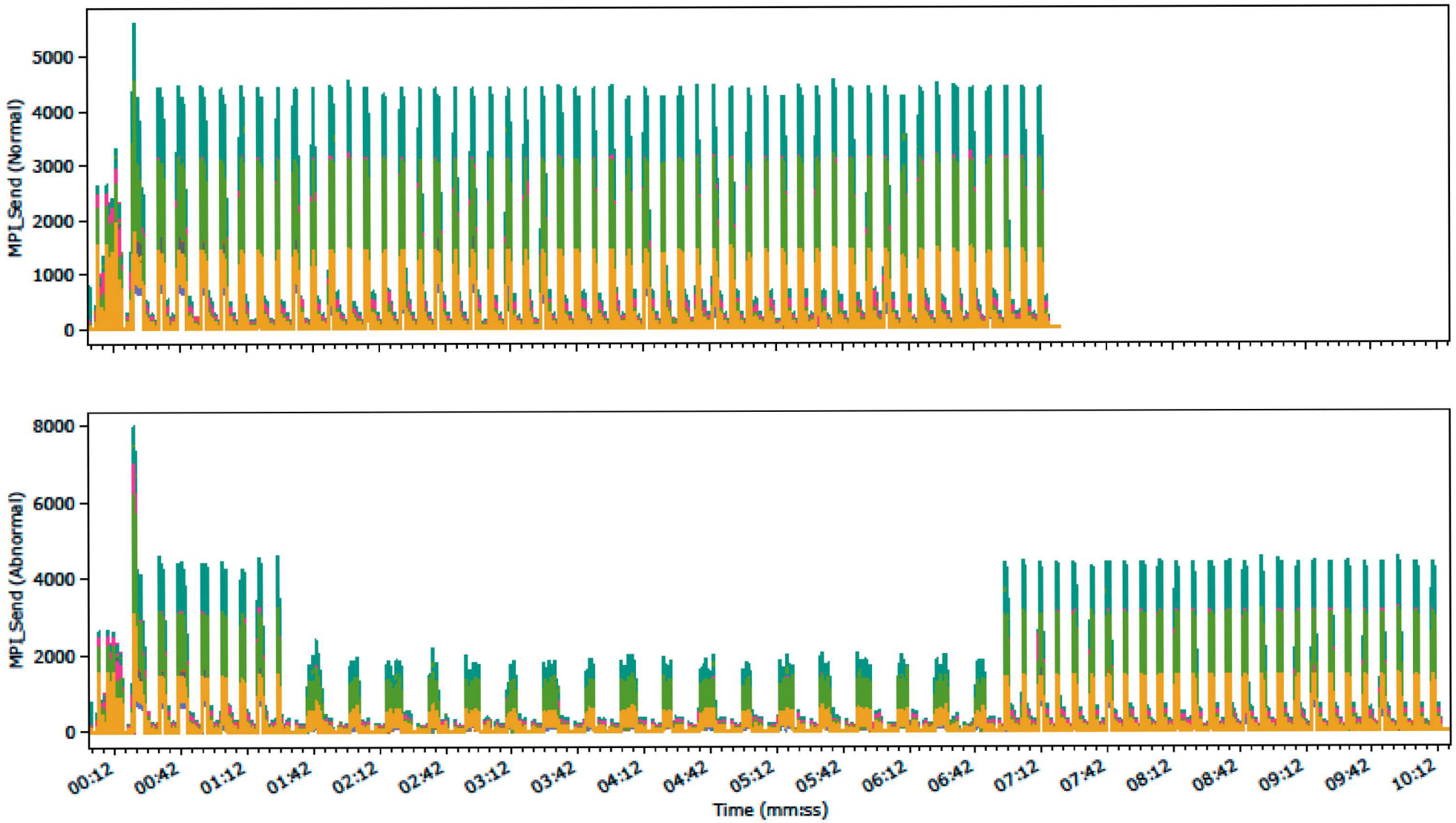
- Periodic sampling of application event counters
  - Via shmem
  - Fixed memory overhead for infinite runtime
  - Code instrumentation or LD_PRELOAD trick used.
- Quantified LDMS impact on KNL and Xeon with MPI example
  - NALU CFD
    - Overhead negligible at < 0.1Hz
    - Overhead negative at 1 Hz
    - Overhead 1-2% at 10 Hz

Production Application Performance Data Streaming for System Monitoring, (UCF)
Izadpanah et al, submitted.

MPI + system data: quicker understanding



MPI data: abnormality detection (cross-run)



Production application analysis

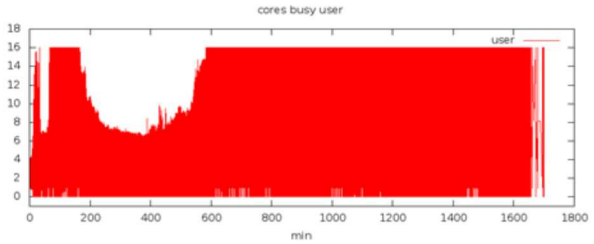
Black box – no source or library tinkering allowed

- Real application runtime is days/weeks
- OpenMP only, single node
- Zero runtime overhead from LDMS

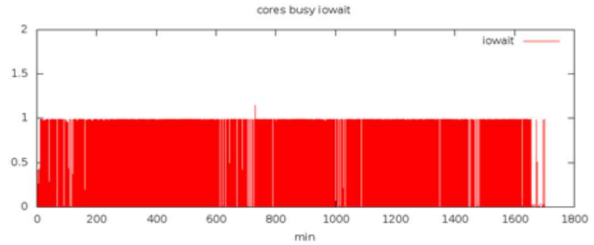
Results suggest trying 2 jobs/node allocation

1 day

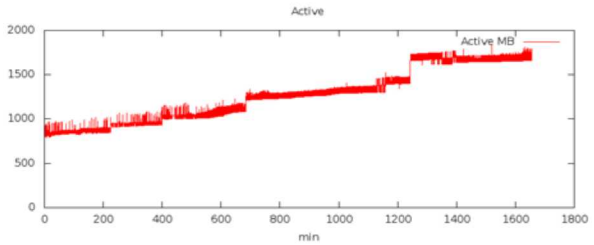
Cores busy user



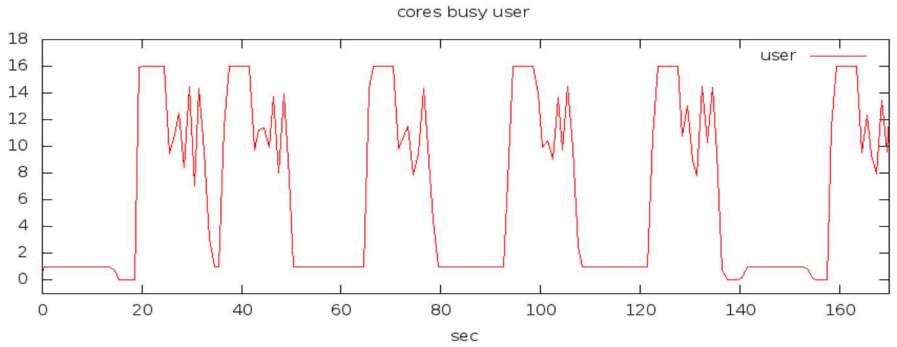
Cores in iowait

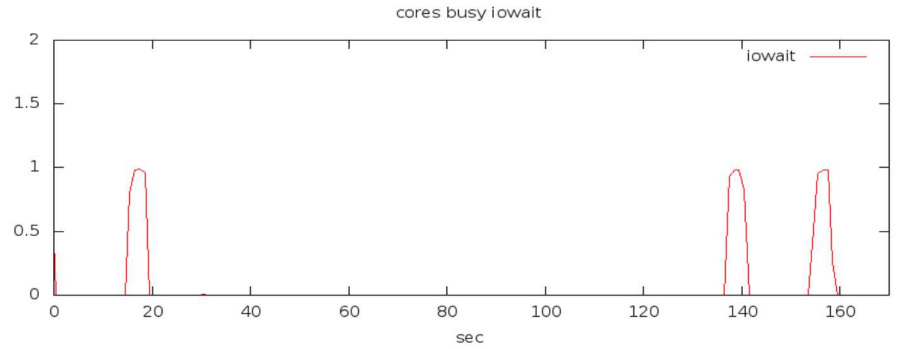


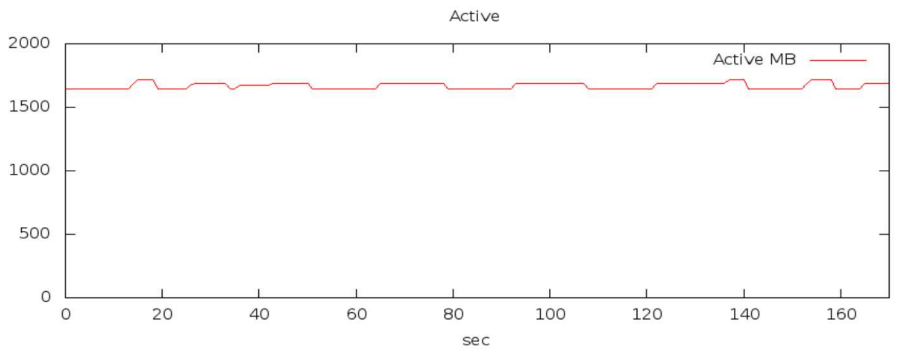
RAM active



2 minutes (zoomed)







Recent community contributions

Cray

- Plugins for monitoring Cray devices
- Build system improvements:
  - Support for third party plugin projects
  - Support for fully relocatable TOSS RPMs

Development process

Internal gitlab (Open Grid Computing, university and lab partners)

External github

Tests/examples suite included in source, browsable on github

SNL Jenkins build/package testing (CLE6, TOSS 3) of public releases

Public releases for TOSS

Pre-release test versions for collaborators

Changes based on feedback from production admins

The End

See also:

https://ovis.ca.sandia.gov

Some details for the curious

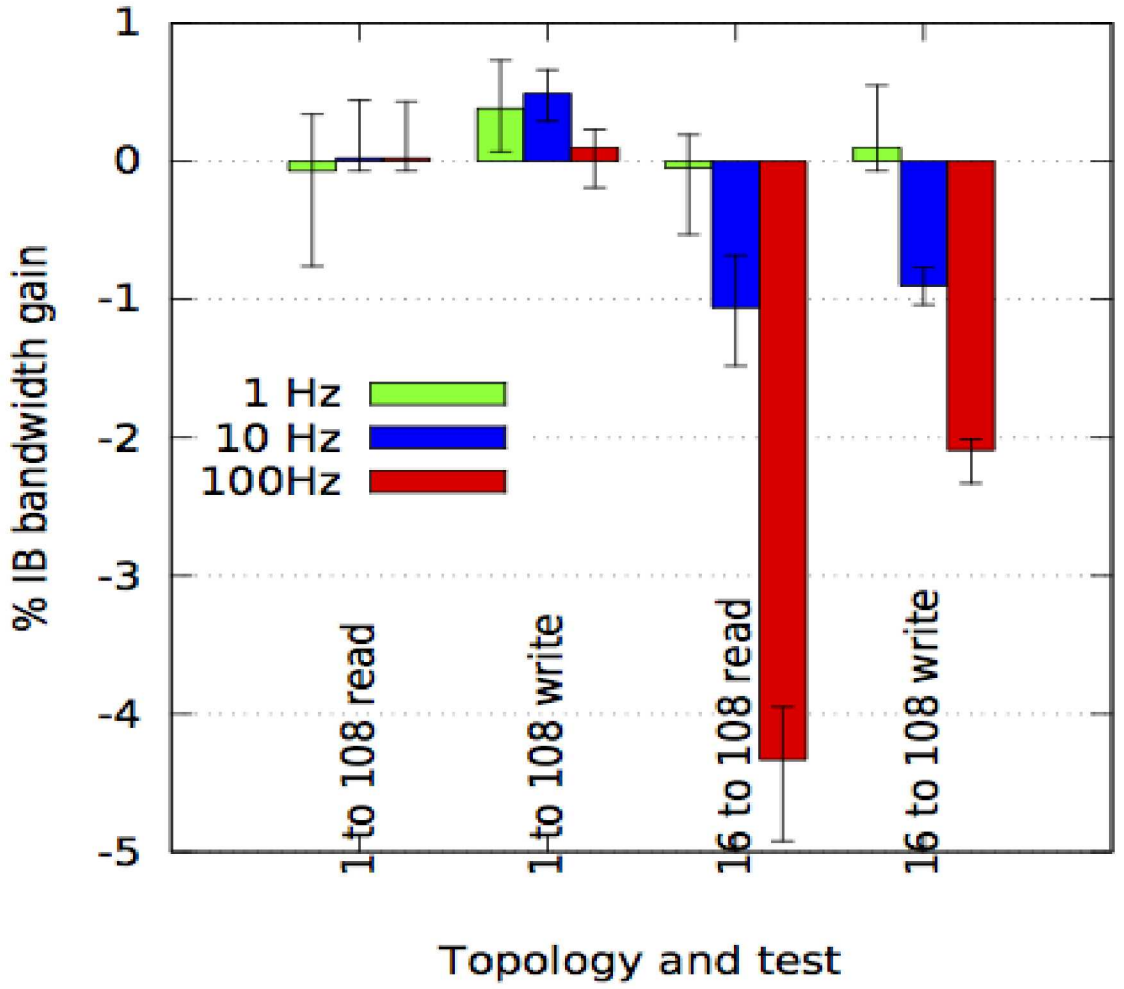
- IB switch sampling bandwidth effects
- NALU MPI counter runtime effects
  - Sampler plugins active impact
  - Sampling frequency impact

IB switch measurement bandwidth impact

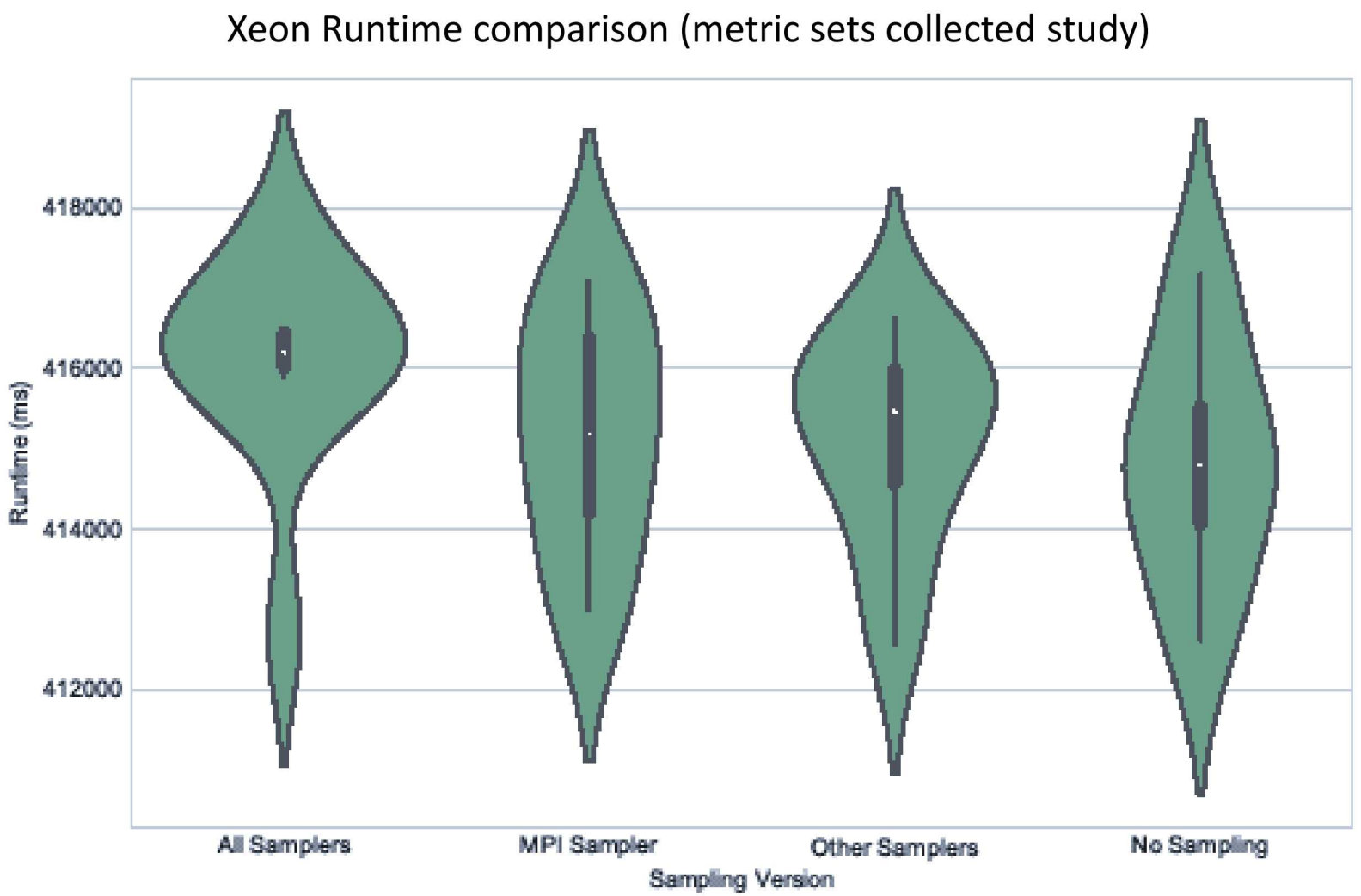
We see no significant read/write bandwidth test overhead (negative % is bad) when collecting without redundancy or at 1 Hz redundantly.

We get significant bandwidth loss for fast, highly redundant collection.

Port data queries travel on VL15, so these results are expected: at some point we distract the switch processor.

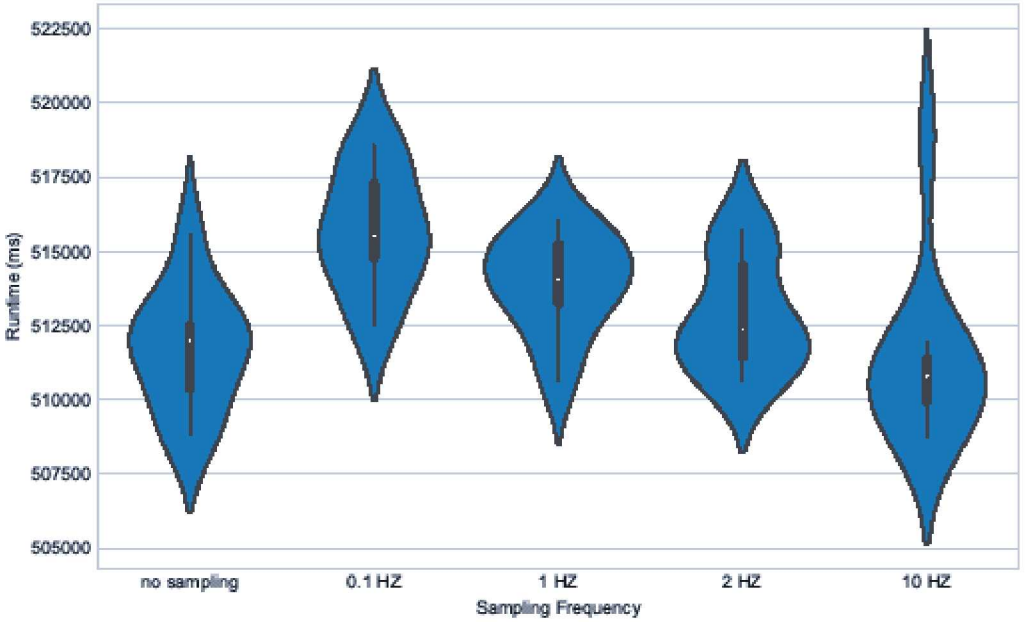


Nalu overhead of MPI counters



NALU overhead of MPI counters

KNL Runtime comparison (frequency study)



Xeon Runtime comparison (frequency study)