Systemic Assessment of Node Failures in HPC Production Platforms

Anwesha Das[1], Frank Mueller[1], and Barry Rountree[2]

[1]North Carolina State University
[2]Lawrence Livermore National Laboratory

Computer Science NC STATE UNIVERSITY

IEEE IPDPS 18th May 2021

Lawrence Livermore National Laboratory

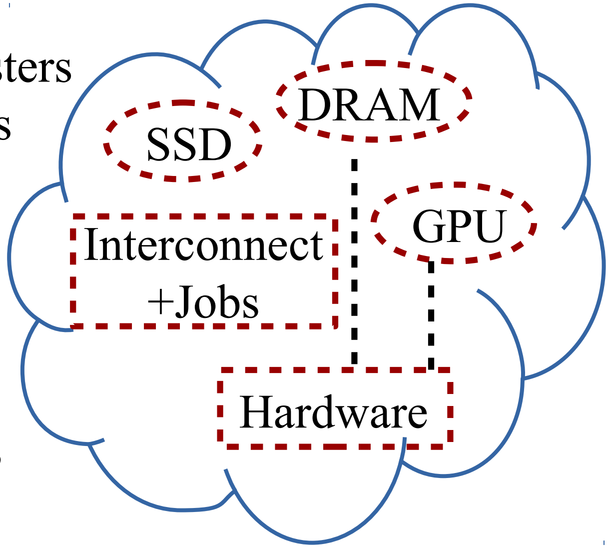
Background

- Existing studies on HPC failure characterization
  - Focus on specific components (e.g., SSD, GPU, DRAM)
  - Consider single or multiple layers and components (e.g., Interconnect & Jobs)
  - Not many studies on holistic analysis (e.g., environmental impact)

- Existing studies on understanding HPC node failures
  - Compute node-specific logs used for analysis
  - Analysis on a single production system or local HPC clusters
  - High-level characterization without sufficient correlations
  - Less work on lead time analysis before nodes fail

- Why another study on node failures?
  - Study environmental impact (e.g., blade/cabinet faults)
  - Integrated approach towards system health
  - Can lead times improve with environmental correlations?

HPC Faults



How nodes fail is not completely understood yet. Study external correlations over space-time to get more insights about what contributes to node failures !!

Background

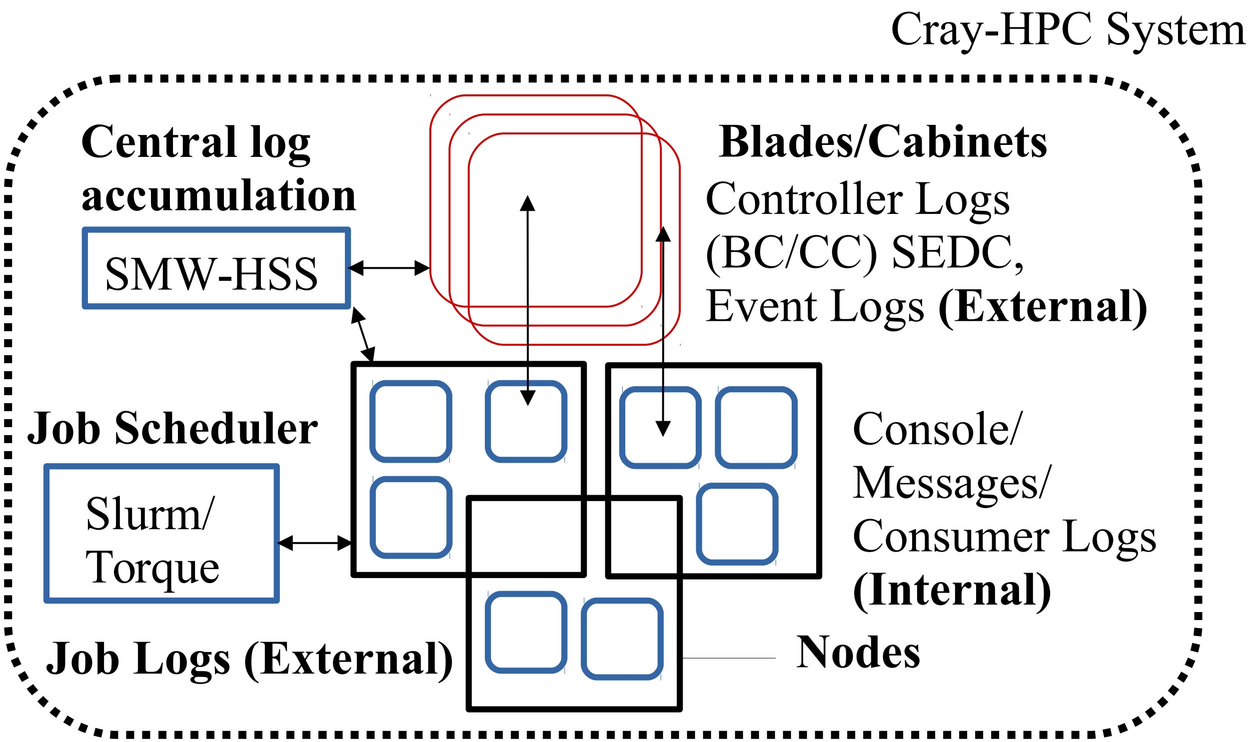
- HPC logs from Cray production platforms
  - Internal: Compute node internal (i.e., console, consumer, messages)
  - External: Job logs (Slurm/Torque), Environmental data (i.e., SEDC; sensor measurements e.g., CPU temp. etc.), Event and Controller logs (additional system events, and blade & cabinet related messages)
  - BC: Blade controller, CC: Cabinet controller, NHF/NVF: Node heartbeat/voltage fault, MCE: Machine Check Exception
  - NHC: Node health checker, mostly threshold-based alarms to alert about node health

- Beyond the scope of this study
  - Resource usage metrics (e.g., memory or network usage, I/O rate)
    - Monitoring tools (e.g., LDMS) were absent in the studied systems
  - Workload details (e.g., user profile, job runtime, queuing delay etc.)
    - Insufficient logs (not our research focus)
  - Manual or electronic reports (e.g., planned outage, recovery details)
    - Not maintained or shared due to privacy policies

Our study: Purely-data driven empirical analysis with internal and external logs; Resource usage data, workload details, and manual reports not considered !!

Background

- System Management Workstation (SMW) + Hardware Supervisory System (HSS) → manage blade & cabinet health; Workload Manager → user job scheduling and submission
- Blade, Cabinet, Node, Job identifiers (IDs) exist → IDs can be leveraged for spatio-temporal correlations across healthy and unhealthy time-frames



This work: Log analysis over 5 HPC systems, 4 of which are Cray production platforms !!

Challenges

- Insufficient data
  - Often production logs have missing or inconsistent data (potential problems with archival or logging daemons)
  - Transient events may not manifest as logs (e.g., cosmic radiations on DRAM)
  - We consider only consistent data

- Harder to assess fail-slow behaviour
  - Fail-stop characteristics are more discernible than gradual degradation
  - Faulty logs may be similar to benign logs of non-faulty times; need to decipher fault propagation through correlations
  - We identify certain faults that are indirectly triggered

- Operator or vendor-level input may not be always available
  - Implication of certain low-level messages can be non-trivial
  - We identify cases that need additional information for decisive causal inference

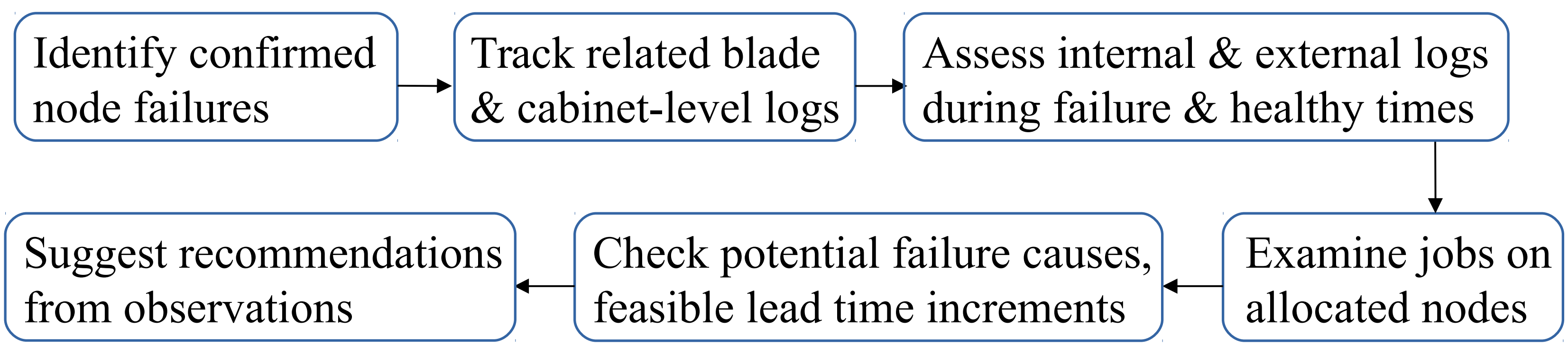
Multiple log components coupled with insufficient data, and transient symptoms inspire failure analysis through external correlations not examined before !!

Methodology

- HPC system log details used in this study: (No GPU-enabled production system)

#Id	Duration	Size	#Nodes	Type	Interconnect	Job Manager
S1	10 mons	373 GB	5600	Cray XC30	Aries Dragonfly	Slurm
S2	12 mons	150 GB	6400	Cray XE6	Gemini Torus	Torque
S3	8 mons	39 GB	2100	Cray XC40	Aries Dragonfly	Slurm
S4	10 mons	22 GB	1872	Cray XC40/30	Aries Dragonfly	Torque
S5	1 mon	3.1 GB	520	Institutional	Infiniband	Slurm

- Overall method:



Methodology

- ❑ Temporal and Spatial Correlations
  - Identified 1200 failed nodes related to unintended faults (prior sysadmin-consulted)
  - Track node ids → blade ids → cabinet ids; job ids on node ids
  - Check timestamps of errors in internal logs (i.e., console logs etc.)
  - Assess external logs during unhealthy + healthy times → early indicators of failures[1]?

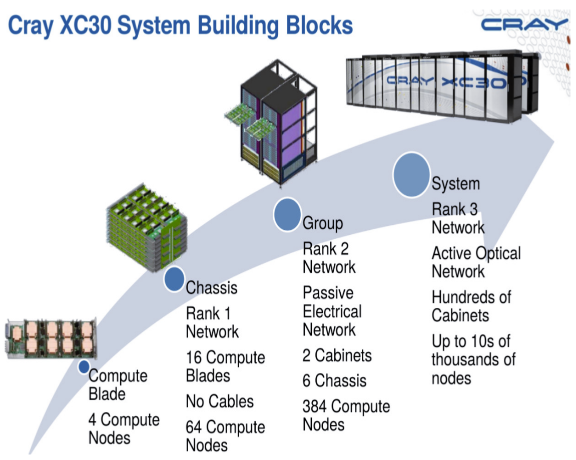
Space

Time

- ❑ Establish correlations with node external and internal logs
  - Are their symptoms of fail-slow characteristics?
  - Are higher lead times possible?
  - Impact of running applications on compute nodes

- ❑ Representative samples in evaluation: Change specific time-frames or duration → does not alter the overall inference

Lower to higher spatial granularity



Study spatio-temporal correlation of single or multiple failed nodes (not system-wide outages) to infer potential cause !!

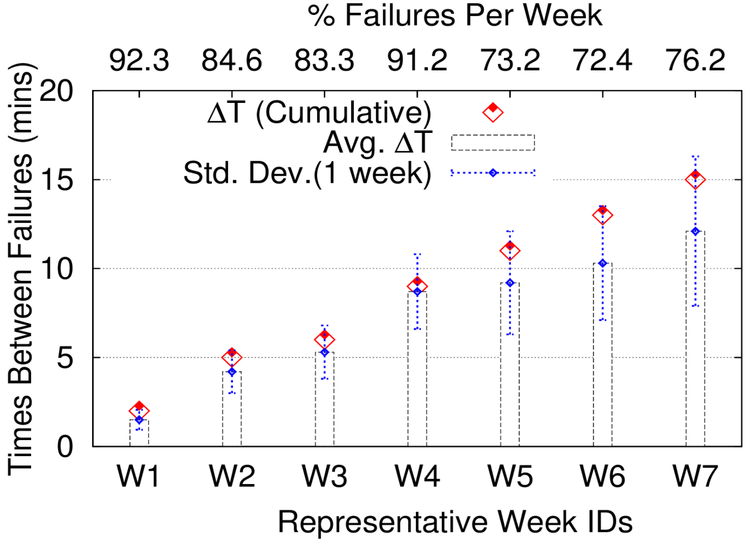
Source[2]

[1] failures indicate node failures unless otherwise mentioned

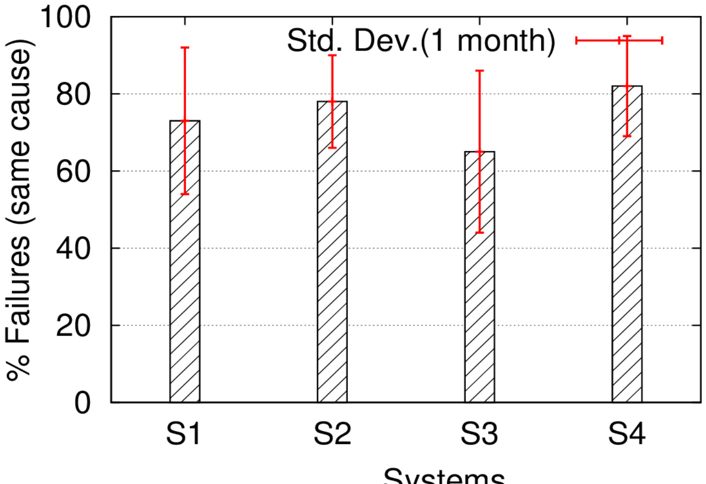
2 https://www.nersc.gov/assets/Uploads/XC30.overview.NERSC.Oct.2013.pdf

Results

- How long are the mean times between failures (MTBFs)? How many nodes share the same dominant failure cause?



Time between failures



Dominant cause

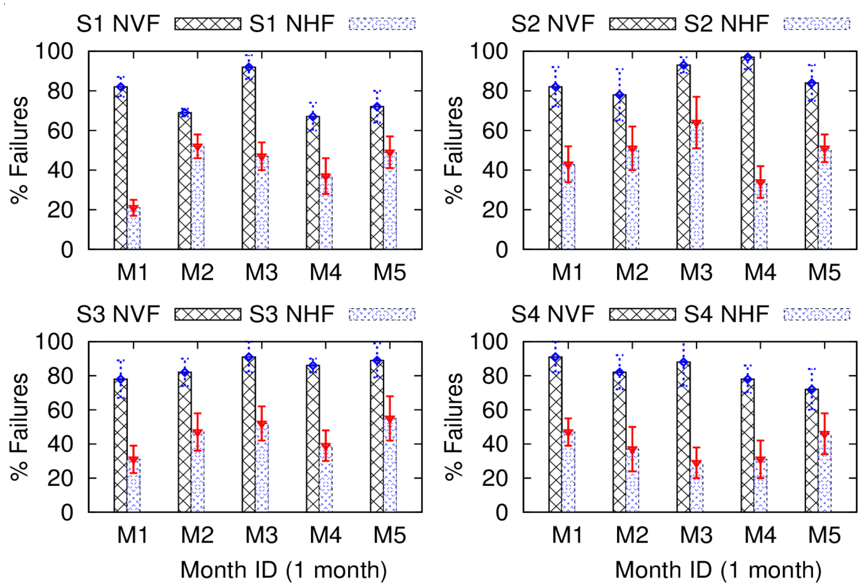
Major Findings:

- 92.3% to 76.2% failures → within 1 & 16 mins of each other; MTBFs → 1.5 to 12.1 mins
- Considerable fraction (65% to 82%) of failed nodes share the same failure cause

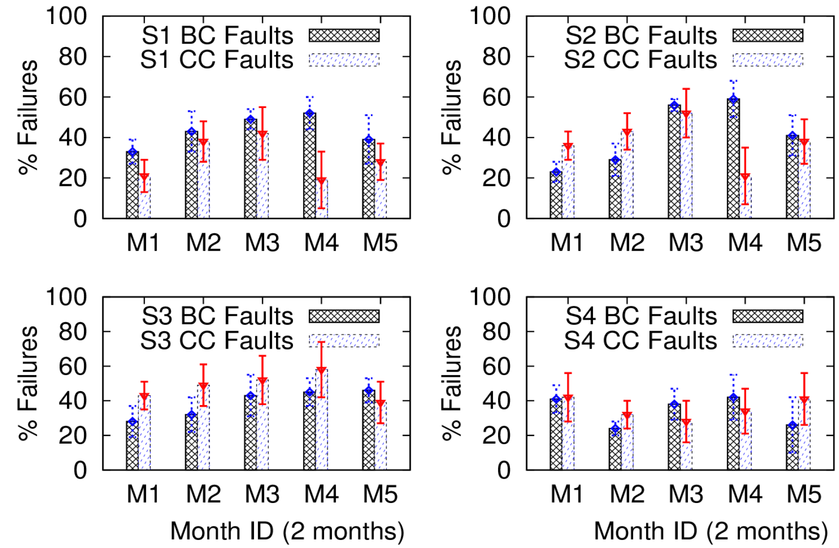
a) Time between failures has reduced (hours → mins) w.r.t. past studies
b) If the dominant cause (e.g., job-based) gets fixed over 50% of the failures can be potentially recovered for certain days

Results

- How much do the blade and cabinet-level messages correspond to failed nodes?



NVF-NHF faults



Blade-Cabinet faults

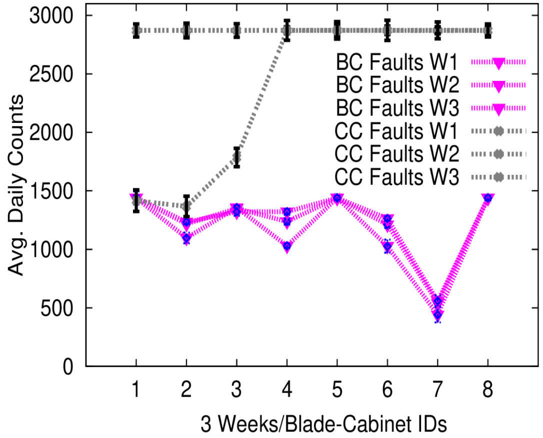
Major Findings:

- 67% to 97% NVFs and 21% to 64% NHFs → manifest as failed nodes
  - Potential early indicators in failure prediction schemes to improve lead times
- Low fraction of failures (19% to 59%) correspond to blade and cabinet faults

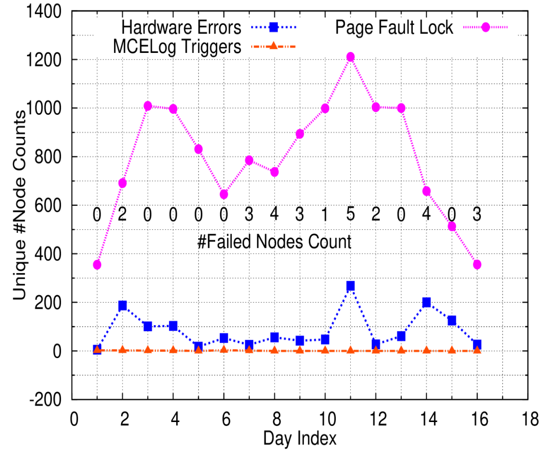
a) Higher correspondence of failures with NHF compared to prior work
b) Blade-cabinet related errors is not the primary cause of failures

Results

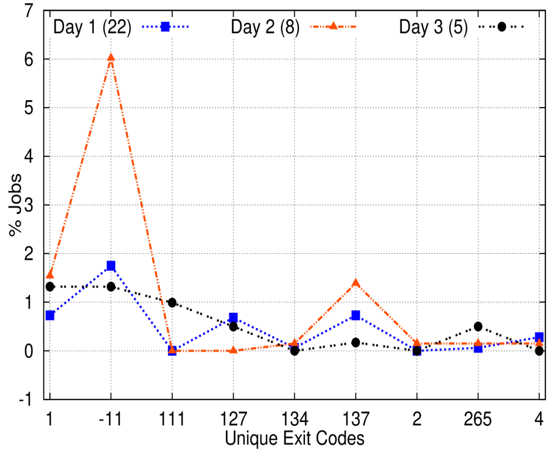
- What faults do not lead to failures?



External health logs



Nodes with various errors



Unsuccessful Jobs

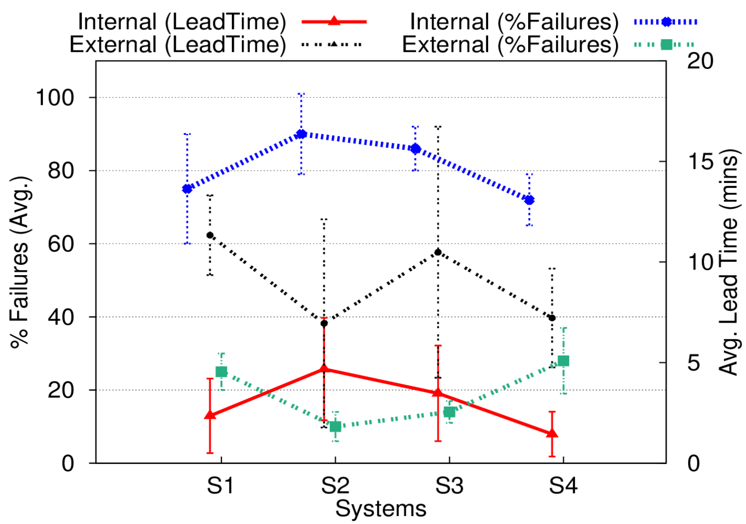
Major Findings:

- Temperature/voltage variations trigger threshold violations → faults and warning messages
  - Warnings appear for healthy blades with no failed nodes as well
- Many nodes encounter errors (Lustre I/O, hardware, MCE log triggers), but few fail
- Failed jobs (non-zero exit codes) may not be due to incapable nodes or application bugs

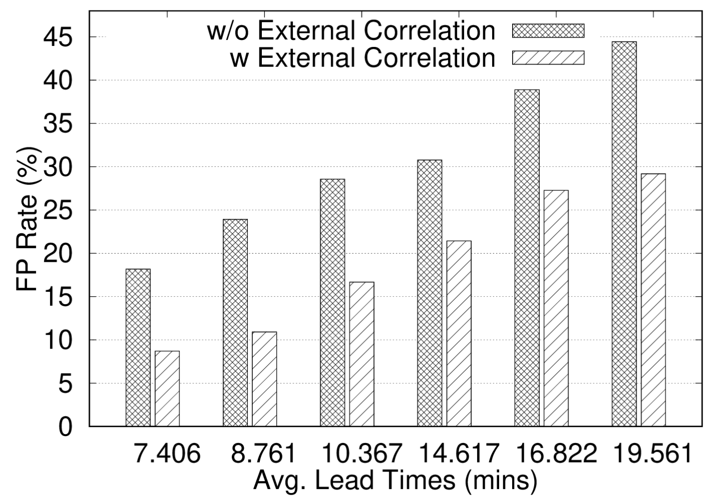
a) Sensor reading violations are not evident cause of failures
b) Increase in error counts need not necessarily degrade system reliability
c) Job errors can relate to unpredictable user behaviour (e.g., memory limit or user killed)

Results

- What is the external influence on node failures?



Lead Time Improvements



False Positive Rate

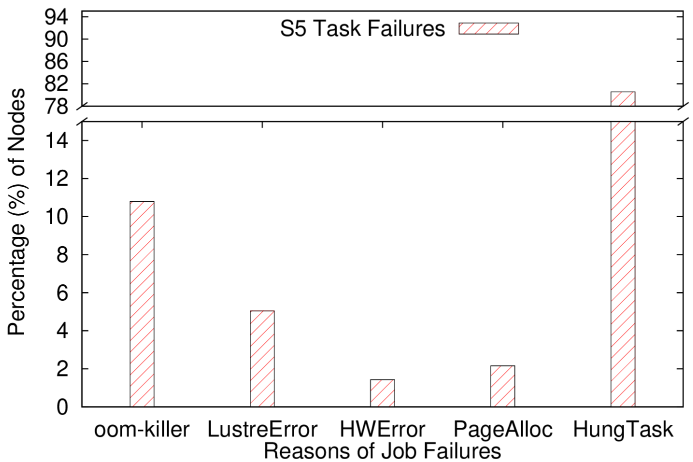
Major Findings:

- Additional environmental correlations besides commonly used node-internal logs
  - Lead time increments not feasible for job triggered failures (early indicators absent)
  - Healthy nodes exhibiting external & internal patterns similar to a faulty node are lower

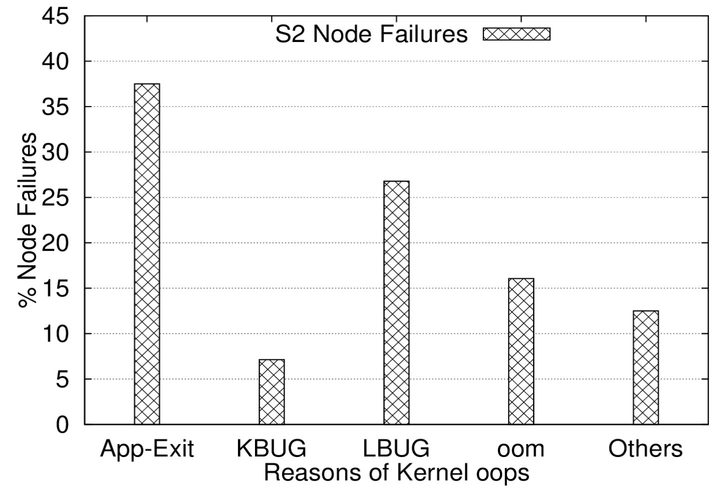
a) Lead time improvements possible (5x) for failures (10% to 28%) with fail-slow patterns
b) Lower false positive rate (30.77% → 21.43%) with external correlations

Results

- How do the applications contribute to node failures?



Institutional Cluster



Production System

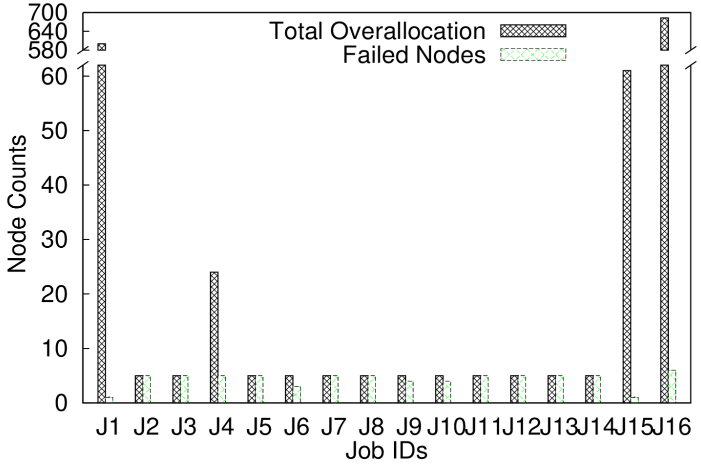
Major Findings:

- Spatially distant nodes sharing the same job → fail during similar times
  - new jobs run → nodes recover
- Institutional cluster: 80.57% of the nodes encounter hung task timeout errors (slow I/O system unable to flush the data)
- Cray systems: File system bugs common, bugs seeming to emanate from the OS (e.g., kernel bugs) can be application-triggered (e.g., resource crunch)

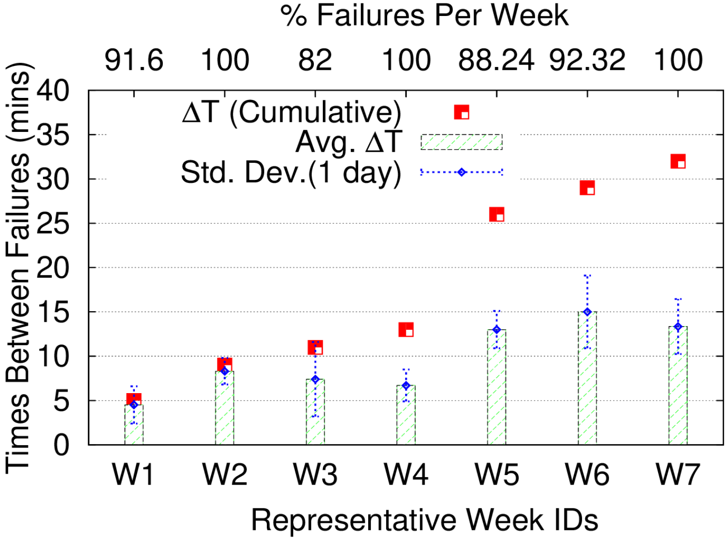
a) Quarantining nodes ineffective for buggy jobs, b) Applications influence Lustre contention and resiliency, c) Several job errors in non-Cray cluster do not fail nodes

Results

- Can memory over-allocation lead to node failures? Do job-triggered failures exhibit temporal locality?



Overallocation



Temporal locality

Major Findings:

- When a fraction of over-allocated nodes fail, jobs fail to complete
- Dominant kernel modules in stack traces → hints at job-triggered versus file system bugs
- Temporal locality of nodes exists for job-triggered failures; Week1: 91.6% failures in ≤ 5 mins

a) Better resource-aware job scheduling needed (e.g., job submission parameters)
b) Stack trace analysis can give cues about cause (e.g., ldml_bl: job, dvs_ipc: OS)
c) Performance-awareness during job scheduling can improve system health

Results

- Unknown cases:
  - *Type:2; Severity:80; Class:3; Subclass:D; Operation: 2* → BIOS problems; no other additional fault evidence; appear during healthy times as well
  - *L0_sysd_mce* → hardware problems related to blade controllers; insufficient data to infer the true failure cause
  - Some shutdown messages exist → no prior anomaly symptoms
    - Potential operator errors by accident

- Characterization through empirical correlations
  - No automated framework with a generic algorithm (not our objective)

- Environmental faults are not primary culprits; not surprising?
  - In certain data centers or GPU-based HPC clusters → temperature variability impacts failures; no such evidence in our study

- Are findings applicable to systems with extreme heterogeneity?
  - Performance-aware diagnosis + stack trace analysis can benefit newer HPC systems as well (e.g., augmented accelerators, deep learning on multi-core processors)

Indecisive cases of failures exist; Findings applicable to systems with extreme heterogeneity; Further actions subject to future developmental efforts !!

Summary

- Higher error count need not lead to a failure; certain faults/blade failures → degrading health
  - Consider non-critical health faults before launching checkpoint/restarts (C/R)
- Major blade-cabinet related faults do not correlate with failures
  - Ignore frequent SEDC/controller warnings unless indicators exist in the internal logs
- Fail-slow symptoms exist (e.g., *ec_hardware_errors*) → potential to enable feasible lead time enhancements in failure prediction schemes
- Application misbehaviour → quarantining may not be effective; inform users about buggy jobs instead of sequestering nodes
- Kernel oops often observed: Machine learning guided stack trace analysis → narrow down the root cause (e.g., job vs. file-system caused)
- Often the origin of the failure lies in the application: Performance-aware workload scheduling + system failure prediction → improve system health

Account for non-critical messages before launching C/R; Resource-awareness w.r.t. overall system performance is important in reducing failures !!

Conclusion

- Major environmental warnings do not strongly correlate with anomalous node shutdowns
- Lead time enhancements feasible only for certain failures, not caused by the application (5x more time with lesser FP rate)
- Better resource-aware scheduling and call trace study unveiling potential insights about failures, can further improve system reliability
- Failures apparently caused by hardware or file system can be triggered by application misbehavior (i.e., node quarantining may not be effective)

Findings from our study can facilitate further research from the community to propose solutions that can aid in taking suitable failure mitigation actions for system health !!

Sample Data Release: https://doi.org/10.5281/zenodo.4114171

Acknowledgments: Lawrence Livermore National Lab under contract DE-AC52-07NA27344, Lawrence Berkeley National Lab, and NSF grants 1525609 & 0958311 for funding & support

Thank you

Questions?