# *Root Cause Analysis of Node Failures in Production HPC*

**Anwesha Das**

**29th November 2018**

# Today's Talk

- Online Log Parsing and Disk Error Prediction
  - Drain – ICWS'17
  - CDEF (Cloud Disk Error Forecasting) – Usenix ATC'18

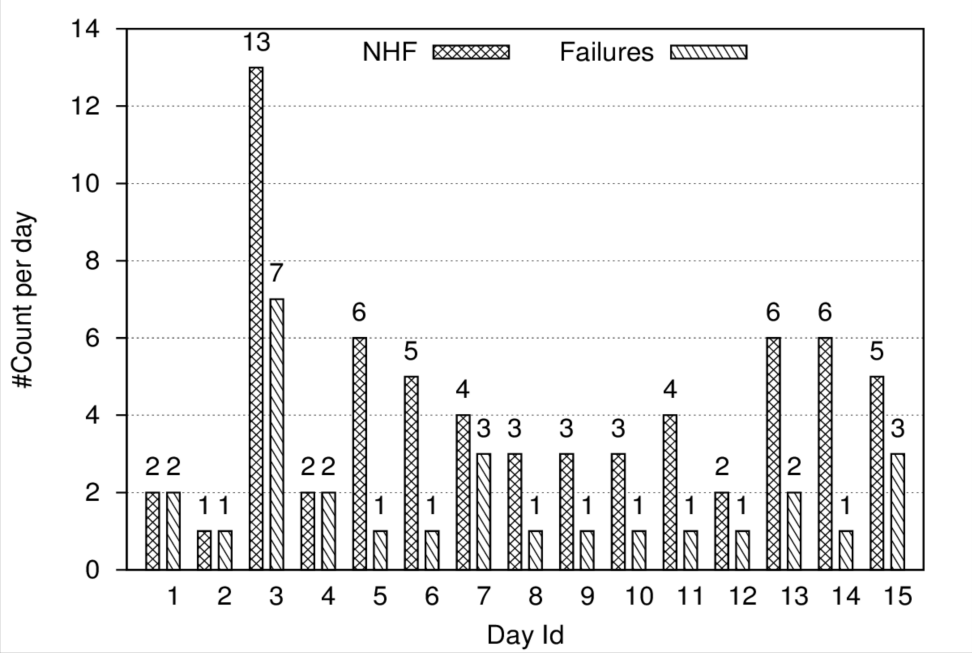
- My work: Node Failures on HPC platform (Cray Supercomputers)
  - RCA – Root Cause Analysis of Compute Node Failures

# Node Failure Analysis

- Environment Consideration
  - SEDC warnings
  - Cabinet Faults
  - Heartbeat Faults
  - L0SYSD_MCE
- Kernel oops
  - Breakdown of diverse reasons
- Application Triggered
    - Job correlations
    - No other external indications

# Heartbeat Faults

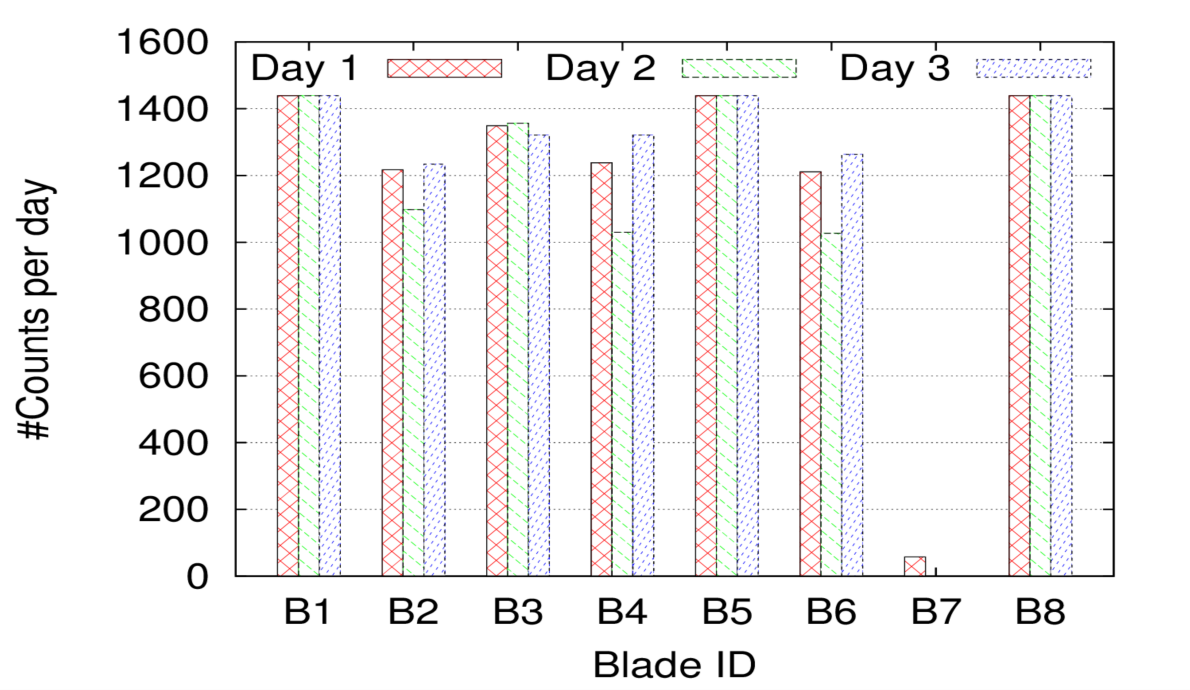
Do all node heartbeat faults eventually result in failure?




- Many NHF (Node Heartbeat Faults) do not eventually manifest as failed nodes
  - ***Might be dead*** case, Missed heartbeat or failed health test
- In 2 weeks, 43.07% NHFs actually caused failed nodes
- On a different system, similar symptoms {(1,1), (3,1), (1,1)} (NHF, Failures)

# SEDC Warnings

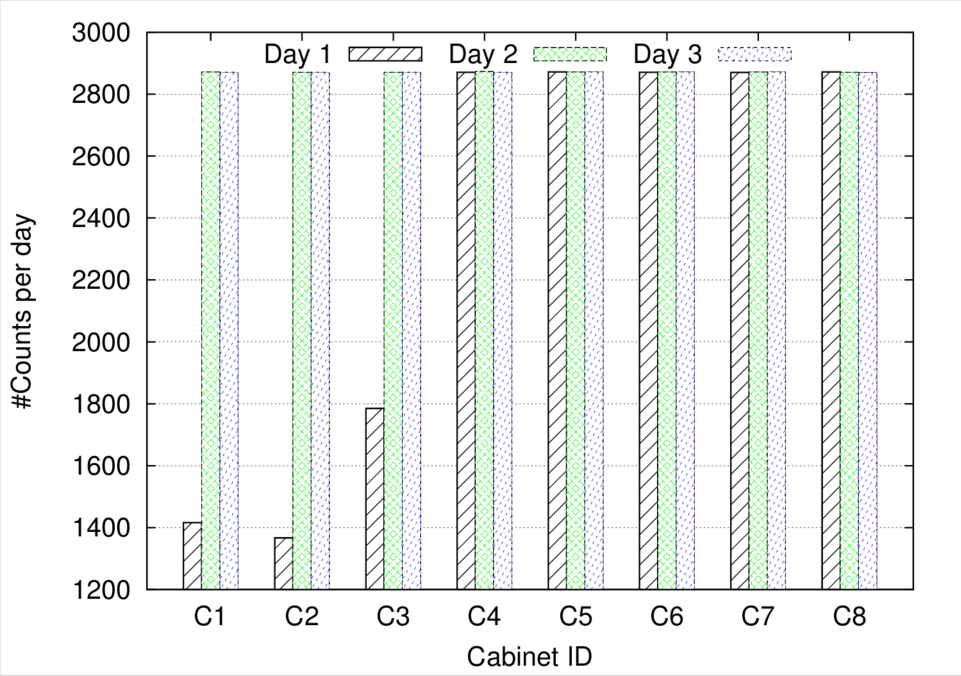
How much does the sedc warnings contribute to the manifested failures?




- Several sedc warnings pertaining to blade, ***do not trigger*** node failures
- There are multiple types of warnings, they occur throughout the day (24 hours) in the order of minutes (exceptions exists, e.g., B7)
- 8 blades underwent health faults (blade-level voltage/temp violations). ***For those 3 days, failed nodes did not correspond to those blades***

# Cabinet Faults

Do the Cabinet faults affect the nodes within the blades in them?




- Cabinet-level sedc faults are higher in logging frequency (in 24 hours) over blades
- Only 32.14% (9/28) nodes belonged to the faulty cabinets
  - ✓ These RPM faults correct themselves without triggering nodes to fail
  - ✓ These faults do not cause failures

# L0_SYSD_MCE

- Blade Controller related or node-specific?
- Usually not coalesced with other indicative faults or errors
- Contains ec_hardware_errors in the event logs
- No more detailed information in the console logs
- At times, can improve the lead time by 1 to 2 minutes

# Job Correlations

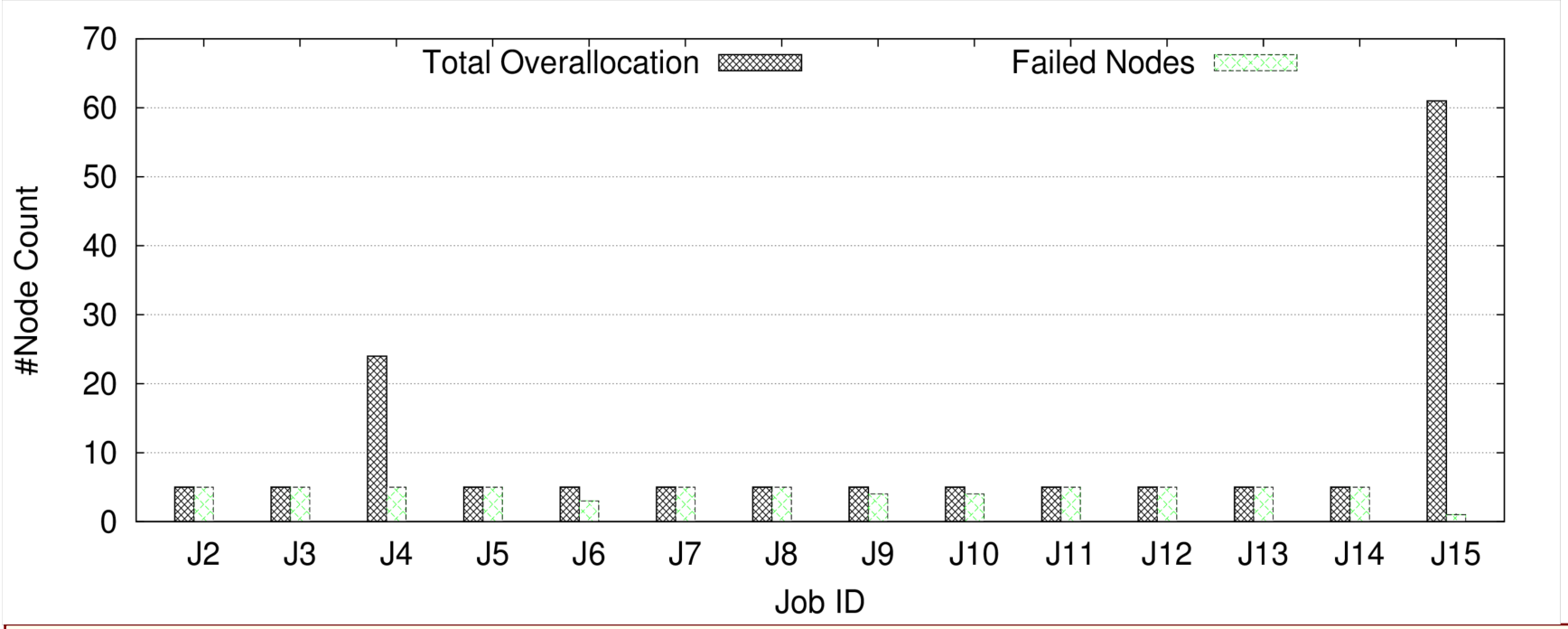
- Analyzed job-based relations for ~80 node failures
- Jobs cause over-allocation of resources throwing errors, with several failures,

e.g., **error: gres/craynetwork job 80117 node nid04551 overallocated resources by 18446744073709551613**

  - All those nodes had similar console messages with similar patterns, indicating **same application based root cause**
  - Typically, job-triggered failures are around the same time, **without logged hardware errors or kernel oops**
  - Nodes were up (no failure indications) next day with **different jobs** scheduled
  - Around similar time-frames (temporal locality) spatially apart nodes fail with different jobs scheduled on them

# Job-based Failures

Do resource overallocation cause failures?




- Specific day: 53 failures, 1 node (no jobs) failed twice, remaining 51 nodes had 16 jobs scheduled (a subset of allocated nodes suffer overallocation error)
- The graph shows what fraction of those overallocated nodes failed
- J1 and J16 had 1 & 6 failures in 600 & 683 total overallocations
- Failed nodes (Green) are a ***subset*** of the Total overallocations (Black)

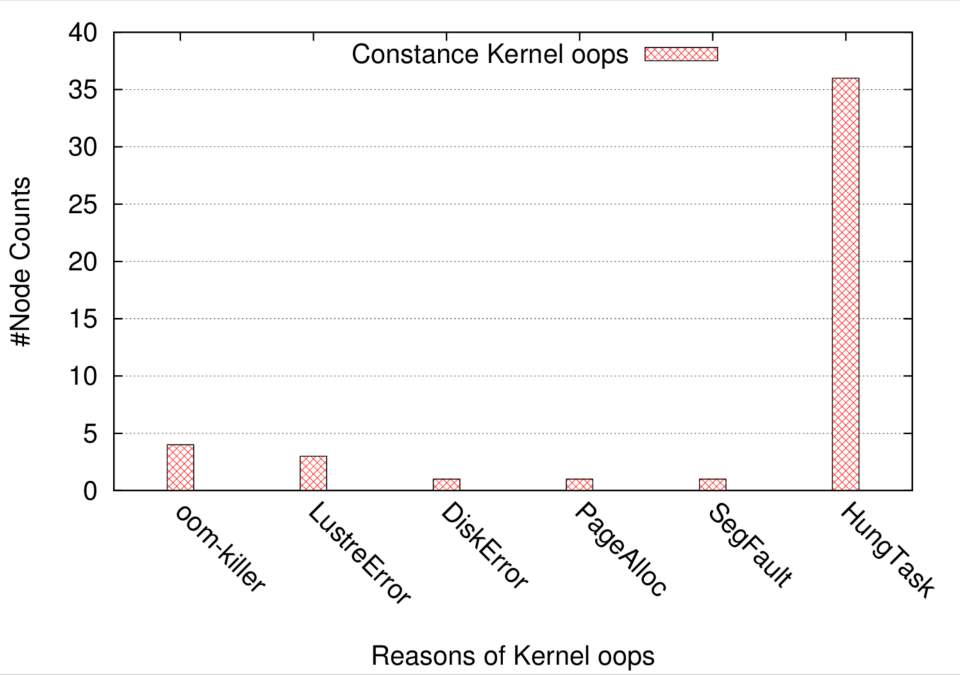
# Analysis of Kernel oops

- Institutional Cluster (PNNL) → Limited data, analyzed 46 nodes with Call Trace Dumps
- Reasons → out of memory, page fault or allocation failure, seg fault, hung_task timeout
- Hopper → Analyzed 56 node failures, LBUG, Application Exit Check, page allocation error or page fault

- These are all application based kernel oops, no additional major tangible hardware or software bugs present

# Constance Categories of Kernel oops

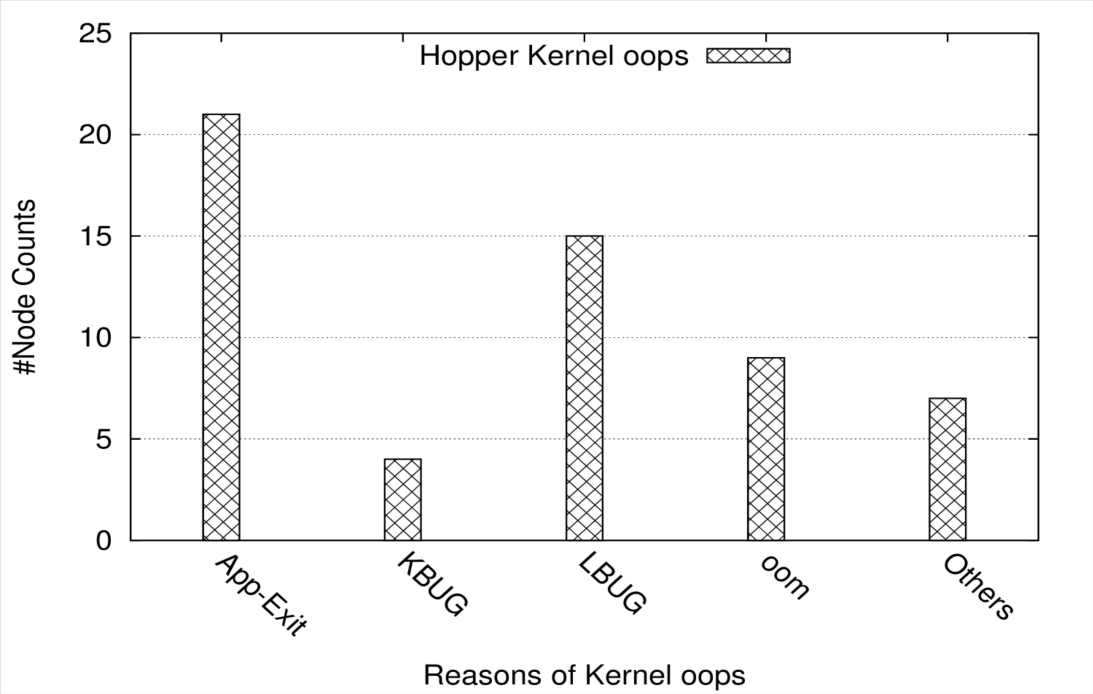
What are the reasons of kernel oops?




- oom kills, may have page faults as well
- Primary root cause app-caused ***memory crunch***
- Hung task – Flushing unable to finish on time due to slow IO

# Hopper Categories of Kernel oops

What are the reasons of kernel oops?




- Process failures, Application exit checks
- Lustre FS Bug (ldlml, race in the code starting threads)
- Kernel Bug (invalid opcode)
- Primary Root Cause: App-triggered **resource exhaustion or FS Bug**

# Root Cause Analysis

- **Application causes**
  - Narrowing down the root causes, Temporal locality without spatial correlation, lead time enhancements not feasible
  - No other major H/W, S/W indicators

- **External causes (not application)**
  - Software Traps under investigation
  - Typical h/w errors, (why processor corruptions happen on certain nodes?)
  - Barring few, environment indicators not helping much yet

# Plans Ahead

- Continue work on RCA
  - Traps
  - Lead time enhancements for non-job triggered failures
  - Analyze more logs
  - Understand the root cause over generic automation