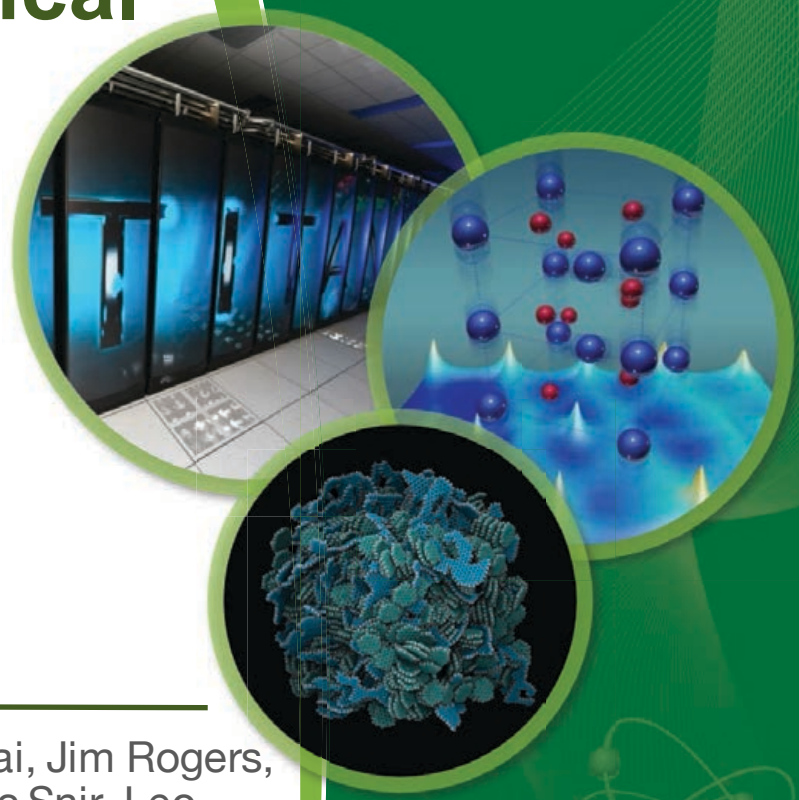
Toward Lightweight, Actionable Analytical Tools Based on Statistical Learning for Efficient System Operations



Devesh Tiwari

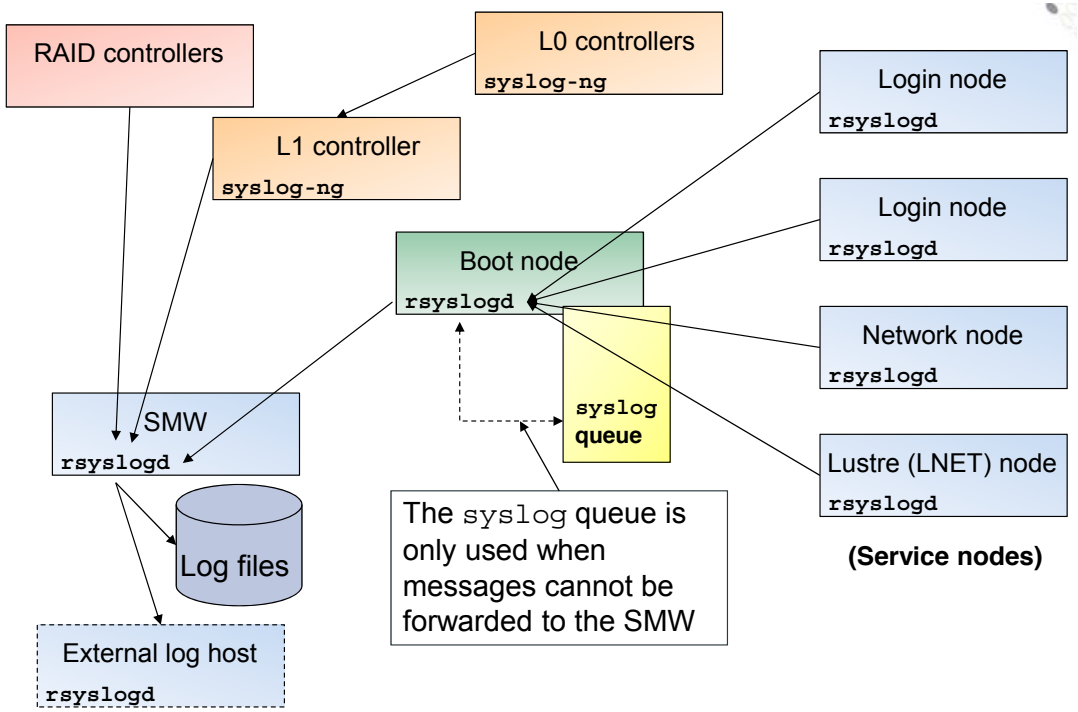
Oak Ridge National Laboratory

tiwari@ornl.gov

Thanks to Saurabh Gupta, Christian Engelmann, Sudharshan Vazhkudai, Jim Rogers, Bin Nie, Evgenia Smirni, Franck Cappello, Rinku Gupta, Sheng Di, Marc Snir, Leo Bautista-Gomez, Swann Perarnau, Nathan Debardeleben, Paolo Rech, Don Maxwell, Changhee Jung, Martin Schulz, Ignacio Laguna, and many more smart folks!

This work used the resources of the Oak Ridge Leadership Computing Facility, located in the National Center for Computational Sciences at the Oak Ridge National Laboratory, which is managed by UT Battelle, LLC for the U.S. Department of Energy, under the contract No. DEAC05-00OR22725.

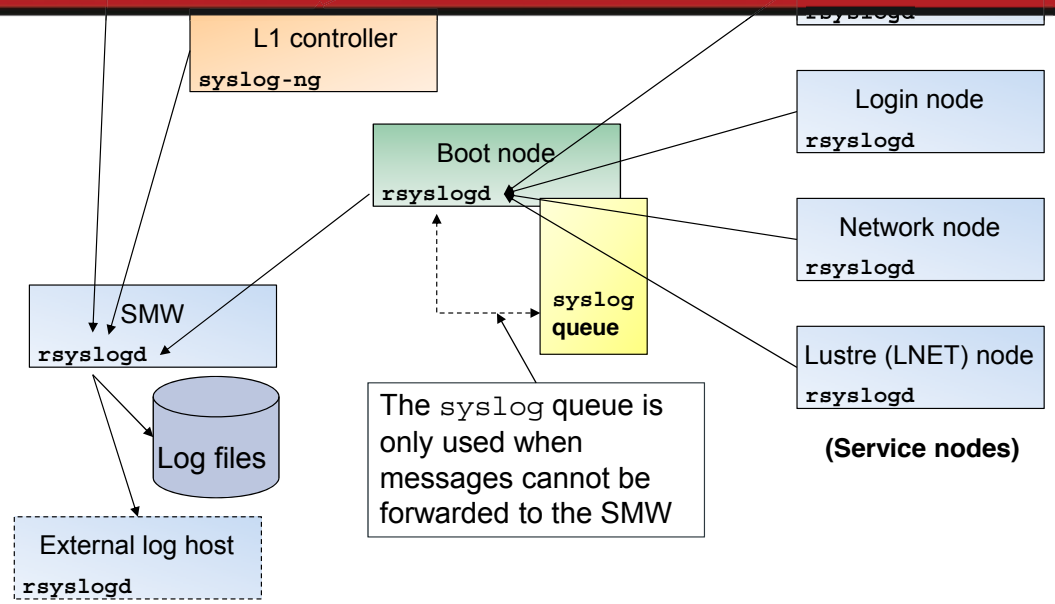
System Generated Data for ModSim Efforts



Courtesy: Rick Slick CUG 2013 Tutorial Slides

System Generated Data for ModSim Efforts

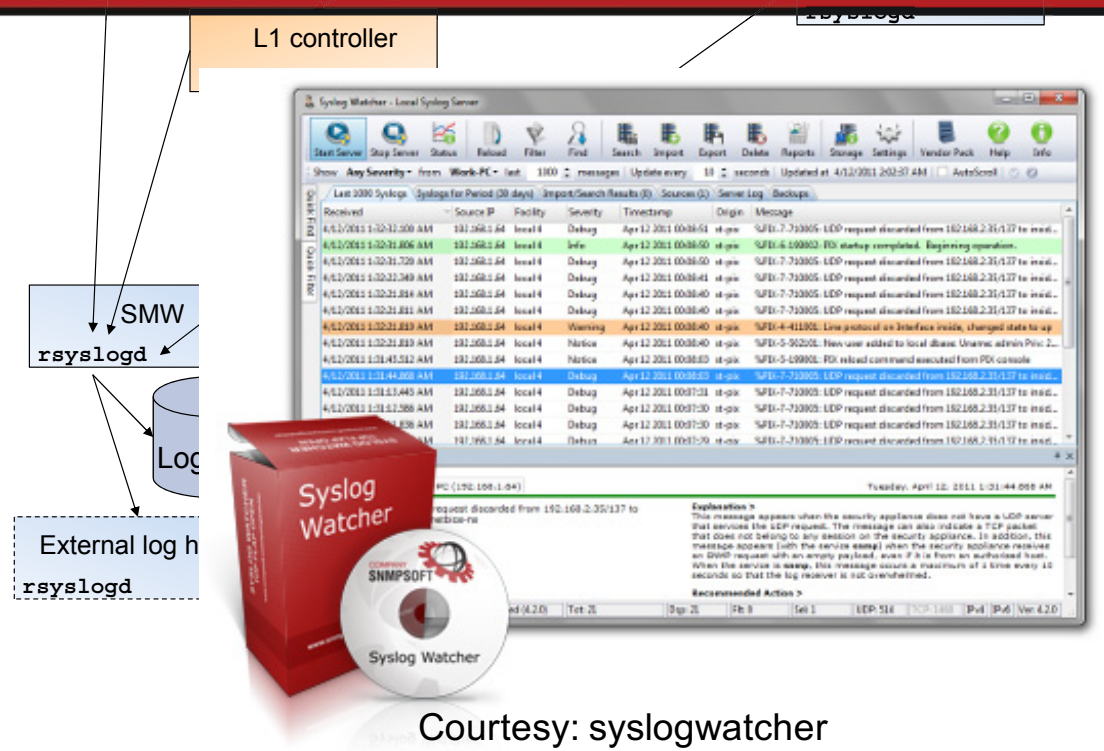
Hard to store and manage



Courtesy: Rick Slick CUG 2013 Tutorial Slides

System Generated Data for ModSim Efforts

Hard to store and manage

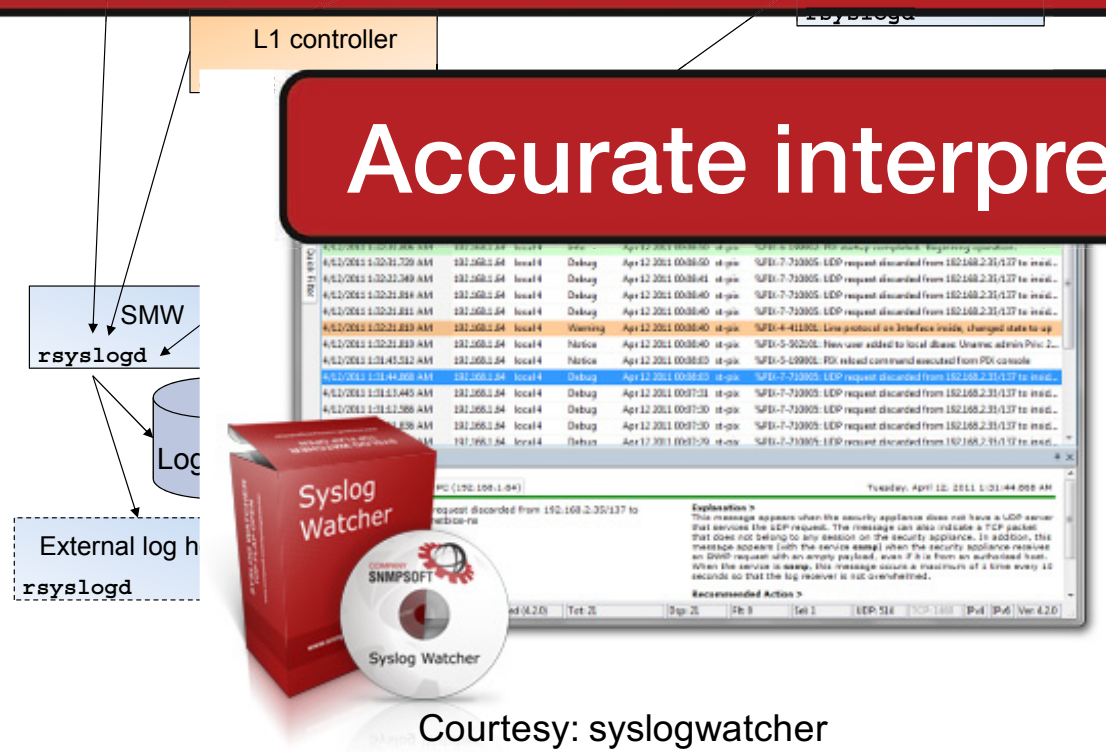


Courtesy: syslogwatcher

System Generated Data for ModSim Efforts

Hard to store and manage

Accurate interpretation hard

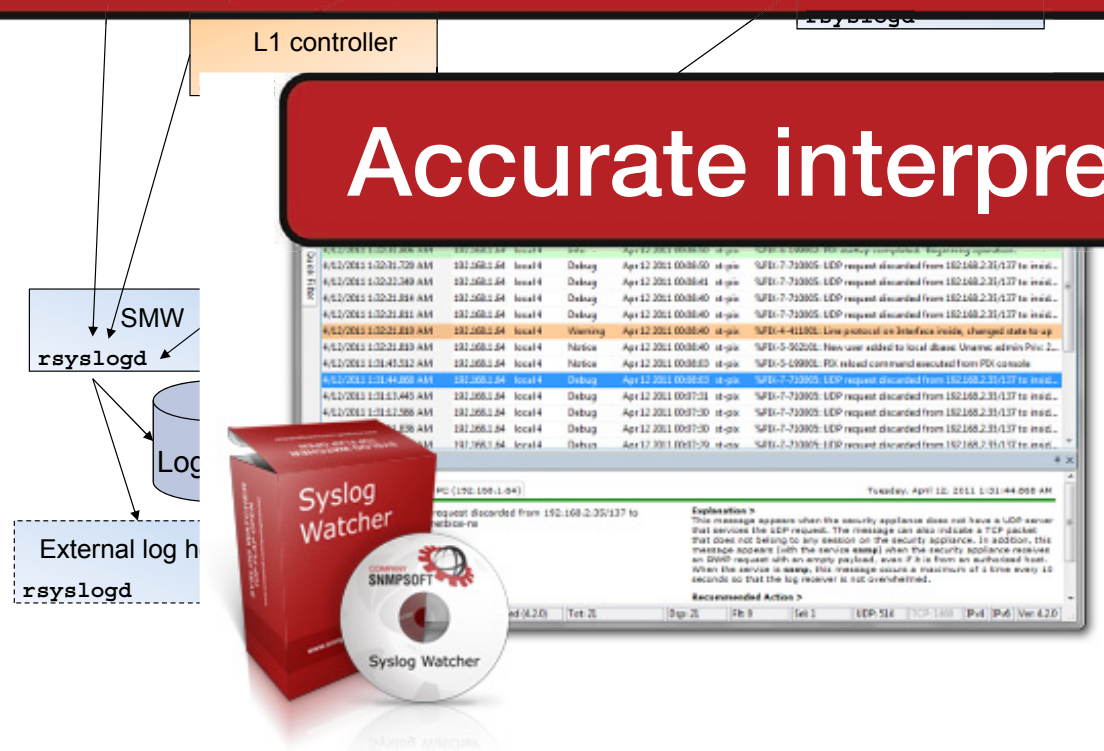


Courtesy: syslogwatcher

System Generated Data for ModSim Efforts

Hard to store and manage

Accurate interpretation hard

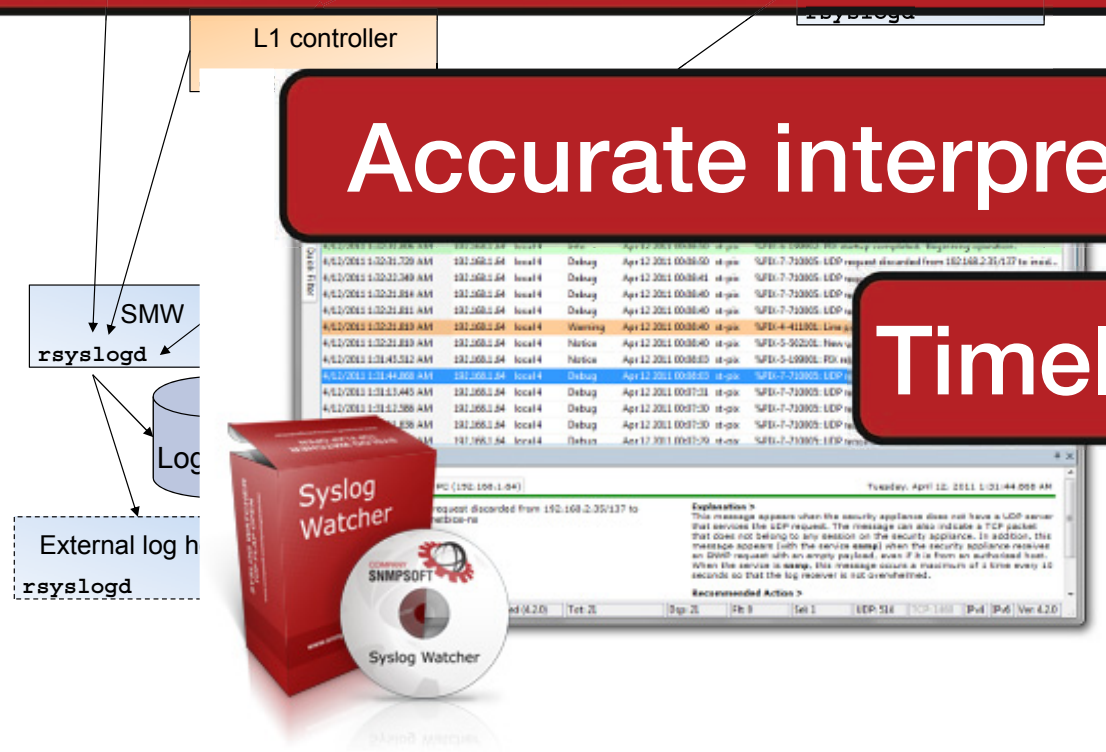


System Generated Data for ModSim Efforts

Hard to store and manage

Accurate interpretation hard

Timely processing and analysis

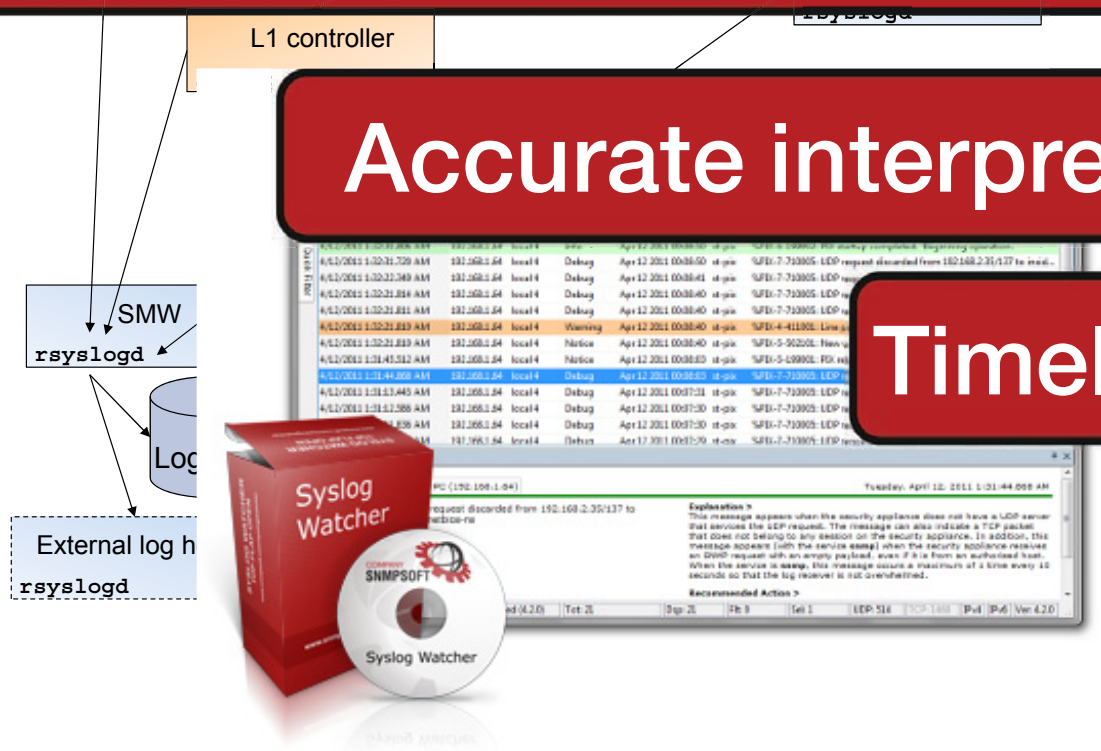


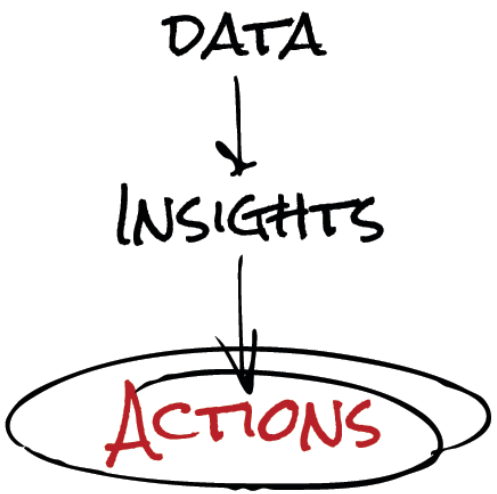
System Generated Data for ModSim Efforts

Hard to store and manage

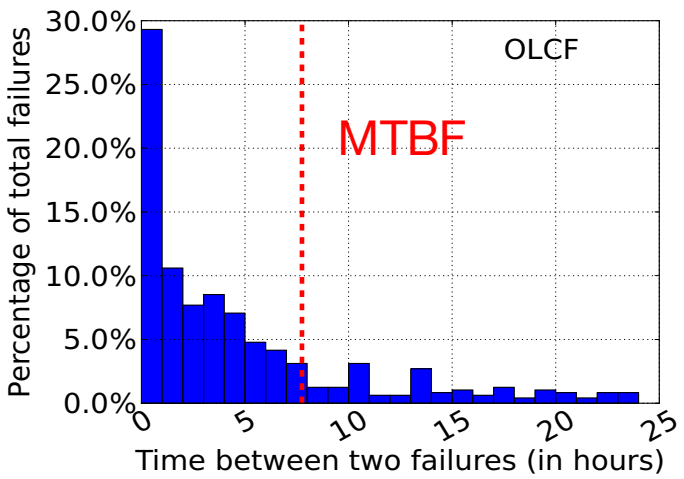
Accurate interpretation hard

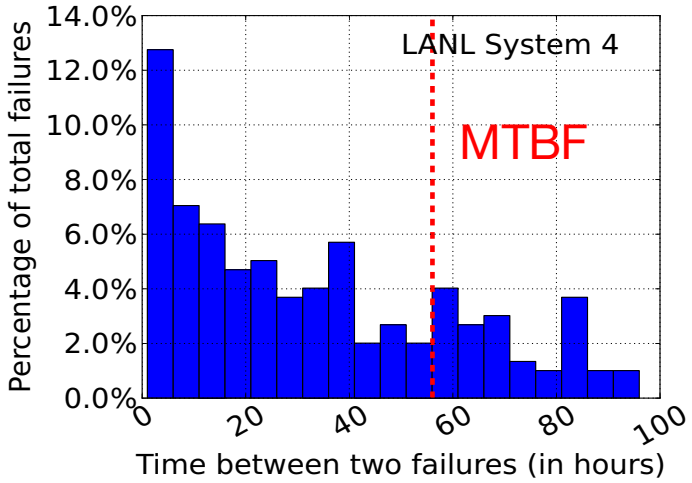
Timely processing and analysis

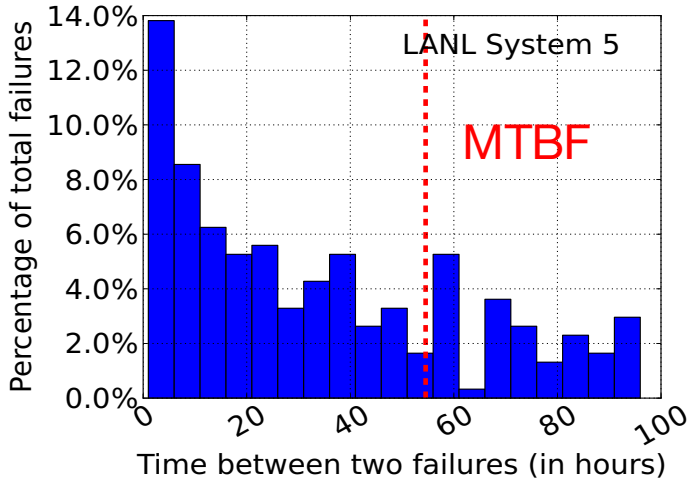




System failures exhibit temporal and spatial locality.

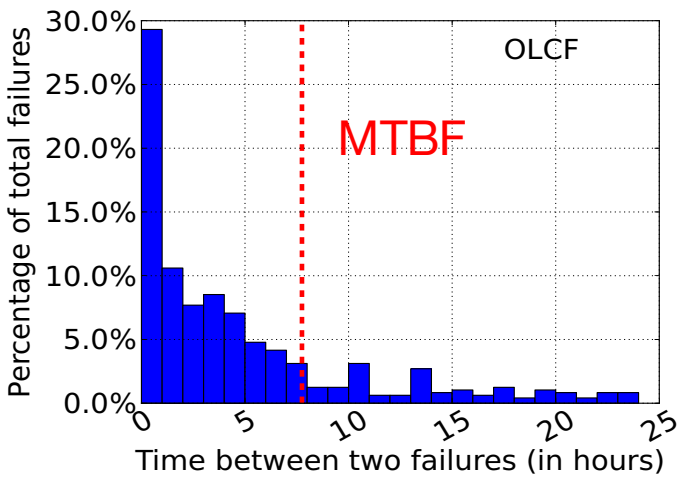


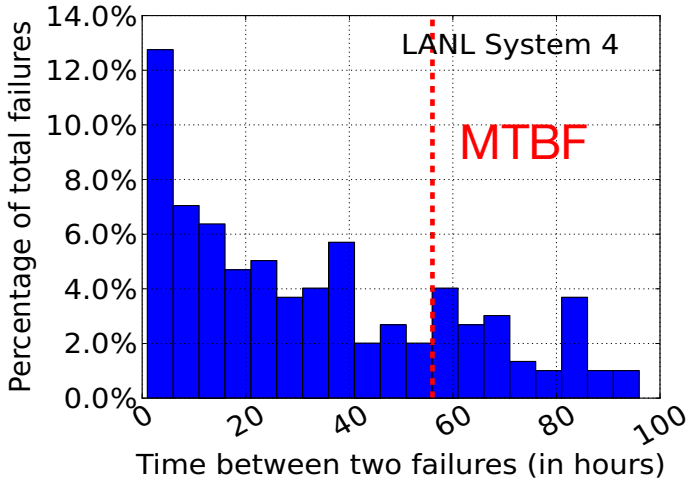


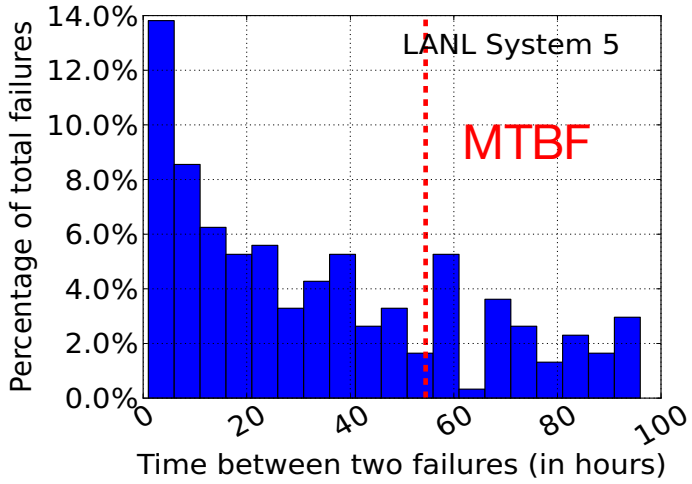


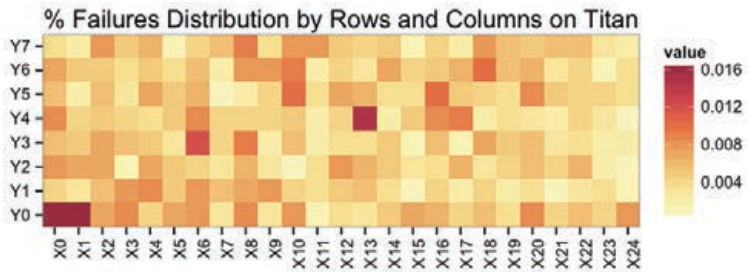
Observation holds true across systems and failure types, consistently across long range of periods.

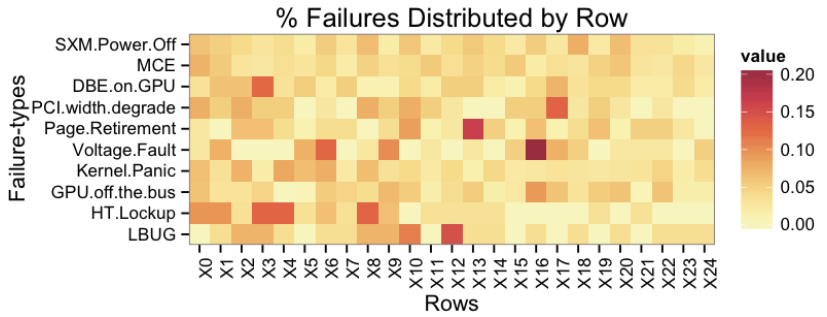
System failures exhibit temporal and spatial locality.

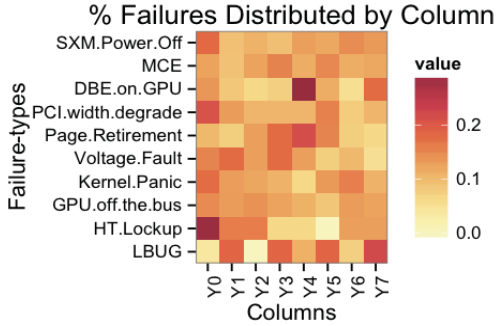






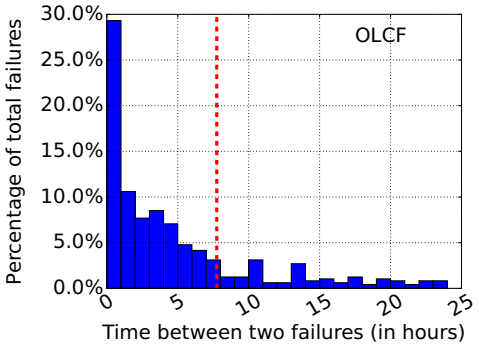






Observation holds true across systems and failure types, consistently across long range of periods.

Lazy Checkpointing Technique



S Simulation/Computation

C Checkpoint

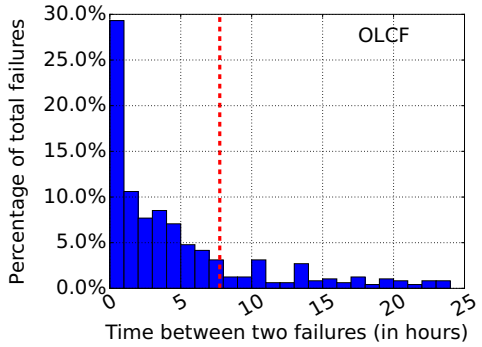
L Lost Work

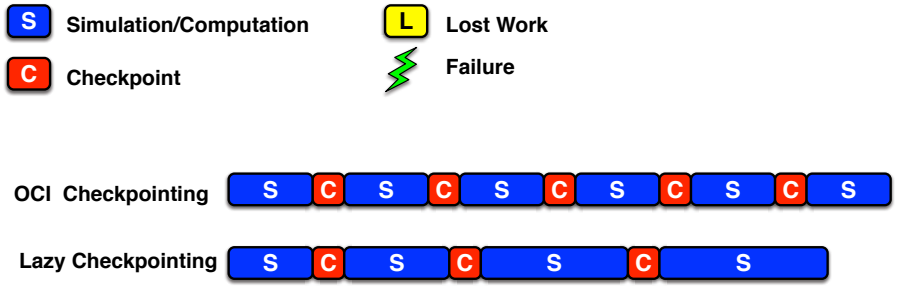
Failure

OCI Checkpointing

Lazy Checkpointing

Lazy Checkpointing Technique





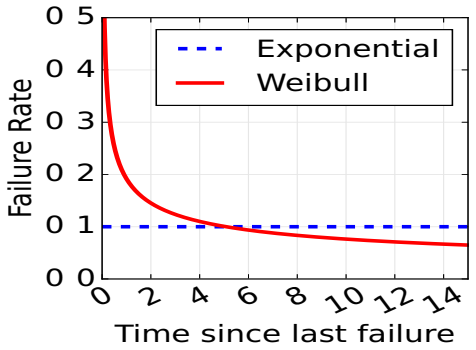
Weibull Distribution

Failure Rate $\frac{k}{\lambda}(\frac{t}{\lambda})^{k-1}$

t Time since last failure

λ Scale parameter

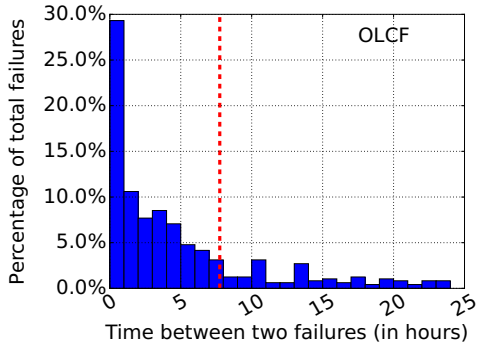
k Shape parameter

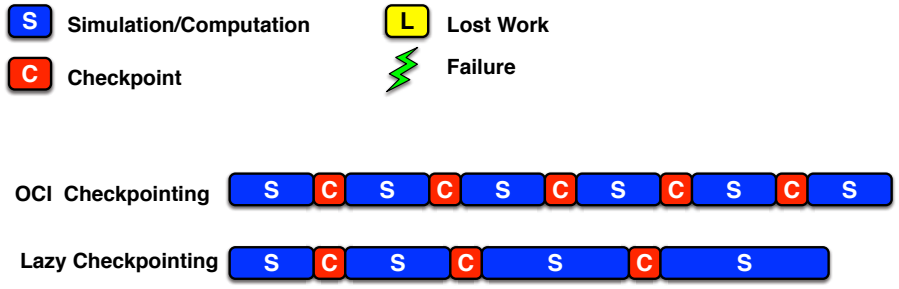


$$\alpha_{oci} = \sqrt{\beta^2 + \frac{\beta\gamma}{\epsilon} + \frac{M\beta}{\epsilon}}$$

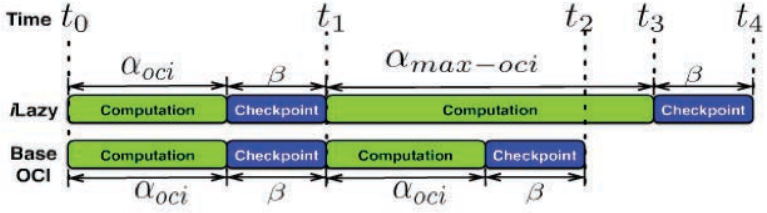
$$\alpha_{lazy} = \alpha_{oci}\left(\frac{t}{\alpha_{oci}}\right)^{(1-k)}$$

Lazy Checkpointing Technique





Key is to balance the trade-off between reduction in checkpointing overhead and possible increase in the waste work



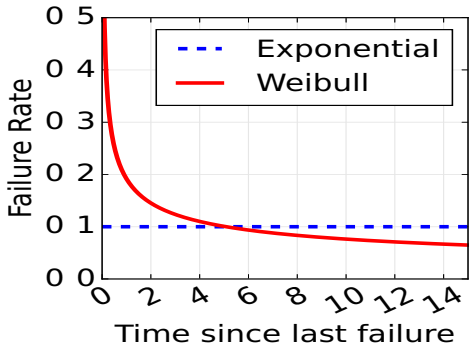
Weibull Distribution

$$\text{Failure Rate } \frac{k}{\lambda}\left(\frac{t}{\lambda}\right)^{k-1}$$

t Time since last failure

λ Scale parameter

k Shape parameter



$$\alpha_{oci} = \sqrt{\beta^2 + \frac{\beta\gamma}{\epsilon} + \frac{M\beta}{\epsilon}}$$

$$\alpha_{lazy} = \alpha_{oci}\left(\frac{t}{\alpha_{oci}}\right)^{(1-k)}$$

$$\text{performance loss} = (\alpha_{max-oci} - \alpha_{oci})(e^{-(\frac{t_2}{\lambda})^k} - e^{-(\frac{t_4}{\lambda})^k})$$

$$= (\alpha_{max-oci} - \alpha_{oci})(e^{-(\frac{2(\alpha_{oci}+\beta)}{\lambda})^k} - e^{-(\frac{\alpha_{max-oci}+\alpha_{oci}+2\beta}{\lambda})^k})$$

$$\text{performance gain} = \beta e^{-(\frac{t_3}{\lambda})^k}$$

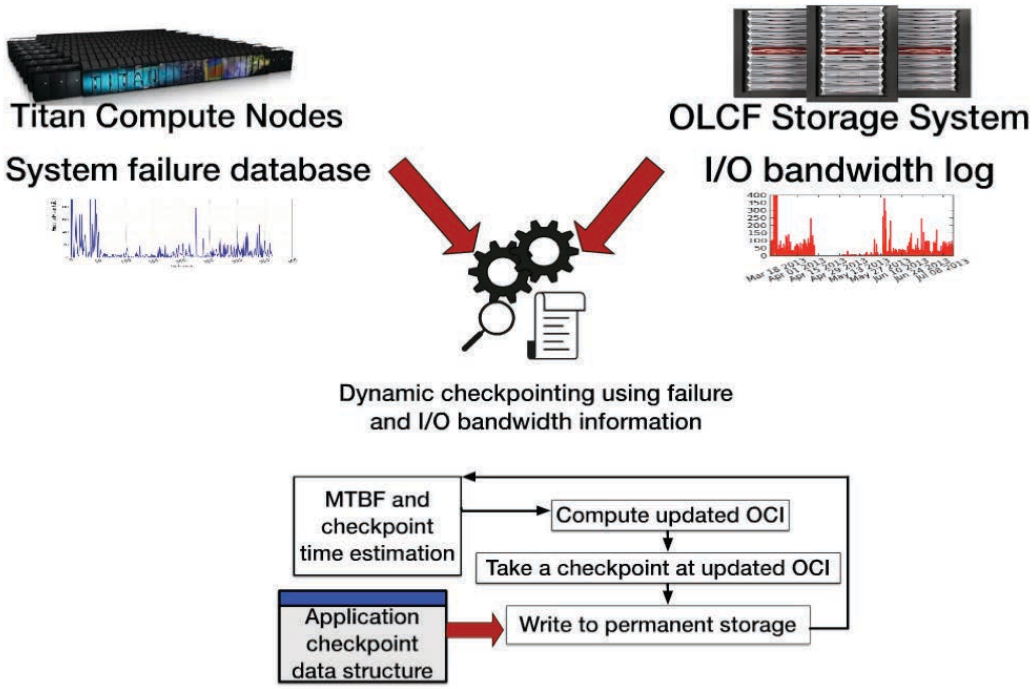
$$\beta e^{-(\frac{\alpha_{max-oci}+\alpha_{oci}+\beta}{\lambda})^k} = (\alpha_{max-oci} - \alpha_{oci})e^{-(\frac{2(\alpha_{oci}+\beta)}{\lambda})^k}$$

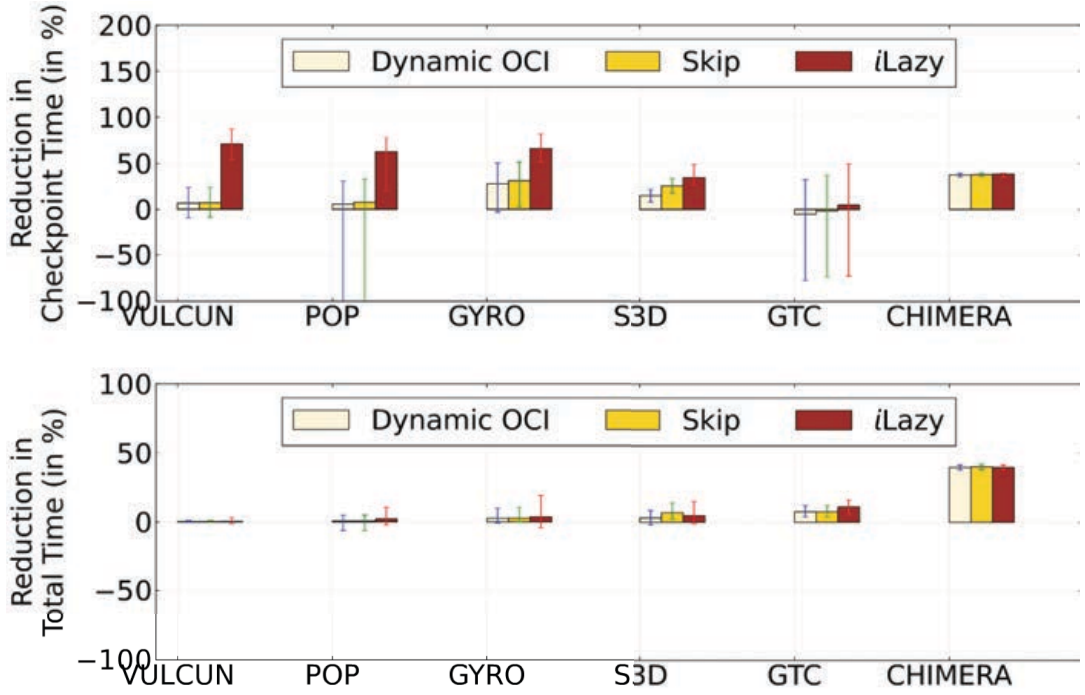
$$-(\alpha_{max-oci} - \alpha_{oci})e^{-(\frac{\alpha_{max-oci}+\alpha_{oci}+2\beta}{\lambda})^k}$$

Refer to the paper for model validation and simulation results

[DSN 2014] Lazy Checkpointing: Exploiting Temporal Locality in Failures to Mitigate Checkpointing Overheads on Extreme-Scale Systems
Devesh Tiwari, S Gupta, S Vazhkudai, IEEE/IFIP Int'l Conference on Dependable Systems and Networks (DSN), 2014.

Lazy Checkpointing Technique



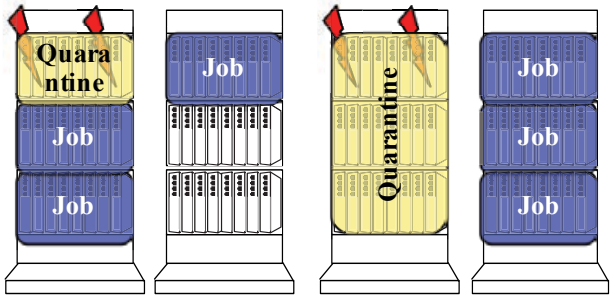


[DSN 2014] Lazy Checkpointing: Exploiting Temporal Locality in Failures to Mitigate Checkpointing Overheads on Extreme-Scale Systems
Devesh Tiwari, S Gupta, S Vazhkudai, IEEE/IFIP Int'l Conference on Dependable Systems and Networks (DSN), 2014.

Quarantine Job Scheduling Technique

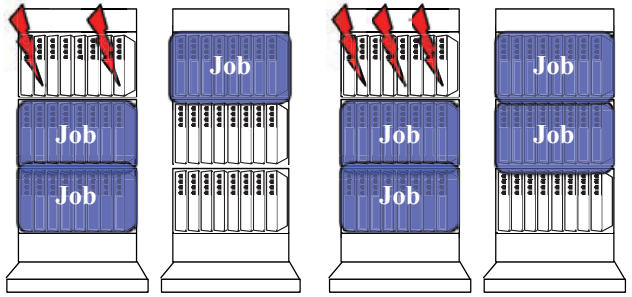
Idea: On job restart or a new job allocation a fraction of compute capacity is not utilized (quarantined)

Quarantine Granularity



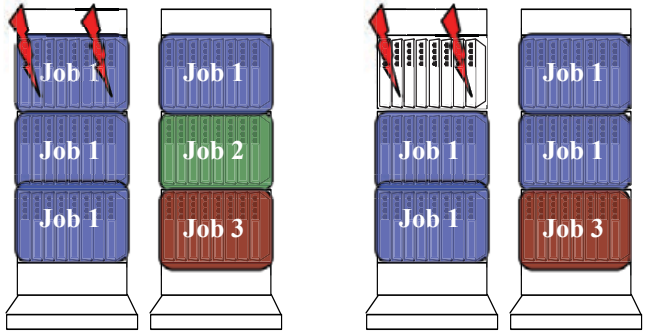
Fraction of avoided system failures versus compute resource waste

Quarantine Time Duration



Diminishing returns on the number of avoided failures

System Utilization vs. Reliability



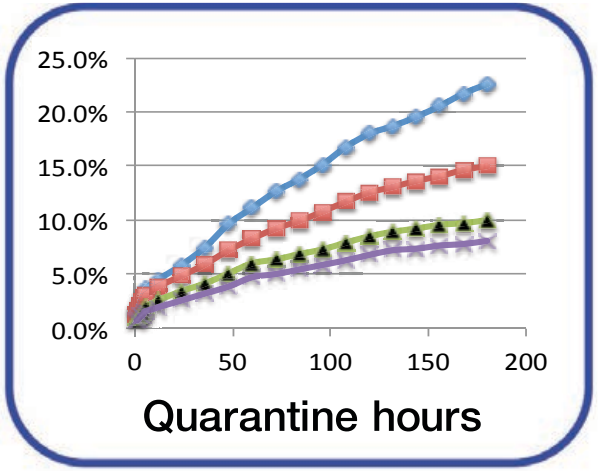
Trading-off lower system utilization for improved reliability

[DSN 2015] Understanding and Exploiting Spatial Properties of System Failures on Extreme-Scale HPC Systems

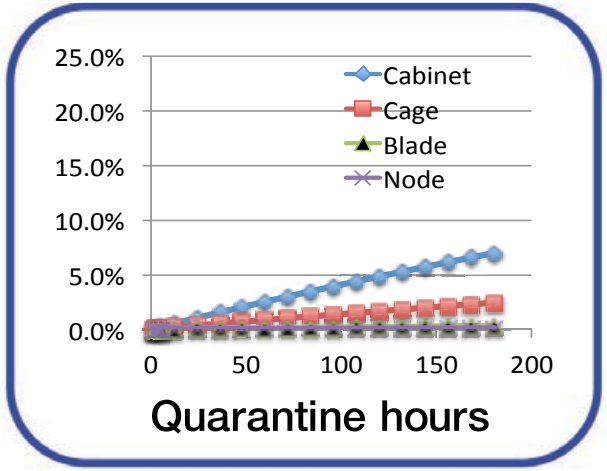
Saurabh Gupta, Devesh Tiwari, Chris Jantzi, Jim Rogers, Don Maxwell, IEEE/IFIP Int'l Conf on Dependable Systems and Networks (DSN), 2015.

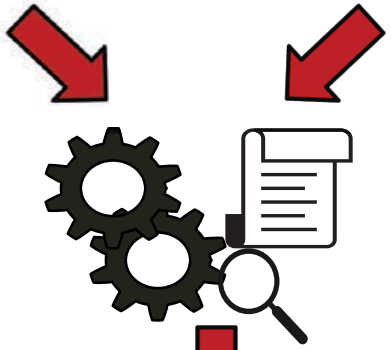
Quarantine Job Scheduling Technique

System Reliability
Fraction of failures avoided



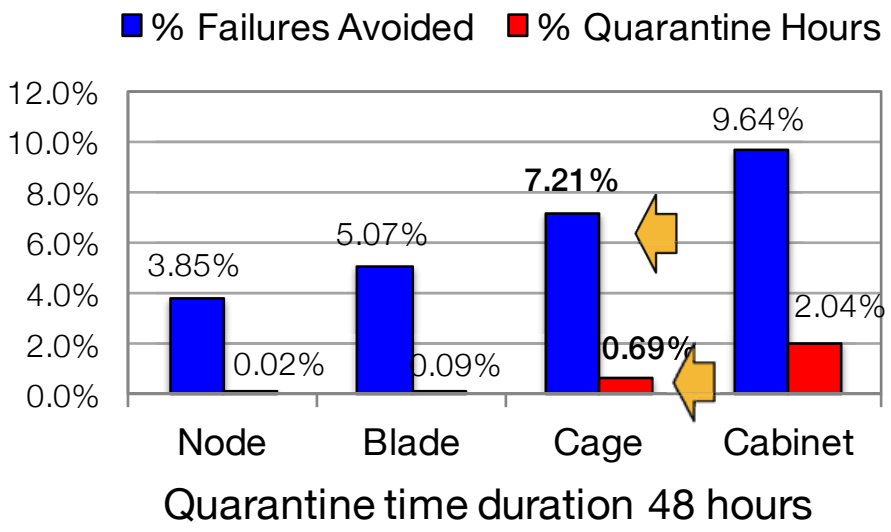
System Utilization
Quarantine node hours





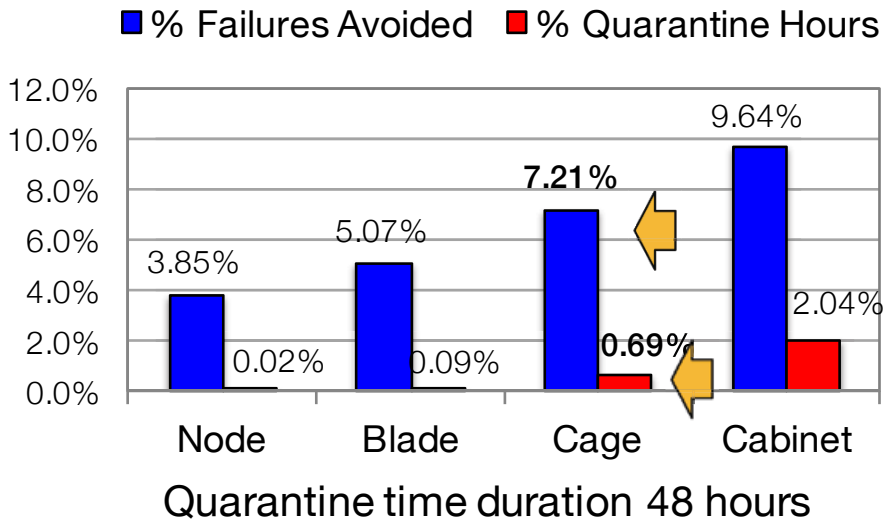
Feedback to the job scheduler

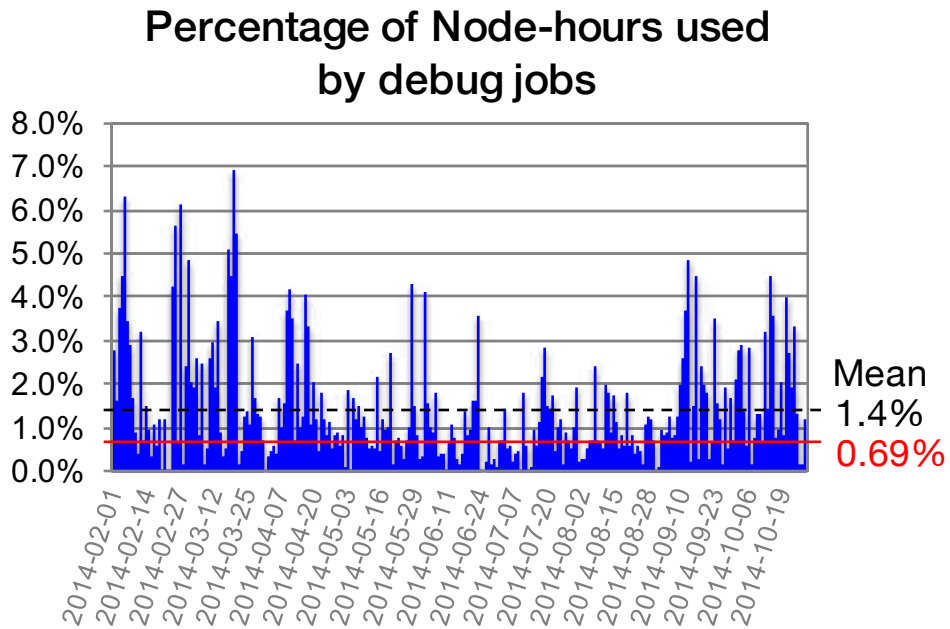
Quarantine Job Scheduling Technique



Quarantine time duration 48 hours

Quarantine Job Scheduling Technique

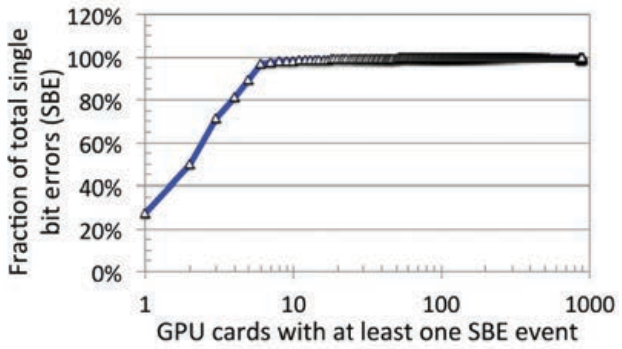


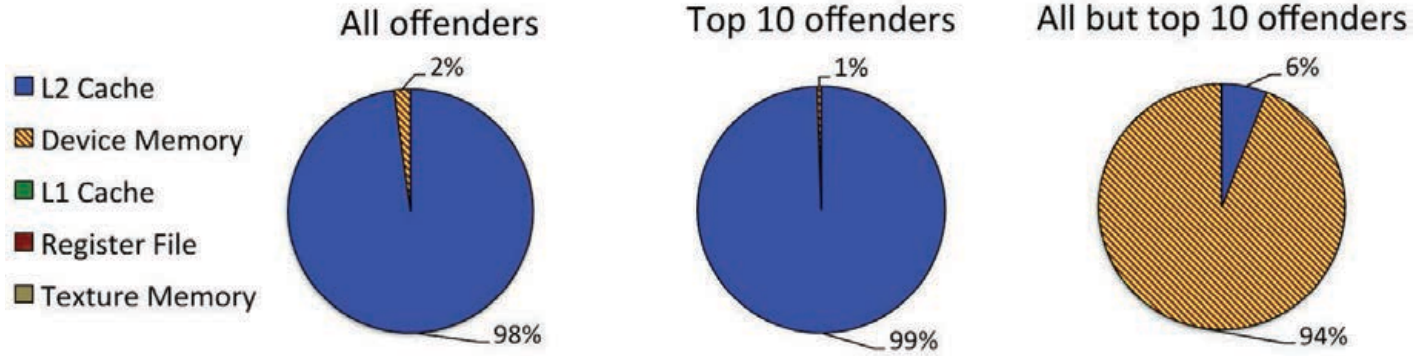


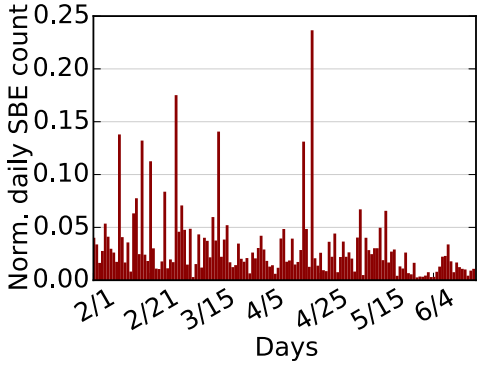
Significant fraction of failures can be avoided from interrupting production applications

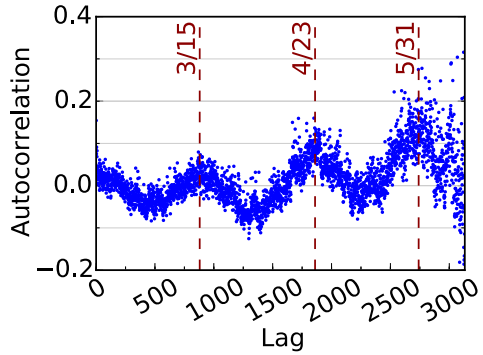
Debug or non-production jobs can be scheduled on quarantine nodes

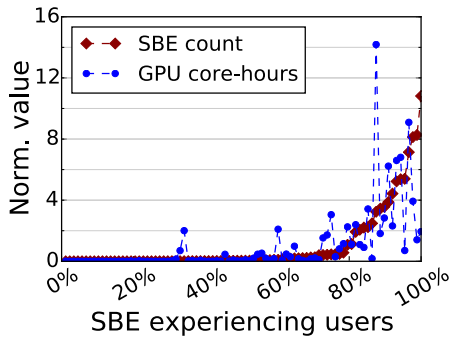
Feedback Loop for ModSim Community

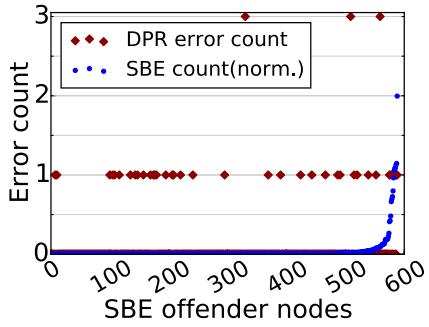






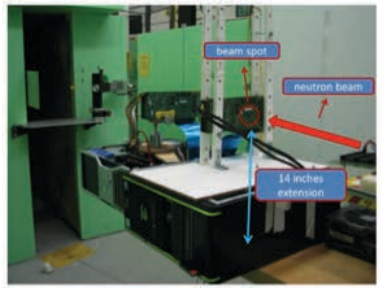


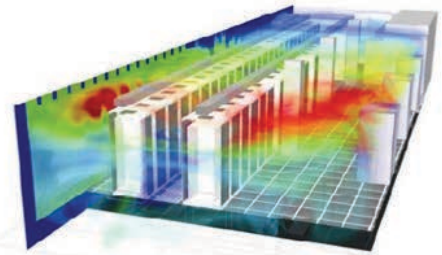


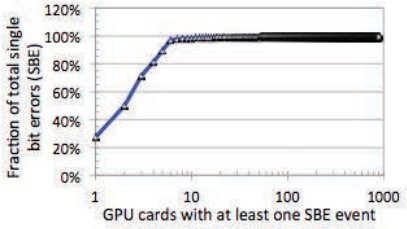


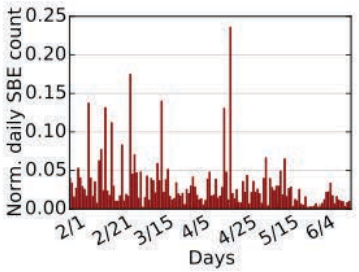
These insights can potentially change the way we design fault-injection modsim frameworks, operate production machines, and plan for future systems.

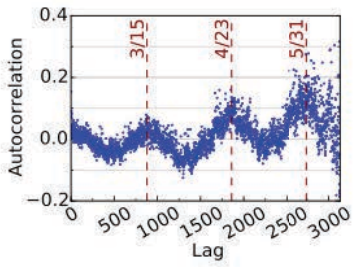
Opportunity for Actionable Analytical Tools

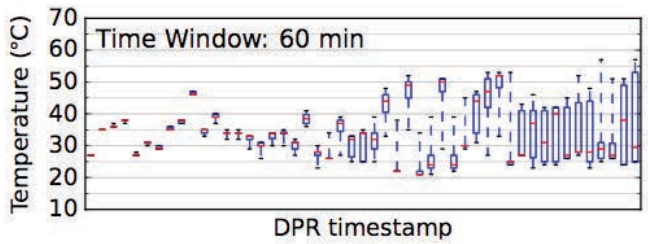


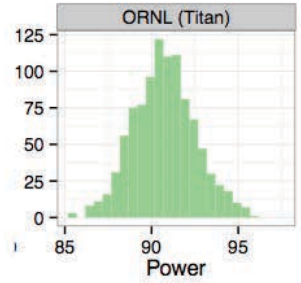












Workload and system generated data

Environment and facilities data

How can we (machine learning) fuse all this data to take meaningful, timely, and profitable decisions at-scale?

My Personal View

Future large scale system will have heterogeneity in terms reliability levels, too.

Parts of large systems will go in transient lower reliability, degraded performance, and large performance variability modes.

Traditional “replace and continue” approach will not be sustainable.

We will need theoretically-sound techniques and tools to “dynamically” manage this new kind of heterogeneity.

Traditional “replace and continue” approach will not be sustainable.

We will need theoretically-sound techniques and tools to “dynamically” manage this new kind of heterogeneity.

Denial and blame shifting will continue to work for some time in near future. ☺

Thanks!

Devesh Tiwari

Oak Ridge National Laboratory

tiwari@ornl.gov

devesh.dtiwari@gmail.com