OAK RIDGE National Laboratory

Smart and Resilient Extreme-Scale Systems

Christian Engelmann, Ph.D.

Senior Scientist & Group Leader
Intelligent Systems and Facilities Group
Advanced Computing Systems Research Section
Computer Science and Mathematics Division
Oak Ridge National Laboratory

ORNL is managed by UT-Battelle, LLC for the US Department of Energy

U.S. DEPARTMENT OF ENERGY

Motivation

- Resilience in extreme-scale supercomputers is an optimization problem between the key design and deployment cost factors:
  - Performance, resilience, and power consumption
- The challenge is to build a reliable system within a given cost budget that achieves the expected performance.
- This requires fully understanding the resilience problem and offering efficient resilience mitigation technologies.
  - What is the fault model of such systems?
  - What is the impact of faults on applications?
  - How can mitigation in hard-/software help and at what cost?

Characterizing Supercomputer Faults, Errors and Failures

Novel Ideas:

- Applies a unified taxonomy for supercomputer faults, errors and failures
- Understanding resilience is a data analytics problem, requiring fusion and analysis of different logs and system health data

Impact:

- Develops an understanding of observed and inferred supercomputer reliability conditions
- Extrapolates this knowledge to future systems
- Enables the systematic improvement of resilience in extreme-scale systems
- Keeps applications running to a correct solution in a timely and efficient manner in spite of frequent faults, errors, and failures

Accomplishments:

- Analyzed 1.2 billion node hours of logs from the Jaguar, Titan, and Eos systems at OLCF
- Developed tools for analyzing logs and creating a fault, error and failure catalog
- Created novel modeling techniques to characterize temporal and spatial failure behavior

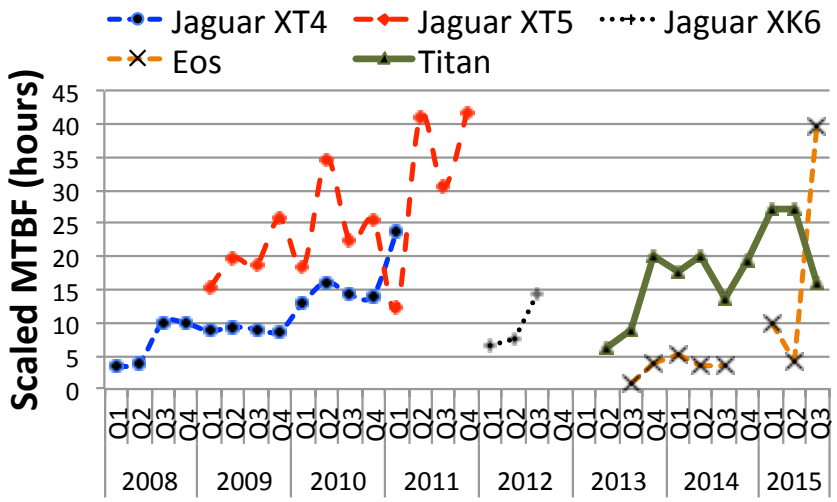
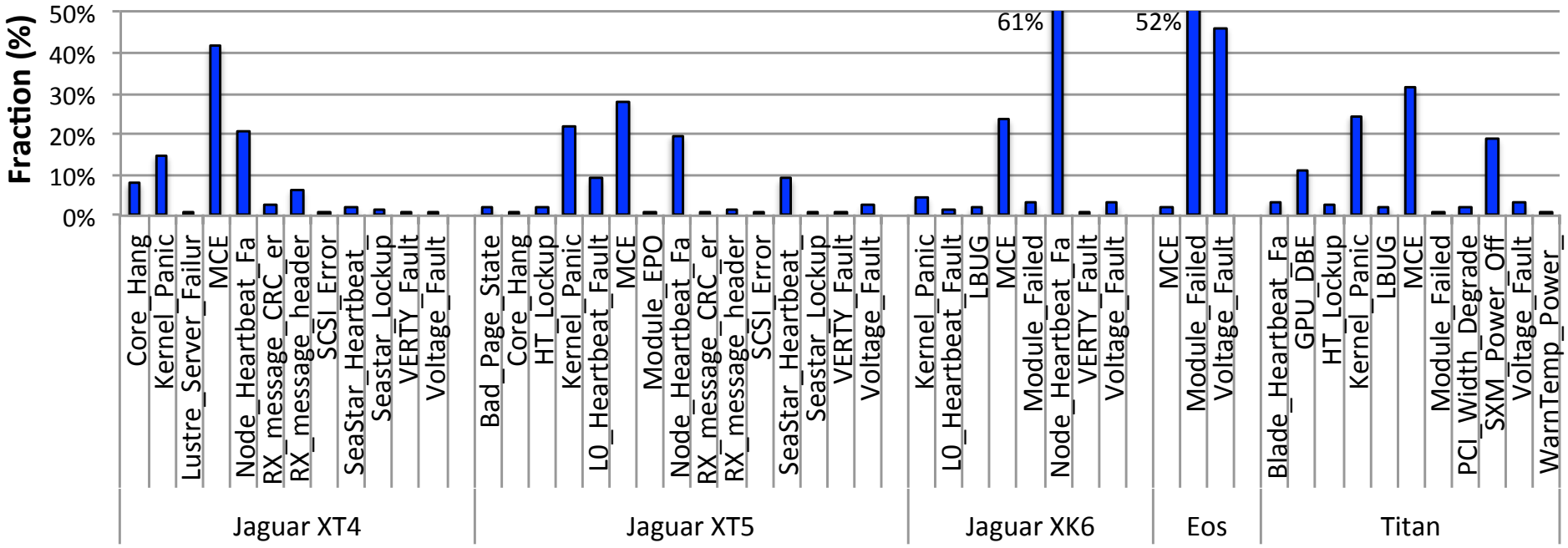


Figure: Each system goes through phases of high and low stability due to continuous efforts of system administrators to improve overall system reliability

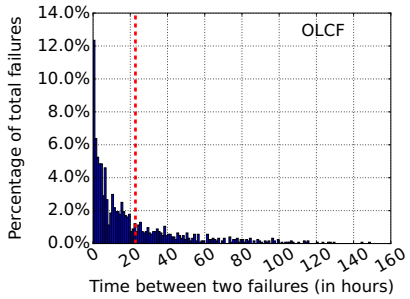
Saurabh Gupta, Devesh Tiwari, Tirthak Patel, and Christian Engelmann. **Reliability of HPC systems: Large-term Measurement, Analysis, and Implications.** SC'17. DOI 10.1145/3126908.3126937.

Characterizing Supercomputer Faults, Errors and Failures

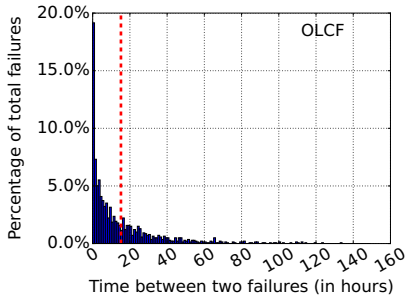


Fraction of each failure type on the studied systems

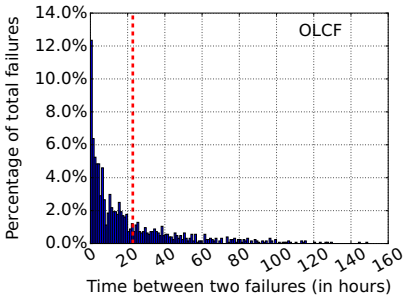
Characterizing Supercomputer Faults, Errors and Failures



(a) Jaguar XT4

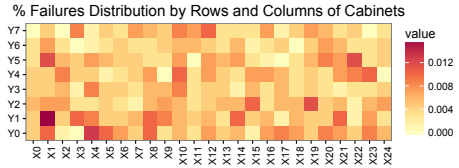


(b) Titan

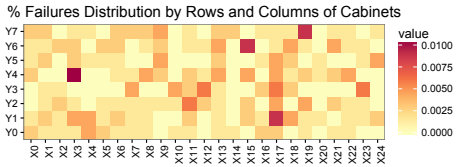


(c) Eos

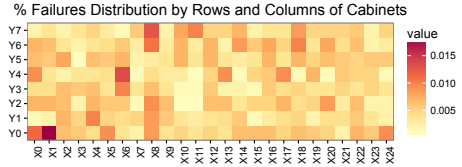
Failure inter-arrival time for 3 studied systems (MTBF as red vertical line)



(a) Jaguar XT5

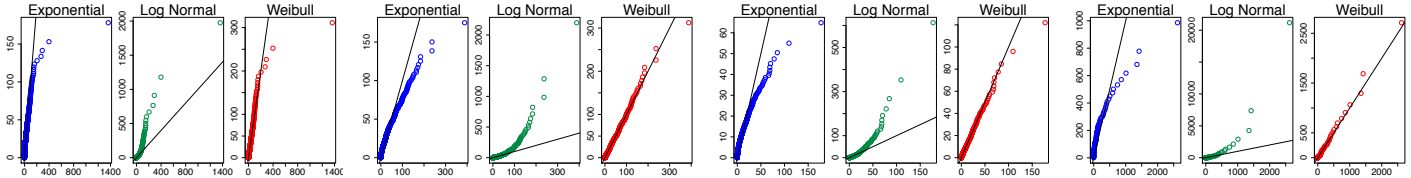


(b) Jaguar XK6



(c) Titan

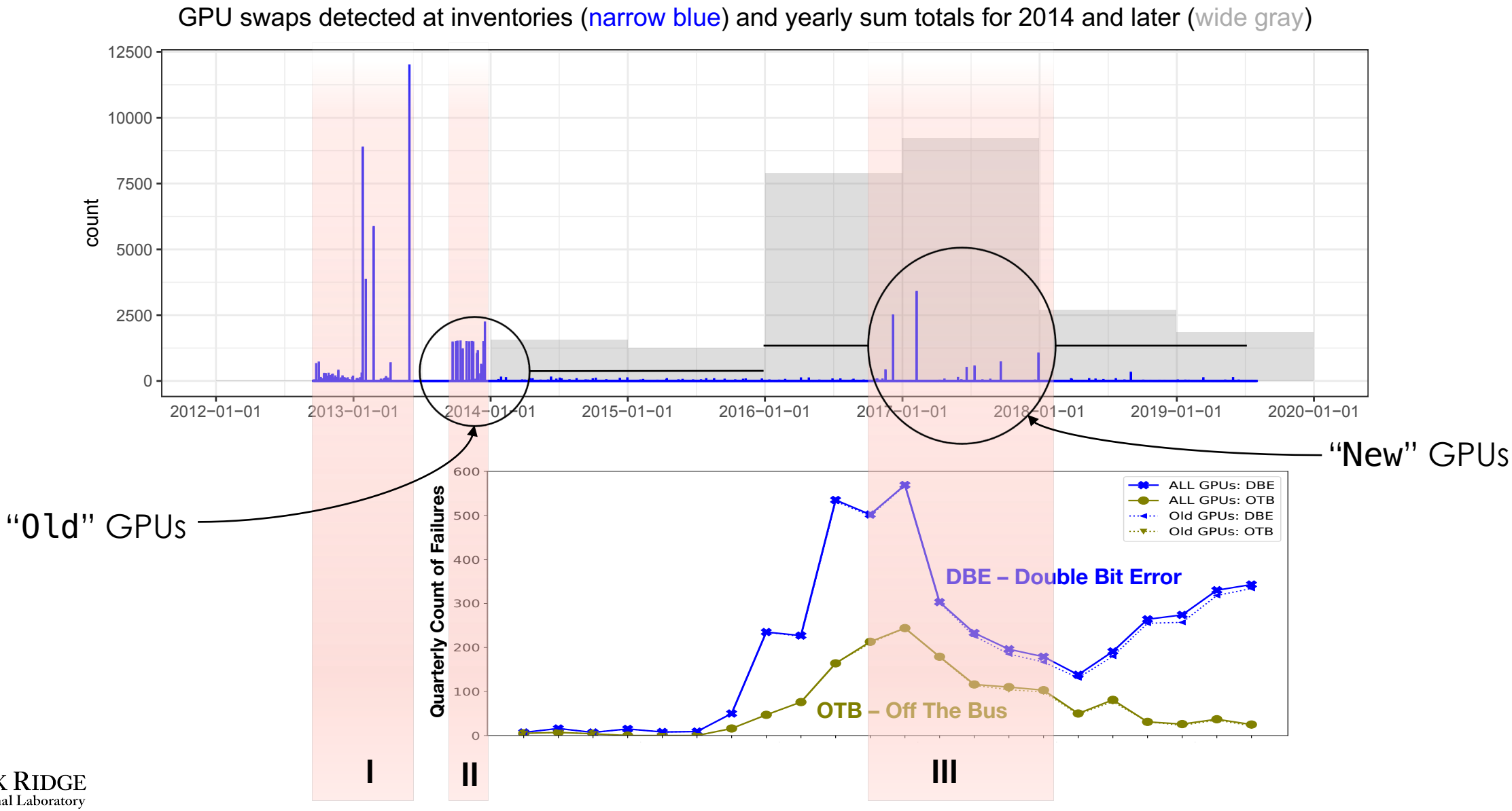
Spatial distribution of failures among cabinets for 3 studied systems



(a) Jaguar XT4 (b) Jaguar XT5 (c) Jaguar XK6 (d) Eos

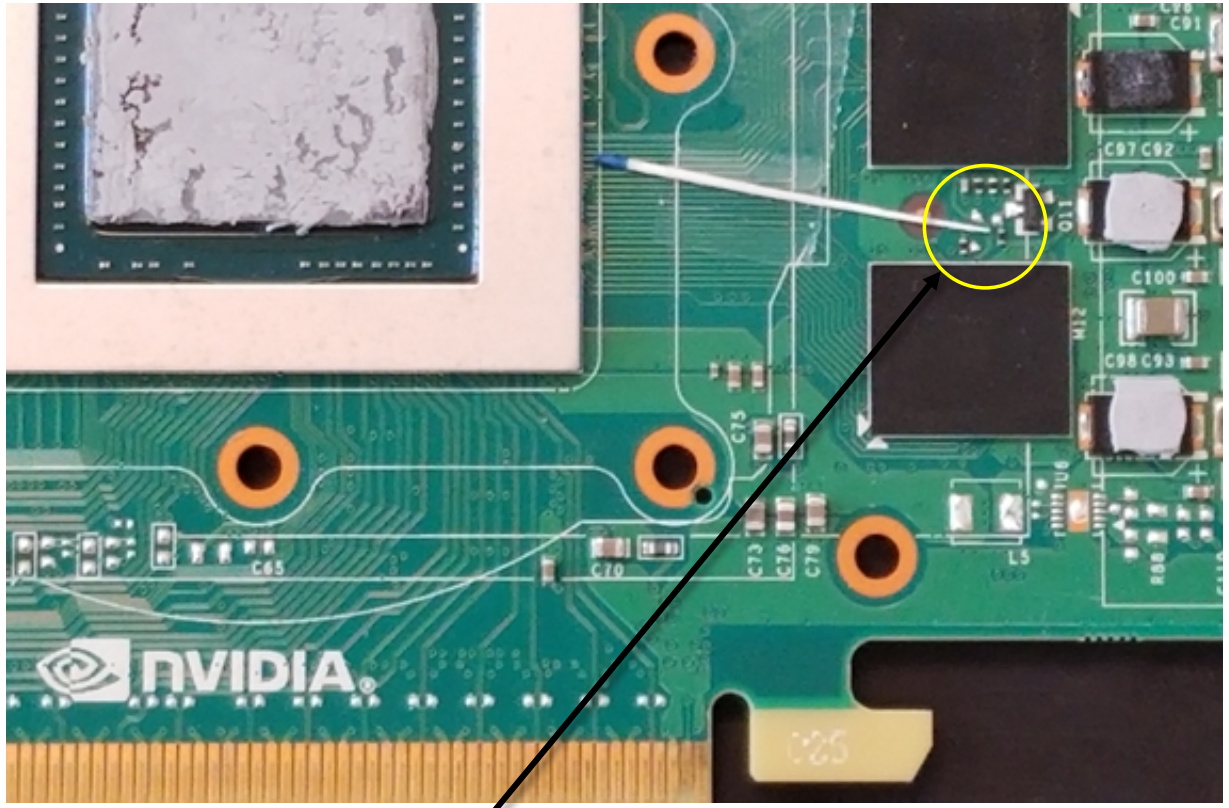
QQ-plots showing goodness of fit for the failure inter-arrival times for 4 studied systems with different failure probability density functions (Weibull fits best)

GPU Failures and Replacements in ORNL's Titan

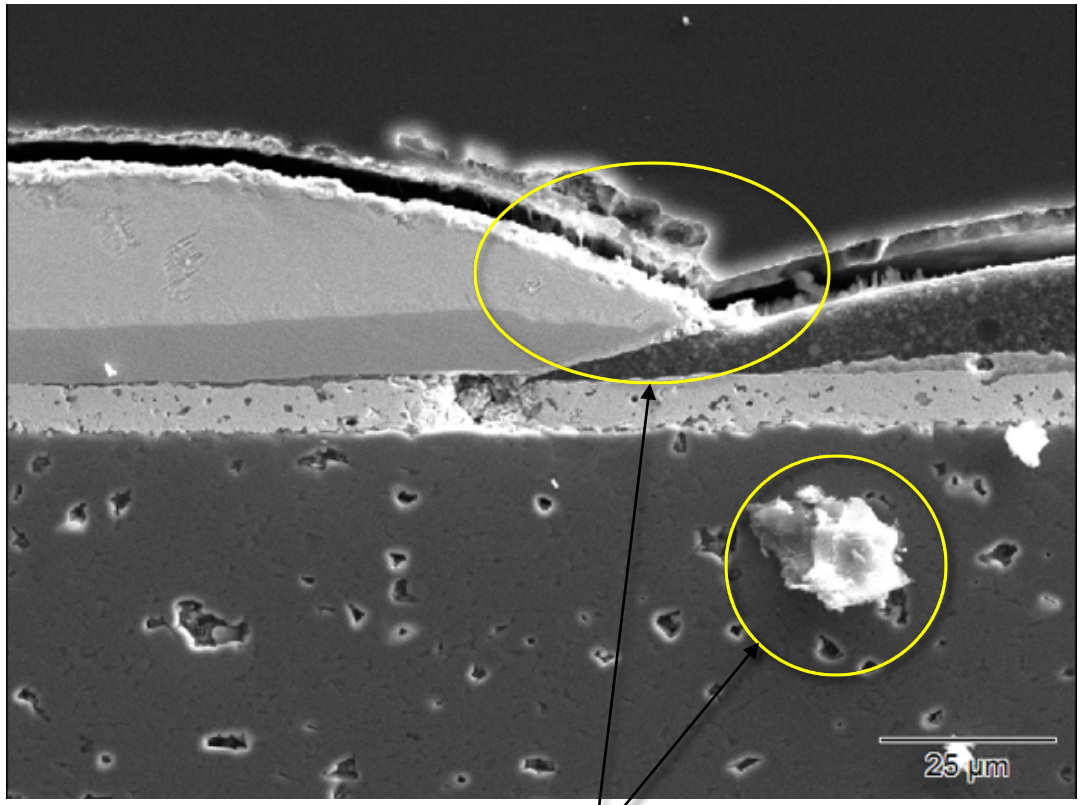


OAK RIDGE National Laboratory

Root Cause: Non-ASR Components on SXM GPU



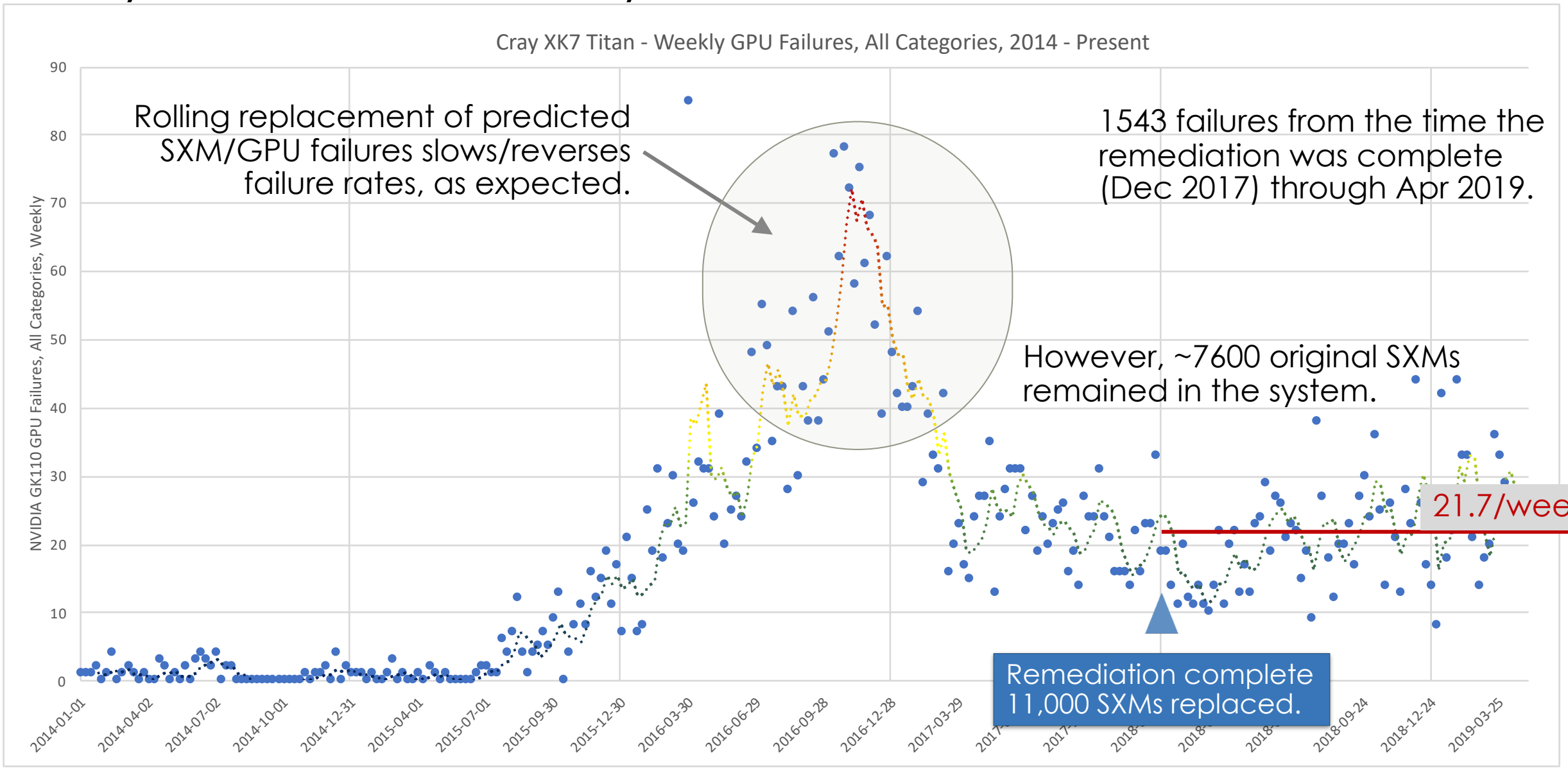
NVIDIA SXM – Location of a non-ASR



Silver-sulfide corrosion
"Flowers-of-Sulfur"

ASR = Anti-Sulfur Resistor

Cray XK7 Titan – Weekly GPU Failures

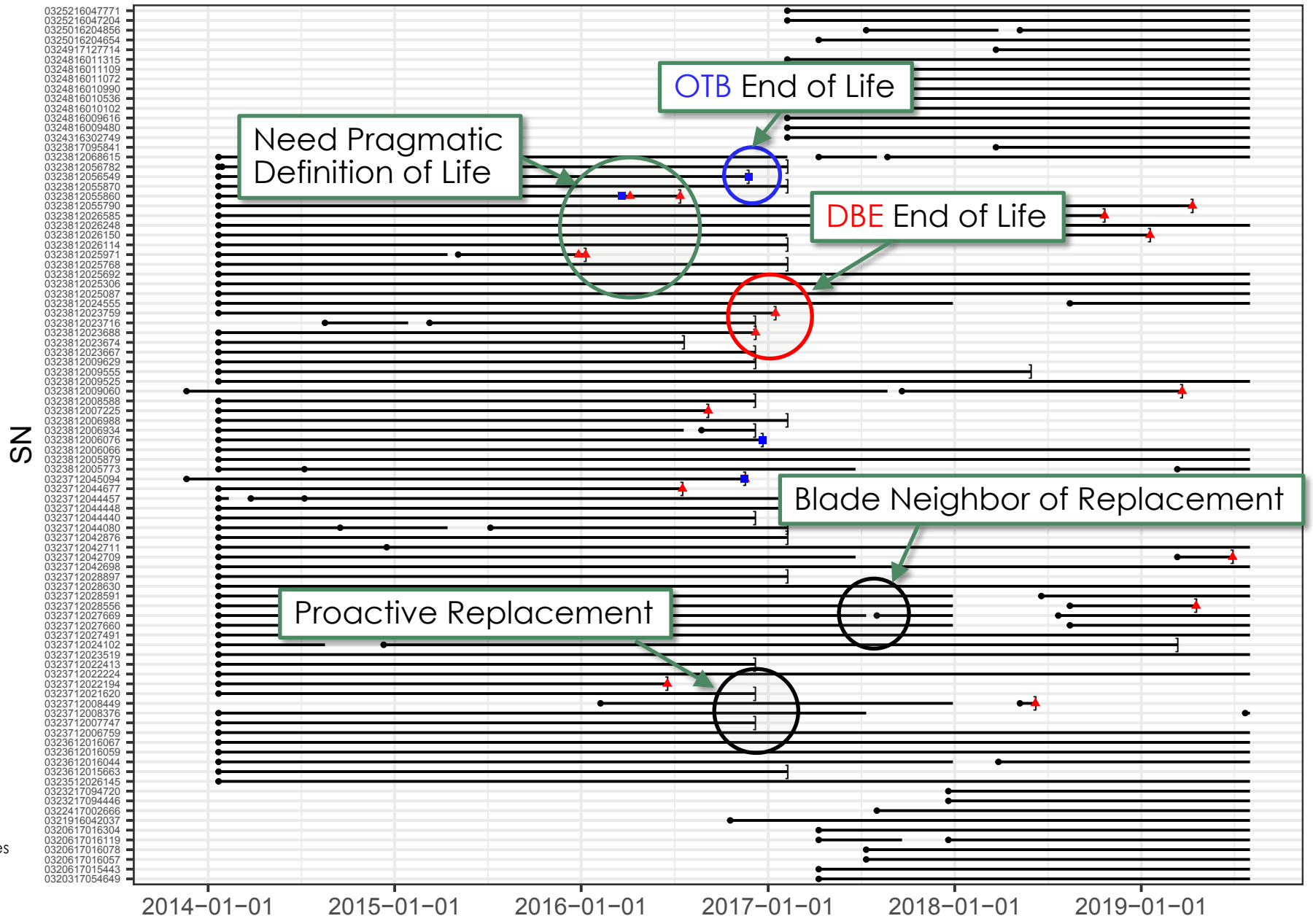


OAK RIDGE National Laboratory

GPU Life Visualization: Serial Number View

Critical for:

- Understanding data
- Defining GPU Life
- Data processing verification

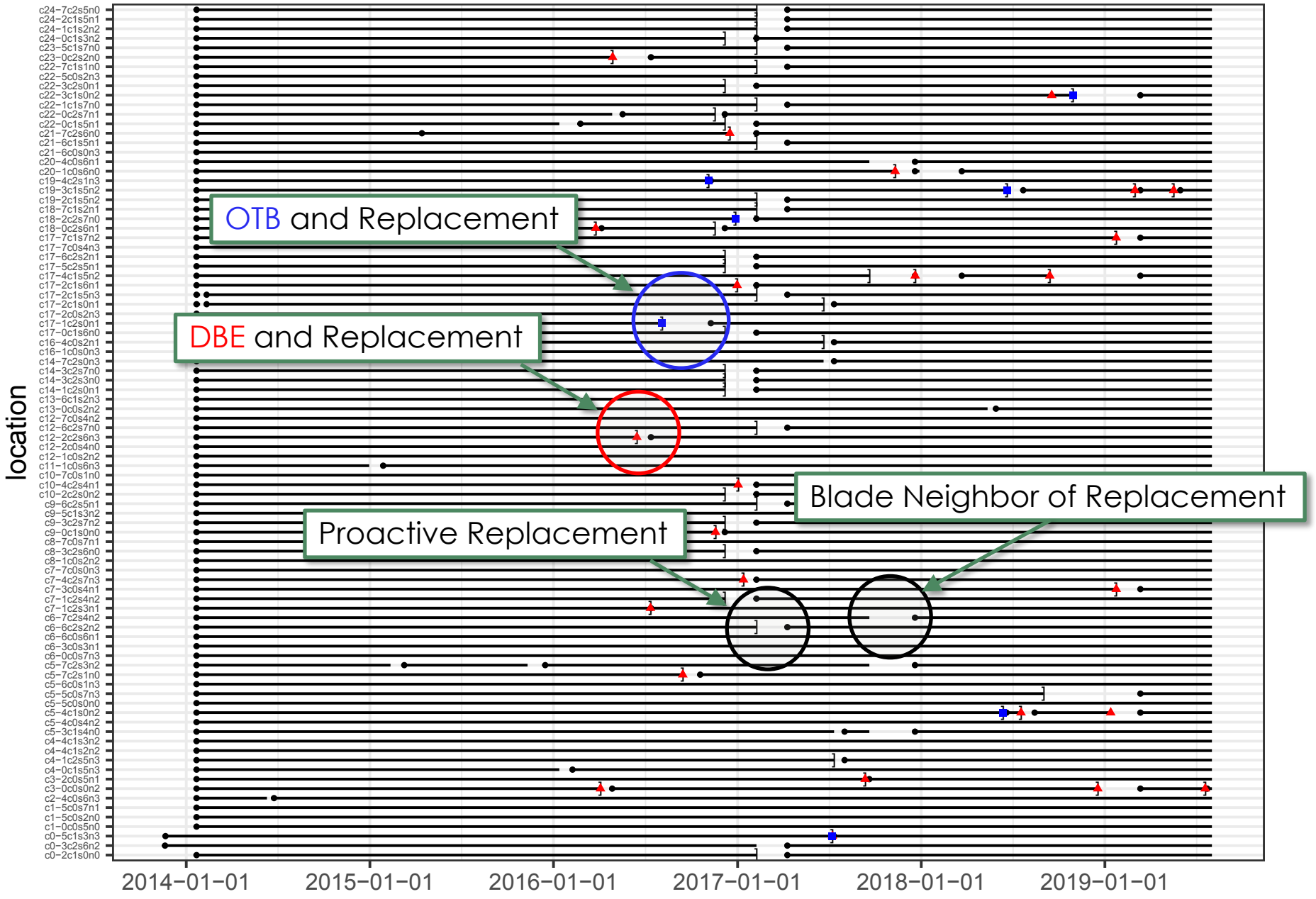


Produced in R via `ggplot2` and `lubridate` packages

GPU Life Visualization: Location View

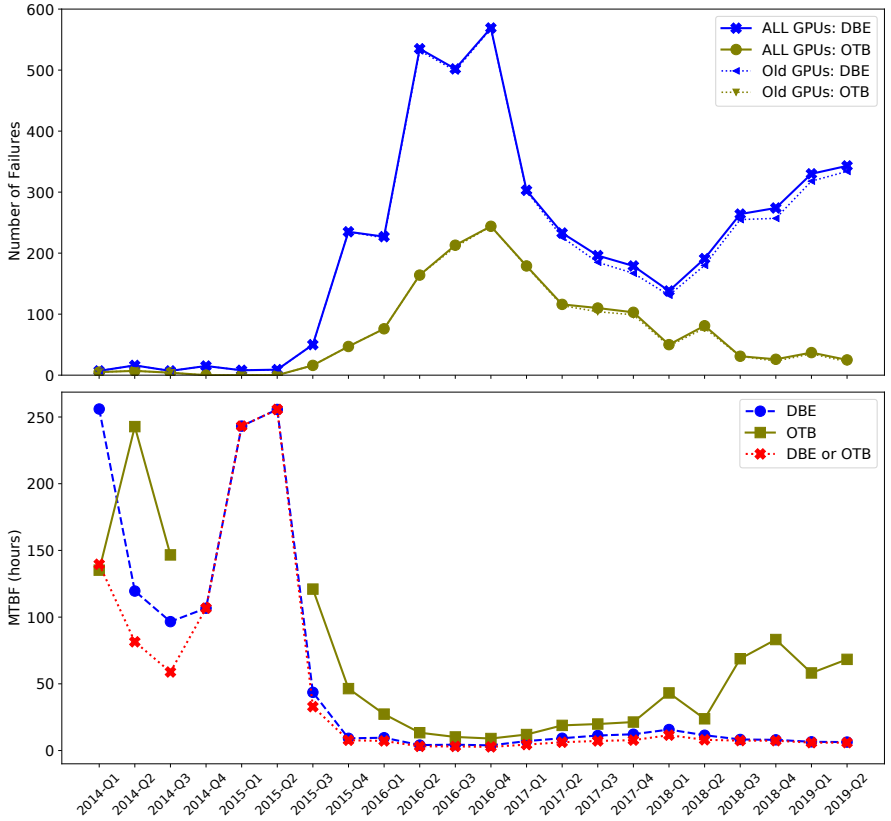
Critical for:

- Understanding data
- Defining GPU Life
- Data processing verification

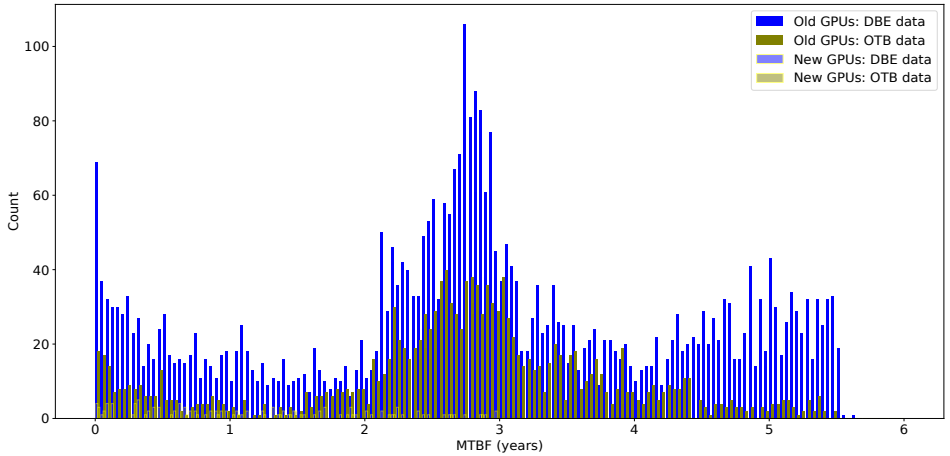


Produced in R via `ggplot2` and `lubridate` packages

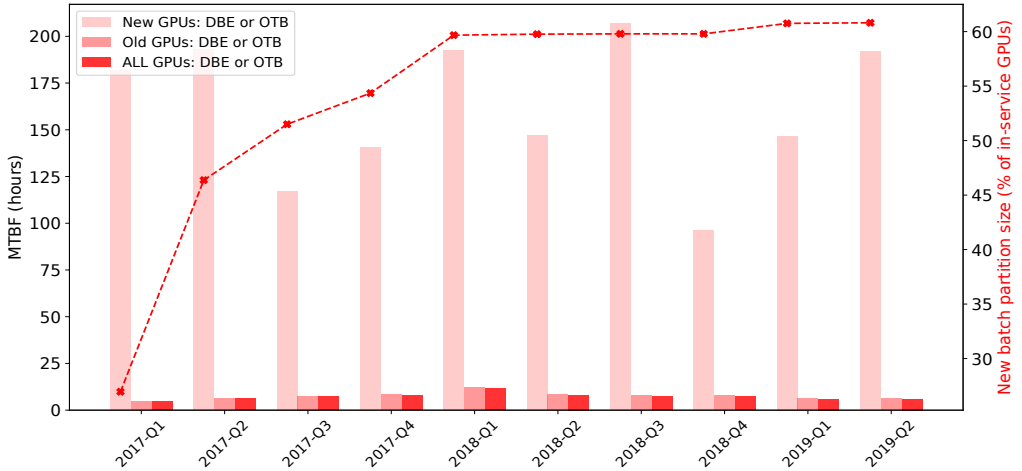
Traditional Reliability in HPC is Focused on MTBF



System-wide Reliability: Quarterly number of failures (top) and MTBF (bottom).



Individual GPU Reliability: MTBF histogram for units that had at least one failure. Interpret carefully: lacks information from units with no failures!

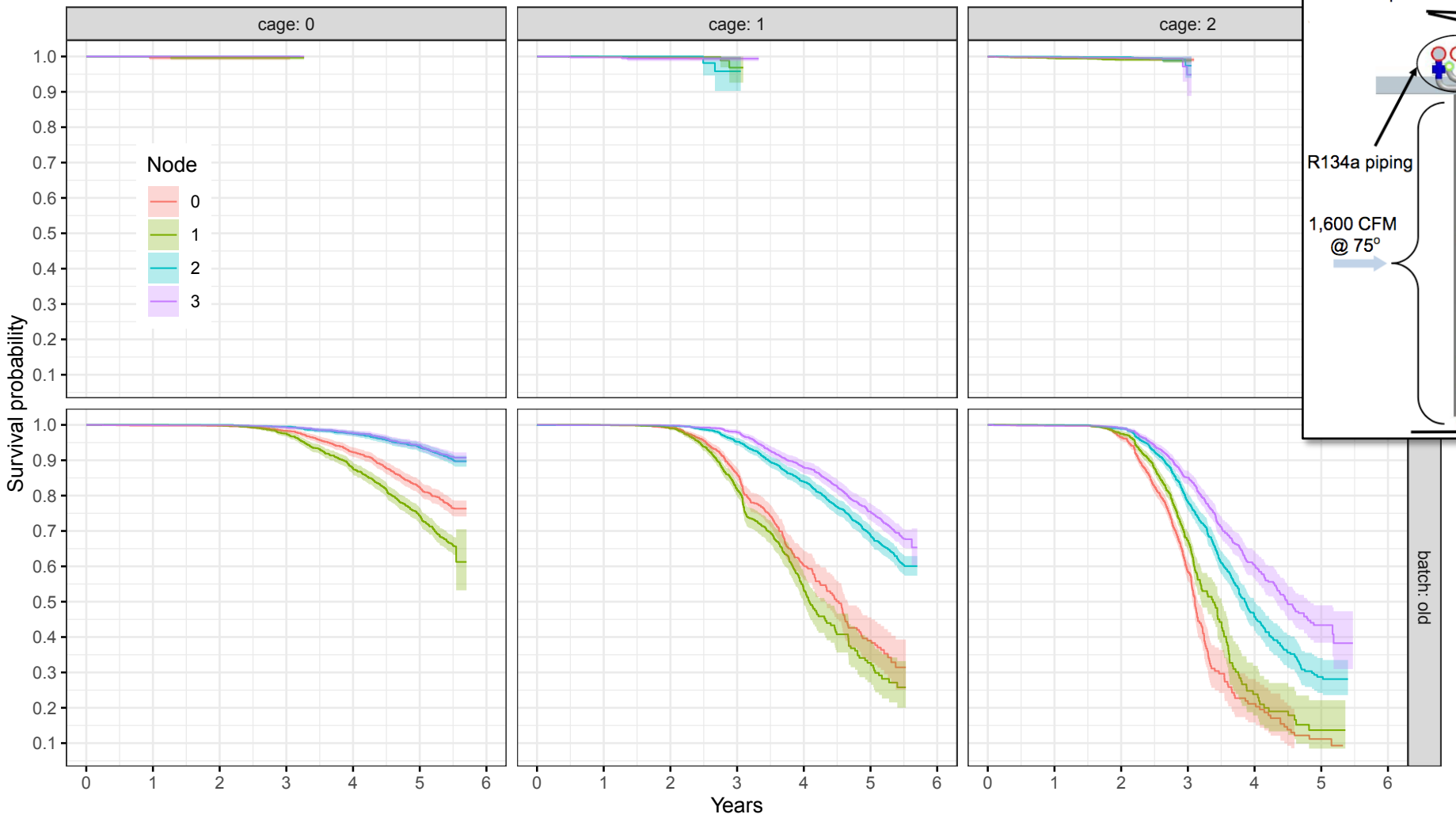


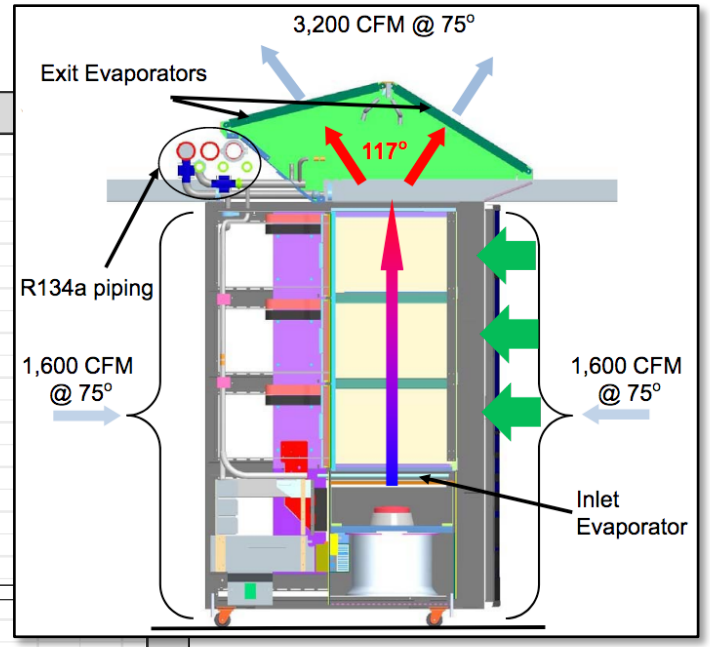
Old–New as Two Partitions: MTBF differs by 12x factor!

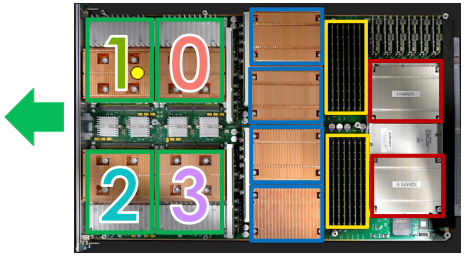
OAK RIDGE National Laboratory

G. Ostrouchov, D. Maxwell, R. Ashraf, C. Engelmann, M. Shankar, and James Rogers. **GPU Lifetimes on Titan Supercomputer: Survival Analysis and Reliability**. 33rd IEEE/ACM International Conference on High Performance Computing, Networking, Storage and Analysis (SC) 2020, pages 41:1-14, Atlanta, GA, USA, November 15-20, 2020. ISBN 9781728199986. DOI 10.1109/SC41405.2020.00045.

Cage and Node Effect Explainable by Airflow in Cabinet

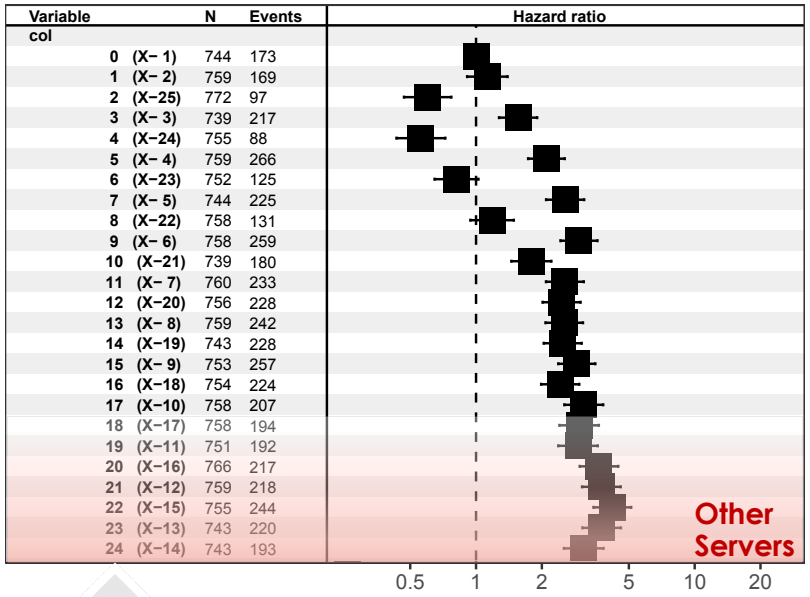




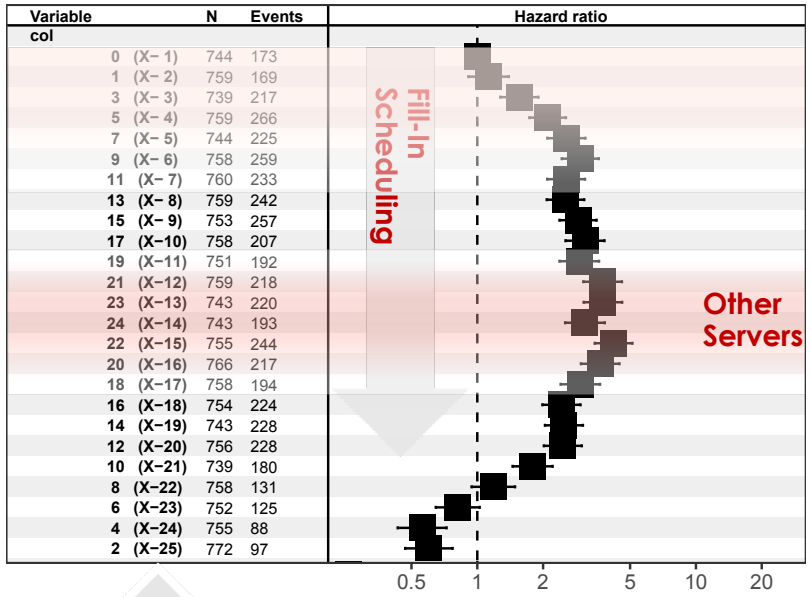


OAK RIDGE National Laboratory

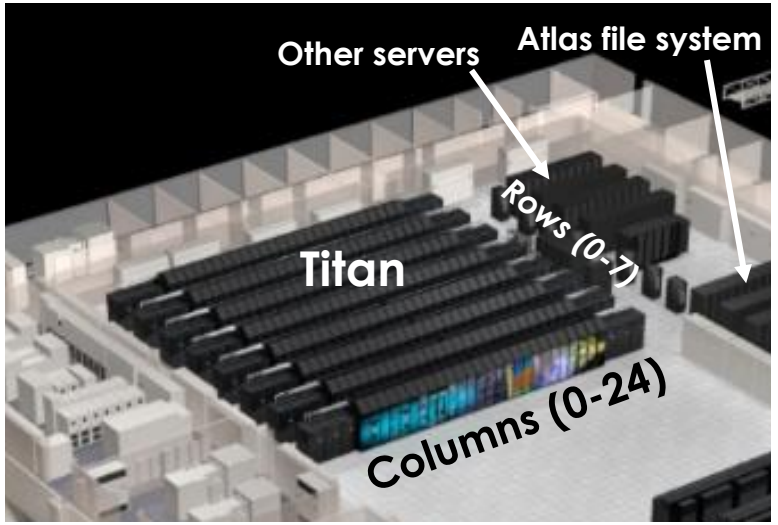
Fill-in Scheduling Effect Explainable via Torus Coordinate



Column Order



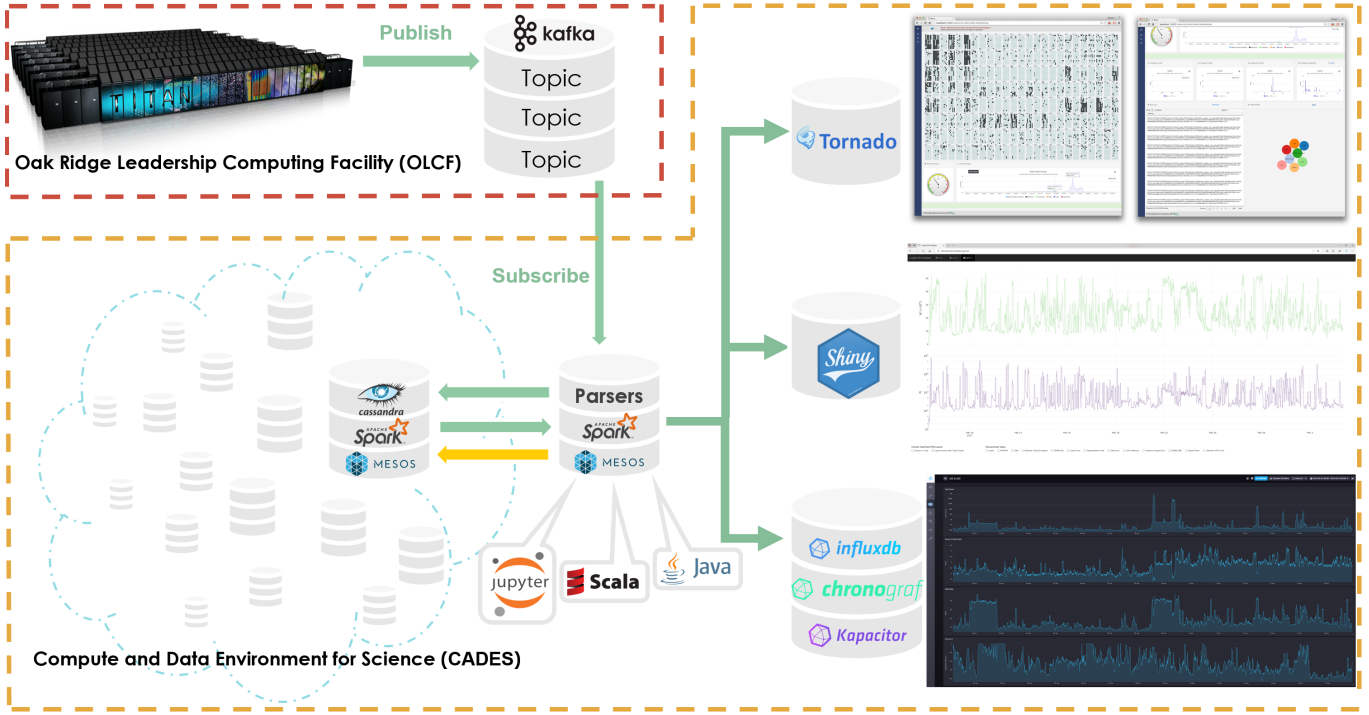
Torus Coordinate Order



OAK RIDGE National Laboratory

LogSCAN: A Big Data Analytics Framework for HPC Log Data

- Improved the analysis of supercomputer reliability, availability and serviceability (RAS) logs using modern Big Data analytics tools to understand resilience issues at scale
- Assessing system status, identifying reliability event patterns and correlating events with application performance improves supercomputer efficiency by identifying error and failure modes
- Created a multi-user Big Data analytics framework – Log processing by Spark and Cassandra-based ANalytics (LogSCAN) – in ORNL's private cloud of Compute and Data Environment for Science (CADES)

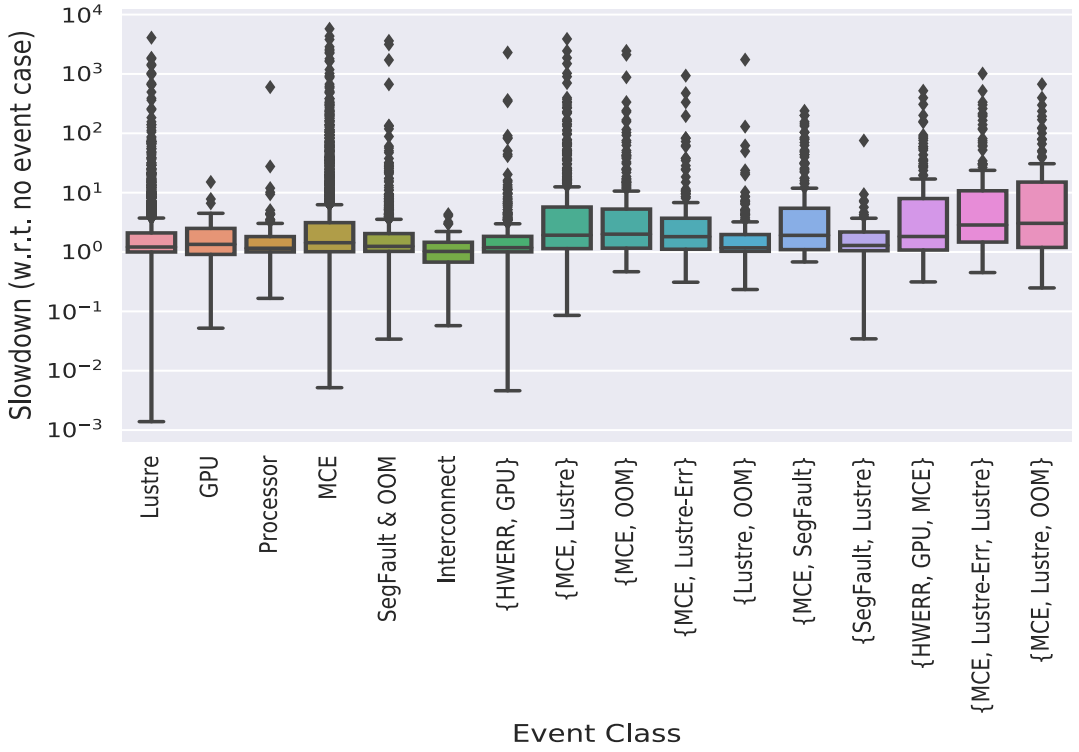


LogSCAN analyzes logs and health data in a combined offline/online fashion

B. H. Park, Y. Hui, S. Boehm, R. A. Ashraf, C. Layton, and C. Engelmann. **A Big Data Analytics Framework for HPC Log Data: Three Case Studies Using the Titan Supercomputer Log**. HPCMASPA'18. DOI 10.1109/CLUSTER.2018.00073.

Analyzing the Impact of System Reliability Events on Applications in the Titan Supercomputer

- Created an understanding of the impact of non-fatal reliability events on scientific application performance.
- Co-analyzed 13 months of application scheduling and reliability event data from ORNL's Titan supercomputer
- Studied the performance characteristics of scientific applications which are most affected by RAS events
- Identified system components that are most likely to impact the performance of scientific applications
- Quantified the slowdown of scientific application jobs due to RAS events from different components

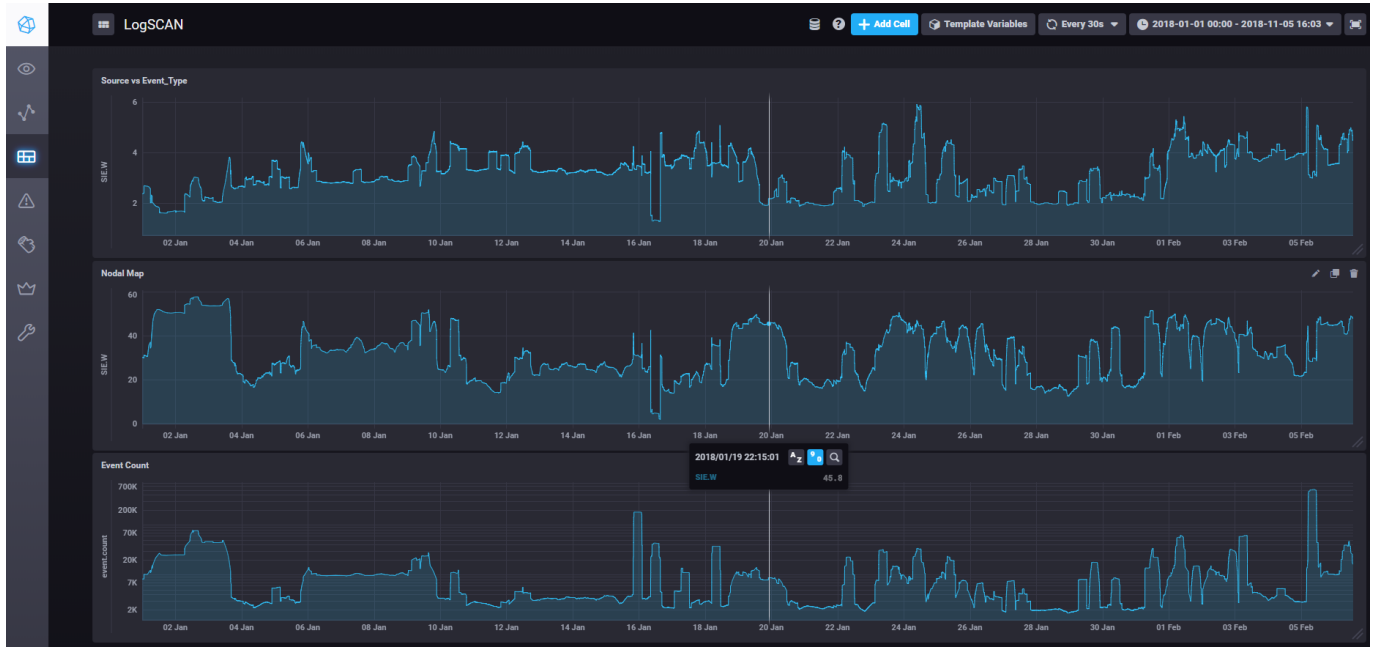


Slowdown assessment of applications executed on ORNL's Titan supercomputer due to reliability issues in various system components:

R. A. Ashraf and C. Engelmann. **Analyzing the Impact of System Reliability Events on Applications in the Titan Supercomputer**. FTXS'18. DOI 10.1109/FTXS.2018.00008

System Information Entropy: A Comprehensive Informative Metric for Analyzing HPC System Status

- Created the System Information Entropy (SIE) metric to concisely represent health status in a time series
- This metric aids operators in assessing system health status by easily and quickly identifying its changes
- Used ORNL's multi-user Big Data analytics framework (LogSCAN)
- Analyzed 3+ years of log data from ORNL's Titan (Jan. 2015 – Mar. 2018)
- Applied Principal Component Analysis and Shannon Entropy Theory to calculate SIEs based on different record vs. feature views of the data



SIE with Source Type layout (top),
SIE with Nodal Map layout (middle), and
Total event count (bottom)

Y. Hui, B. Park, and C. Engelmann. **A Comprehensive Informative Metric for Analyzing HPC System Status using the LogSCAN Platform**. FTXS'18. DOI 10.1109/FTXS.2018.00007.

Evaluating System Health with System Information Entropy (SIE)

3D

General Form of Data Table

	Feature 1	Feature 2	...	...	Feature N
Record 1					
Record 2					
...					
...					
...					
...					
Record M					

2D

Variance Distribution of Principal Components

$$\xi_i = \frac{\sigma_i}{\sum_1^k \sigma_i}$$

1D

Shannon Entropy

$$H = -\sum_1^k \xi_i log_b(\xi_i)$$

Entropy: in a general "b-ary" form

2D

Principal Components in Feature Space

$$\text{SVD} \Longrightarrow \sigma_i$$

σ_i: *i*-th variance out of k eigenvalues of the SVD decomposition

1D

System Information Entropy (SIE)

$$W(t) = b^{H(t)}$$

b: the logarithmic base used in calculating H. In our analysis, b = 10.

System Reliability Event Counts

$$\vec{A} = [a_1\ a_2 \cdots a_M]$$

a_i: total event counts for the application "i"

Application System Impact (ASI)

$$\text{ASI} = \frac{\|\vec{A}\|_{l_2}}{\|\vec{A}\|_{l_1}} = \frac{\sqrt{\sum_1^M a_i^2}}{\sum_1^M a_i}$$

$\| \|_{l_1}$ and $\| \|_{l_2}$ represent the L_1- and L_2-norm applied on $\vec{A}$, respectively.

The value of ASI is limited to the range (0, 1). When ASI approaches 1, it represents high sparsity or a time interval in which only a few applications are generating most of system reliability events and vice versa.

DOE Early Career Award: Resilience Design Patterns

Novel Ideas:

- Design patterns that cover the hardware and software architecture aspects of resilience
- Methods and metrics to holistically evaluate and coordinate fault management
- Reusable programming templates for resilience portability
- Tools for trading off performance, resilience, and power consumption at design and run time

Impact:

- Enables the systematic improvement of resilience in extreme-scale systems
- Keeps applications running to a correct solution in a timely and efficient manner in spite of frequent faults, errors, and failures

Accomplishments:

- Resilience design pattern specification with taxonomy, survey, and pattern anatomy, classification, catalog and language
- GMRES solver with portable multi-resilience against process failures and data corruption
- Performance, reliability and availability models for 15 structural resilience design patterns

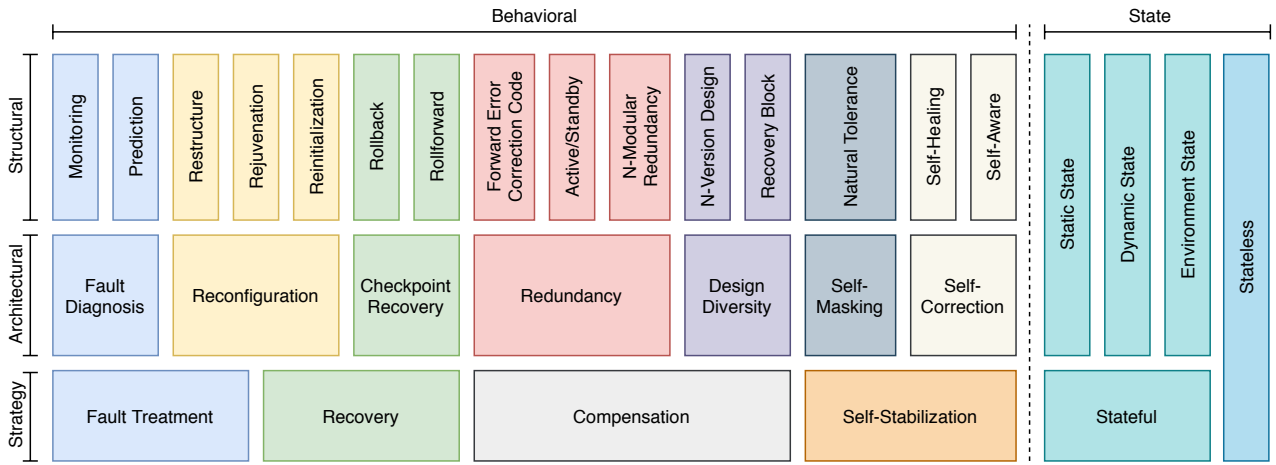
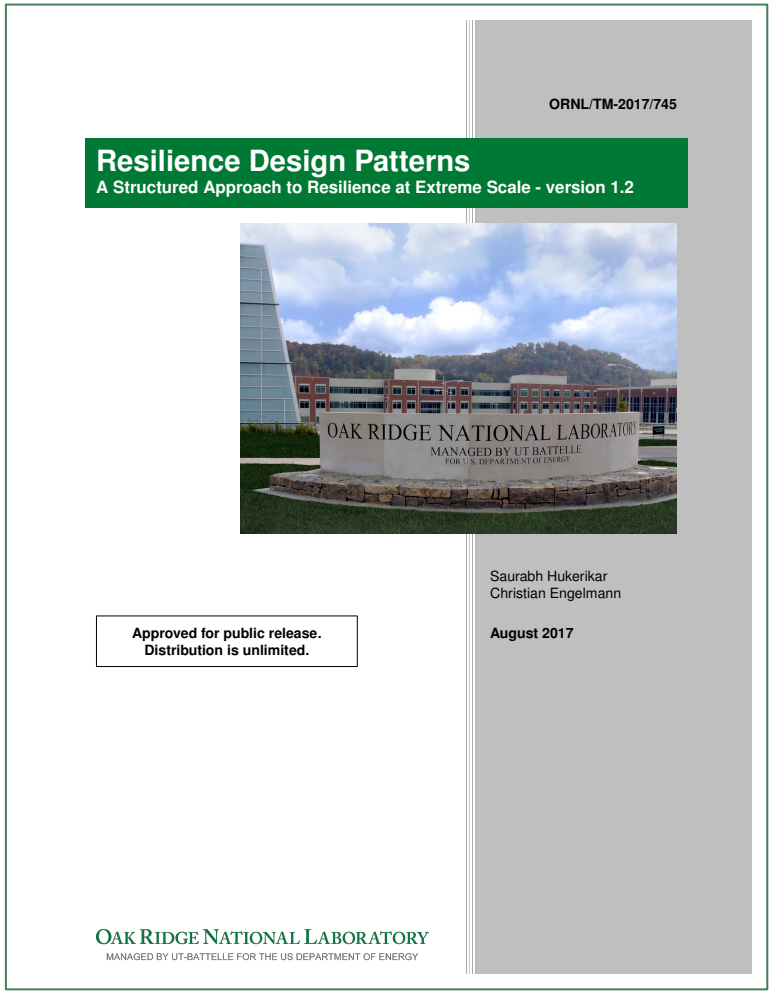


Figure: The 31 identified resilience design patterns

Resilience Design Patterns Specification

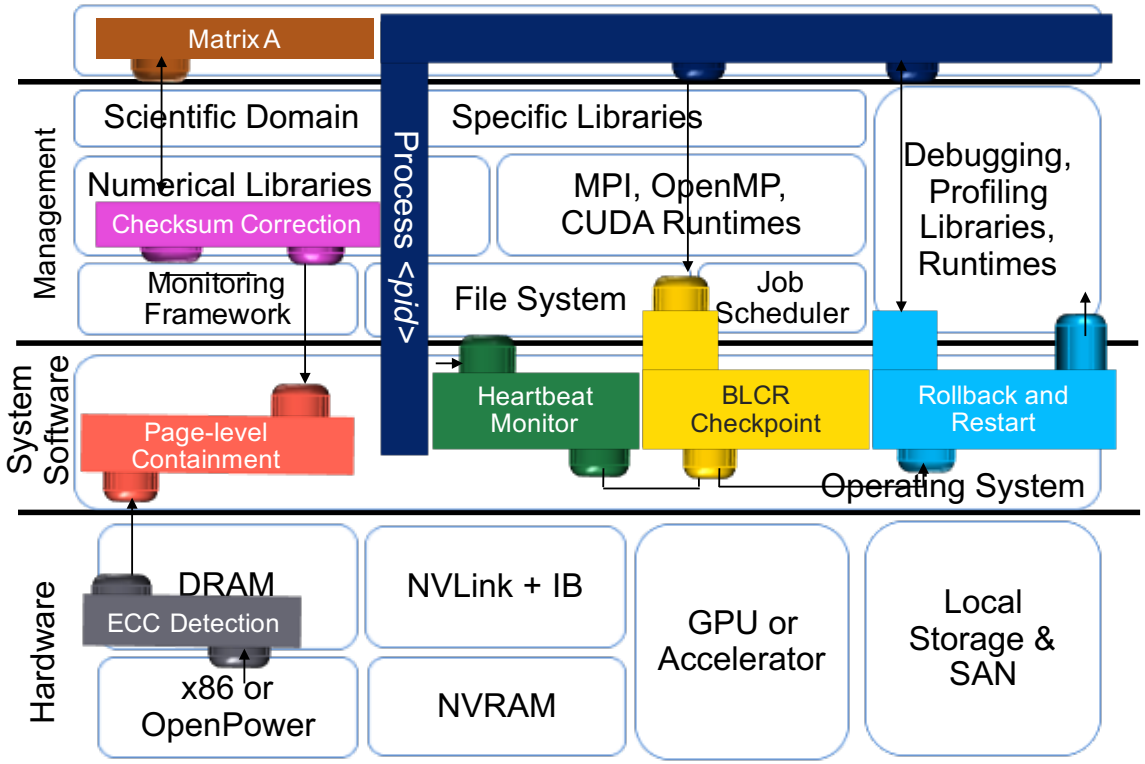
- Taxonomy of resilience terms and metrics
- Survey of resilience techniques
- Classification of resilience design patterns
- Catalog of resilience design patterns
  - Uses a pattern language to describe solutions
  - 4 strategy patterns, 7 architectural patterns, 15 structural patterns, and 5 state patterns
- Case studies using the design patterns
- A resilience design spaces framework
- Version 2.0 to be released by the end of 2020



Saurabh Hukerikar and Christian Engelmann. **Resilience Design Patterns: A Structured Approach to Resilience at Extreme Scale (Version 1.2)**. Technical Report, ORNL/TM-2017/745, Oak Ridge National Laboratory, Oak Ridge, TN, USA, August, 2017. DOI: 10.2172/1436045

Design Space Exploration for Resilience

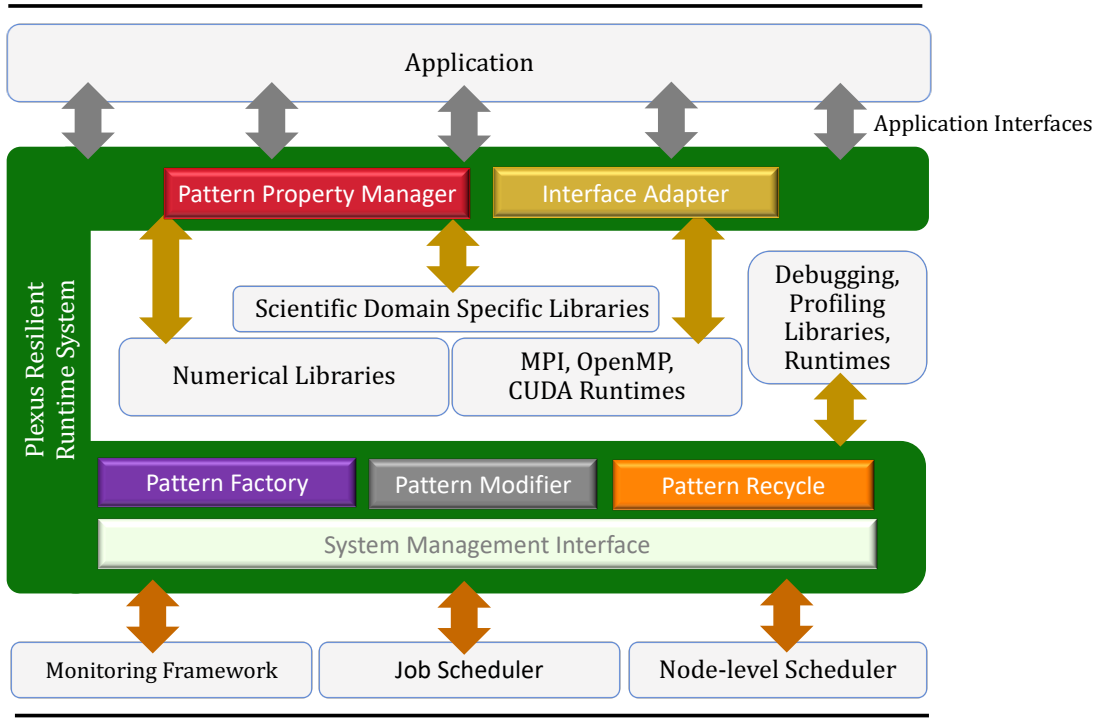
- Vertical and horizontal pattern compositions describe the resilience capabilities of a system
- Pattern coordination leverages beneficial and avoids counterproductive interactions
- Pattern composition optimizes the performance, resilience and power consumption trade-off



PLEXUS: A Pattern-Oriented Runtime System Architecture for Resilient Extreme-Scale High-Performance Computing Systems

- PLEXUS implements pattern instances to provide a resilient environment for HPC applications
- Offers strategies for the resilience patterns to be instantiated, modified and destroyed by the runtime based on policies to meet resiliency needs
- Prototype covers MPI process failures and transient data corruption for a GMRES solver.

S. Hukerikar and C. Engelmann. **PLEXUS: A Pattern-Oriented Runtime System Architecture for Resilient Extreme-Scale High-Performance Computing Systems**. *25th IEEE Pacific Rim International Symposium on Dependable Computing (PRDC) 2020*, Perth, Australia, December 1-4, 2020.



Architecture of the Plexus resilient runtime system, interfacing with programming model runtimes, libraries, system monitoring and job and resource management.

Future Research and Development Needs

- We need to design the HPC hardware/software ecosystem to be able to deal with high error and failure rates, expected and unexpected!
  - Resilience research and development is, in part, risk mitigation against the unexpected
  - There is always a cost/benefit trade-off that needs to be considered
  - Resilience mitigation mechanisms should be a toolbox with lots of options
- Resilience should be by design and not as an afterthought
  - Resilience is a crosscutting issue that should be considered everywhere (and not only in architecture)
    - After 25 years, MPI is still not fault tolerant, while PVM was fault tolerant 28 years ago in 1993

Short-term Future Research and Development Needs

- Portable system/center monitoring and analysis solutions
  - Collecting the right metrics
  - Proper identification of faults (online)
  - Fast and accurate root cause analysis (online and offline)
  - Using advanced statistical techniques and ML
  - ❖ There is some ongoing work at the facilities, but it is disconnected from recent research
- Low-overhead software mitigation techniques (beyond global checkpoint/restart)
  - OS/R and programming model runtime resilience features
  - Resilience for workflows
  - ❖ There is some ongoing work in fault tolerant programming models, but it is underfunded and community adoption is low

Other Future Research and Development Needs (1/2)

- Smart systems and facilities
  - Autonomous resource management that considers the system/facility state and the involved trade-offs
  - Automatic adaptation of systems and facilities in real-time to emerging reliability issues using AI
  - Machine-in-the-loop operational intelligence (OODA loop to improve productivity and lower costs)
- Resilience in federated/distributed/complex computing environments
  - Instruments/laboratories using edge and center computing for science feedback on experiments
  - Real-time and urgent computing that has specific resilience needs
- Understanding the resilience problem in non von Neumann architectures
  - E.g., neuromorphic computing

Other Future Research and Development Needs (2/2)

- Resilience by design
  - Design space exploration that considers resilience in addition to performance and power/energy
  - Performance/energy/resilience co-design
  - Programming for resilience (higher-level abstractions and programming models)
- Resilient algorithms and probabilistic/approximate computing
  - Algorithm-based fault tolerance
  - Coded computing
  - Naturally resilient algorithms
- End-to-end resilience (integrity of data and computation)