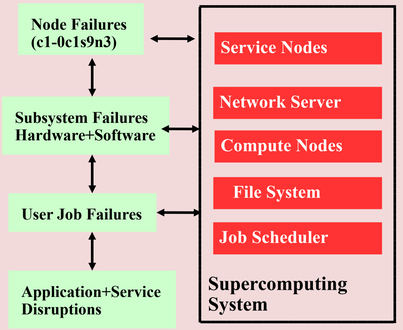
Aarohi: Efficient Online Failure Prediction

Anwesha Das, Frank Mueller --- North Carolina State University

Background

Node Failures cause costly checkpoint/restarts, job failures & service disruption in HPC systems



Exascale Computing Era

Increase in Failures

Lower MTBF (Mean Time Between Failures)

What is required?

- ✓ Pro-actively Flag Failures
- ✓ Procure Short Lead Times
- ✓ Fast Failure Detection
- ✓ Reduce Computation + Energy Wastage

➢ Past M/L based solutions [3]: Limited Scalability, Feature extraction hard
➢ New Systems, Complex Logs: Unsupervised Fast Log mining techniques required

Motivation

System + Job Logs

ERROR: Type:2; Severity:80; Class:3; Subclass:D; Operation: 2

AER: Multiple Corrected error received

MCElog: failed to prefill DIMM database from DMI data

CorrectableMemErr : Link CRC error (cnt: 4)

- ✓ **Log analysis in Cloud+HPC**
- ✓ **Failure Characterization [2]**
- ✓ **Anomaly Detection [1, 3, 4]**
- ✓ **Root Cause Diagnosis**

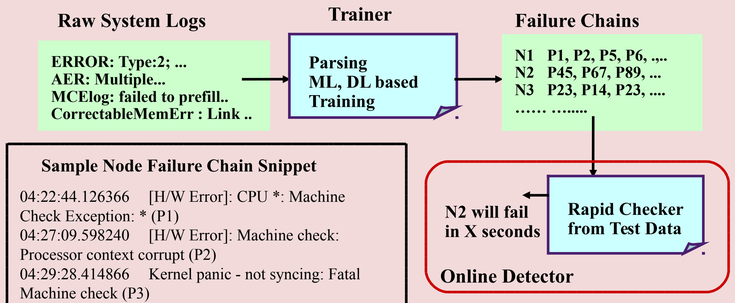
What is missing?

- *Real-time Failure Detection*
- *Accurate lead times with location*
- *Scalable data mining techniques*
- *Reusable paradigms sustainable with system evolution*

Challenges ?

- *Detection speed slow w.r.t log generation speed (msecs to µSecs)*
- *ML based schemes effective off-line trainers, unfit for real-time processing speed*
- *Software/Logging upgrades breaks the detection scheme, minimize overhead*

Research Goals



Sample Node Failure Chain Snippet

04:22:44.126366 [H/W Error]: CPU *: Machine Check Exception: * (P1)
04:27:09.598240 [H/W Error]: Machine check: Processor context corrupt (P2)
04:29:28.414866 Kernel panic - not syncing: Fatal Machine check (P3)
04:30:40.338310 Call Trace: (P4)
04:30:54.620132 Processor has catastrophic error!(P5)
04:32:58.100694 socket * reports MCERR * (P6)
04:33:59.685959 cb_node_unavailable: *_found_in_unavailable_event (P7)

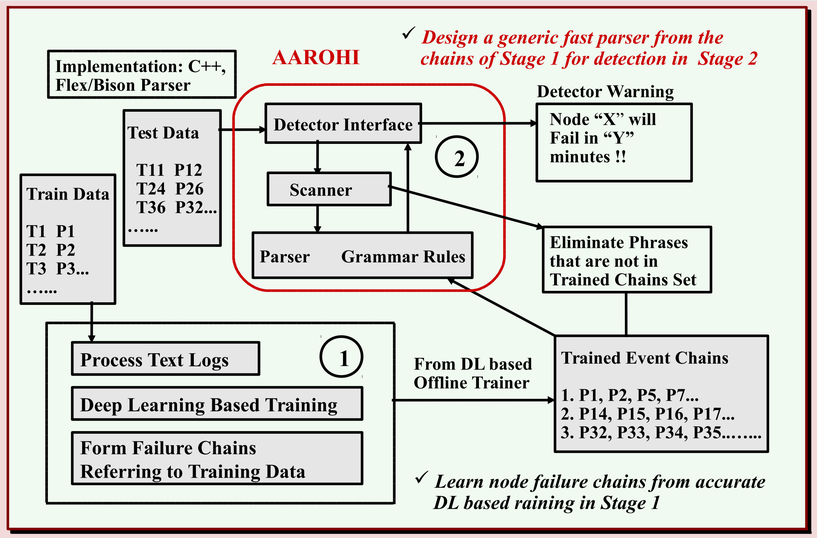
~3 mins lead time achievable if failure is flagged at P5.

Focus - Efficient Failure Detection

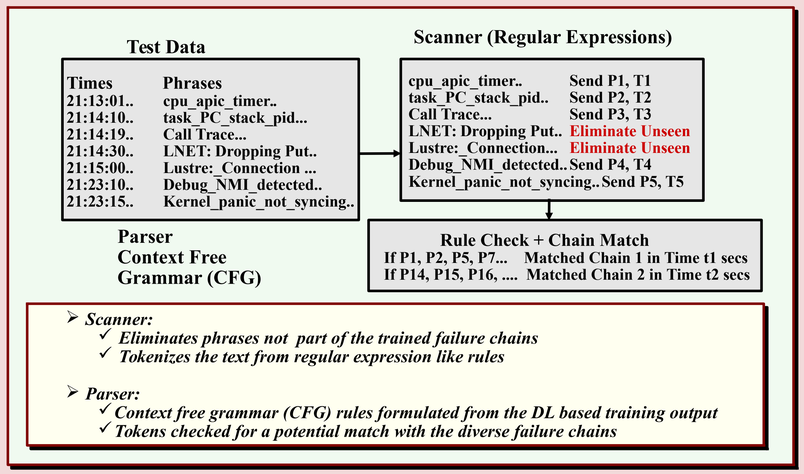
Generic, Scalable, Fast, Adaptive

- ✓ *Raw system logs processed + trained by any Deep Learning (DL) based log mining technique, learn failures from data*
- ✓ *Develop rapid checkers for effective real-time detection*

Solution Design



Solution Design



- ➢ *Scanner:*
  - ✓ *Eliminates phrases not part of the trained failure chains*
  - ✓ *Tokenizes the text from regular expression like rules*
- ➢ *Parser:*
  - ✓ *Context free grammar (CFG) rules formulated from the DL based training output*
  - ✓ *Tokens checked for a potential match with the diverse failure chains*

Results (Extrapolated from Related Work)

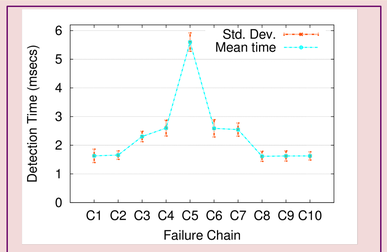


Figure 1

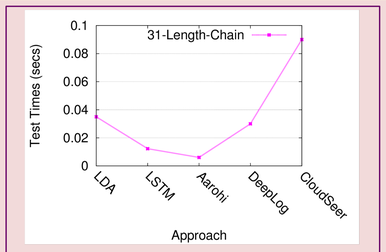


Figure 2

Observations:

- *Figure 1 – The detection time standard deviation is within ± 0.32. The detection times are stable and vary based on the length of the failure chains.*
- *Figure 2 – Aarohi's time complexity is comparatively less than the other popularly used M/L techniques such as LDA, LSTM and related work such as DeepLog[1] & CloudSeer [4]. (Preliminary estimate, extrapolated from the available/obtained performance numbers)*

Results (cont.)

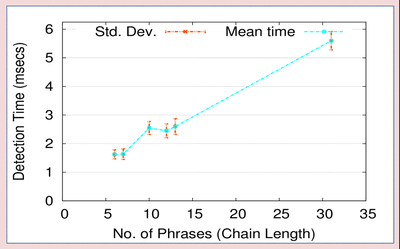


Figure 1

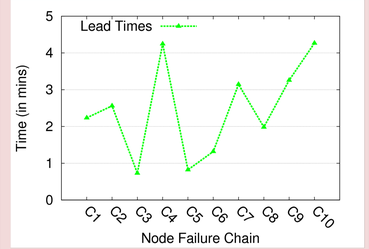


Figure 2

Observations:

- *Figure 1 – Same sized chains have similar detection times. (e.g., C1, C2 & C8 → length 7 → ~1.5 msecs, C3 & C6 → length 12 → ~2.10 msecs)*
- *As #phrases increases from 6 to 31, failure detection time increases from 1.6 to 6 msecs.*
- *The scanning time increases with the chain size.*
- *Figure 2 – Lead times range from ~1 min to 4.2 mins for the considered 10 node failures. These times do not depend upon the chain length but on the time difference between two adjacent event phrases. This time is calculated before the terminal node failure message occurs, higher lead times are possible, if warning is flagged ahead in the failure chain.*

Conclusions

- ➢ ML/DL based learning not fast enough for online detection
  1. Obtain accurate trained failure chains through log mining
  2. Aarohi enhances the detection speed after step 1.
- ➢ Offline trainers require fast stream parsers for real-time processing
- ➢ Aarohi, compiler based approach, detects node failure chains
  - Regular Expressions, Context Free Grammar (CFG)
- ➢ Better detection time
  - scanning, parsing and detection
- ➢ Generic for diverse system types (system specific failure definition)
  - If logs or format change, only scanner will change, parser still remains flexible, needs minimal updates
- ➢ Acceptable lead time
  - Suffices for migration/cloning like recovery actions
- ➢ Further investigation
  - ➢ Time optimization, multi-instance parsing,
  - ➢ Adaptability evaluation with log or software upgrades

References:

[1] M. Du, F. Li, G. Zheng, and V. Srikumar. 2017. DeepLog: Anomaly Detection and Diagnosis from System Logs through Deep Learning. In CCS.
[2] S. Gupta, T. Patel, C. Engelmann, and D. Tiwari. 2017. Failures in Large Scale Systems: Long-term Measurement, Analysis, and Implications. In SC.
[3] Z. Lan, Z. Zheng, and Y. Li. 2010. Toward automated anomaly identification in large-scale systems. IEEE TPDS.
[4] X. Yu, P. Joshi, J. Xu, G. Jin, H. Zhang, and G. Jiang. 2016. Cloudseer: Workflow monitoring of cloud infrastructures via interleaved logs. SIGOPS OS Review.

Acknowledgments: This work was supported in part by DOE subcontracts from Lawrence Berkeley National Lab and Sandia National Lab, a subcontract from Virginia Tech University under contract AFOSR-FA9550-12-1-0442, and NSF grants 1525609 and 0958311.