# BlueGene/L Failure Analysis and Prediction Models

*Liang, Y.; Zhang, Y.; Jette, M.; Anand Sivasubramaniam; Sahoo, R.;*

The International Conference on Dependable Systems and Networks, 2006
(DSN 2006)

Presented by:
Ignacio Laguna
Dependable Computing Systems Lab (DCSL)

PURDUE
UNIVERSITY

---

# The Complexity of Today's Supercomputers

**IBM BlueGene/L at Lawrence Livermore National Laboratory (LLNL)**
The fastest supercomputer (The Top500 Supercomputers list by Ago, 05)

- 64 racks of 128K PowerPC 440 700MHz processors
- Each rack consists of 2 midplanes
- Each midplane has
  - * 1024 processors
  - * 16 node cards
  - * 4 I/O cards
  - * 24 midplane switches
- Huge amount of event logs collected at a centralized point (1,318,137 entries)

- Challenge: How to analyze and (possibly anticipate) failures in such a complex environment?

PURDUE
UNIVERSITY

## Key Contributions of the Paper

- Presented a careful study of collected event logs from BlueGene/L over > 100 days

- Developed three prediction algorithms based on:
    * Failure characteristics
    * Correlation between fatal events and non-fatal events

- Evaluated the effectiveness of the algorithms to anticipate failures

PURDUE
UNIVERSITY

---

## Outline

- Motivation for Failure Prediction
- Description of Event Logs
- Data Processing mechanisms
- Failure Prediction Algorithms
    * Based on failure characteristics
    * Based on the occurrence of non-fatal events
- Summary

PURDUE
UNIVERSITY

## Motivation for Failure Prediction in BlueGene/L

- Different applications may span several thousand processors
  * Hydrodynamics, quantum chemistry, climate modeling

- Failures are becoming a norm rather than an exception
  * Transient hardware failure increasing (from memory to combinational circuits)
  * Permanent hardware device failures leading to immense heat dissipation
  * In addition, software bugs may increase application crashes

- Applications running for a long time may be aborted because of failures → waste of effort

PURDUE
U N I V E R S I T Y

---

## Motivation for Failure Prediction in BlueGene/L

- Low availability impacts response time
  * A real example:
    - LLNL has found frequent L1 cache failures for long running jobs
    - To finish these jobs, L1 cache has been disable for jobs > 4 hours
    - Results → much prolonged execution times for jobs

- Checkpointing techniques are not effective
  * Much overhead compared to the gain
  * Checkpointing a job of tens of thousands tasks may take at least ½ hour

- Failure prediction is considered challenging
  * One reason is the lack of suitable data from real systems

PURDUE
U N I V E R S I T Y

# Event Logs Description

- Event logs have been collected for more than 100 days (~1,3 millions of entries)

- Attributes in each record of the logs:
    * RECID — sequence number of an entry
    * EVENT_TYPE — mechanism through which the event is recorded
    * FACILITY — The component where the event is flagged:
        • LINKCARD, APP, KERNEL, HARDWARE, DISCOVERY, CMCS, BGLMASTER, SERV_NET
    * SEVERITY — denotes increasing order of severity:
        • INFO, WARNING, SEVERE, ERROR, FATAL, or FAILURE (these two ones usually lead to application crashes)
    * EVENT_TIME — timestamp
    * JOB_ID — the job that detect this event
    * LOCATION — a combination of job ID, processor, node, and block (or a separate field)
    * ENTRY_DATA — gives a short description of the event

---

# Data Processing Schemes

- Log entries may be repeated or redundant
    * Filtering tools have been developed in previous work (*DSN 2005*)

- Steps on filtering data:
    1. *Extracting and Categorizing Failure Events*
        * Extract all the events with severity levels of FATAL or FAILURE (called failures).
            * These events will lead to application crashes
    2. *Temporal Compression at a Single Location*
        * Clusters—failure events from the same location that often occur in bursts
    3. *Spatial Compression Across Multiple Locations*
        * A failure can be detected or reported by multiple locations.
        * For example, a network failure is likely detected by multiple locations
        * It removes failures that are close to each other (from the same job) but from different locations

* After these steps, unique failures can be identified

# Filtering Thresholds for Clustering

| Log size with $T_{th} =$ | Memory | Network | APP-IO | Midplane Switch | Node Cards |
|---|---|---|---|---|---|
| 0 | 8,206 | 10,554 | 178,292 | 166 | 96 |
| 30 sec | 267 | 9,418 | 178,015 | 83 | 6 |
| 1 min | 251 | 9,418 | 173,491 | 52 | 6 |
| 5 min | 246 | 9,415 | 102,442 | 30 | 4 |
| 30 min | 241 | 9,219 | 89,333 | 22 | 4 |
| 1 hour | 237 | 8,705 | 81,834 | 17 | 4 |

(a) Number of failure events after temporal filtering using different $T_{th}$

| Log size with $S_{th} =$ | Memory | Network | APP-IO | Midplane Switch | Node Cards |
|---|---|---|---|---|---|
| 0 | 246 | 9,415 | 101,196 | 30 | 4 |
| 30 sec | 217 | 139 | 331 | 30 | 4 |
| 1 min | 217 | 139 | 318 | 30 | 4 |
| 5 min | 215 | 139 | 299 | 30 | 4 |
| 30 min | 208 | 114 | 237 | 22 | 4 |
| 1 hour | 199 | 105 | 225 | 17 | 4 |

(b) Number of failure events after spatial filtering using different $S_{th}$

**Table 1. Filtering thresholds**

---

# Temporal Characteristics of Failures



Number of Fatal Failures

(a) entire system — Days
(b) memory failures — Days
(c) network failures — Days
(d) application I/O failures — Days

PDF

(e) entire system — Inter-failure time (hours)
(f) memory failures — Inter-failure time (hours)
(g) network failures — Inter-failure time (hours)
(h) application I/O failures — Inter-failure time (hours)
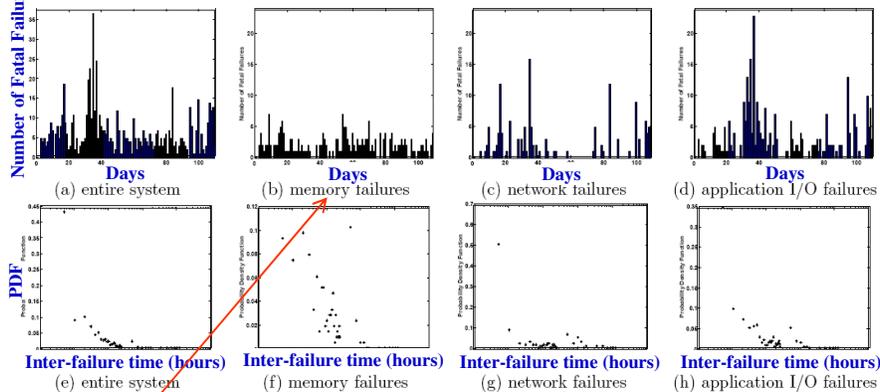
- Memory failures occur everyday
- Many TBF values have comparable likelihood (not a single value dominates)

- Network and application I/O failures occur in burst
- Small TBF values are more popular than larger ones
- For example, 50% of the network failures occur within half an hour after the previous failure

# Failure Prediction Based on TBF

- Failure Prediction Strategy for network and application I/O failures:
  * When a failure is reported, the system is monitored closely for a period of time, since more failures are likely to occur

- Predicting too close failure is not very useful



- This prediction algorithm was run for application I/O and network failures
  * A failure can be predicted by another failure in a window of 5 min ~ 2 hours
    - Rationale: < 5 min is not useful, and > 2 hours incurs in high overhead
  * 52 network failures predicted out of 139 (37%)
  * 143 application I/O failure predicted out of 299 (48%)

---

# Spatial Characteristics of Failures

Number of failures for each midplane



(a) entire system  (b) memory failures  (c) network failures  (d) application I/O failures
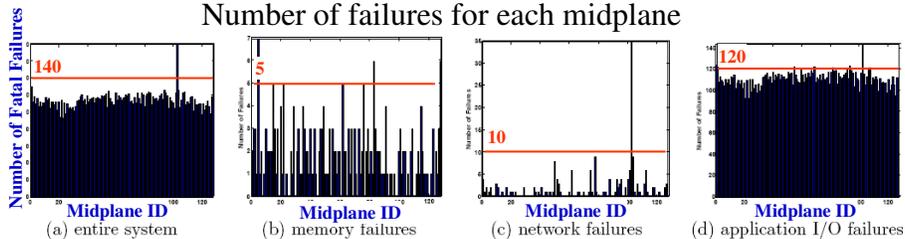
- Memory failures are evenly distributed across all the midplanes
  * 104 out 128 midplanes have reported failures
- All the midplanes have similar probabilities of having memory failures
  * It is hard to predictic memory failures based on spatial characteristics
- Network failures show more pronounced skewness
  * 61 out of 128 midplanes have network failures
  * Midplane 103 alone experiences 35 failures (26% of the total)

# Failure Prediction Based on Spatial Skewness

- For network failures, the focus is for midplanes that has reported more failures than others
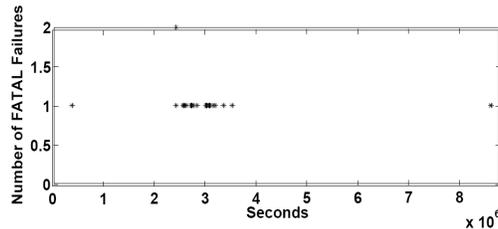


**Figure 5.** The time series of failure occurrence on midplane 103

- Most of the failures on midplane 103 are close to each other
  * A simple prediction strategy is very promising (most of the failures are clustered together)

---

# Predicting Failures Using the Occurrence of Non-Fatal Events

- Correlation between fatal events and non-fatal events is studied

- Conducted a quick experiment to evaluate the likelihood of such correlation
  * Filter all fatal events (by JOB_ID), and see whether the same job has reported non-fatal events before
  * Results of the experiment:

| Type of Fatal Failure | Number of Jobs Terminated by the Fatal Failure | Number of Jobs that Reported One or More Non-Fatal Events Before |
|---|---|---|
| Memory | 134 | 82 |
| Network | 34 | 15 |

- It is promising to predict fatal failures by the use of the occurrence of non-fatal failures!

## Predicting Failures Using the Occurrence of Non-Fatal Events (Cont'd)
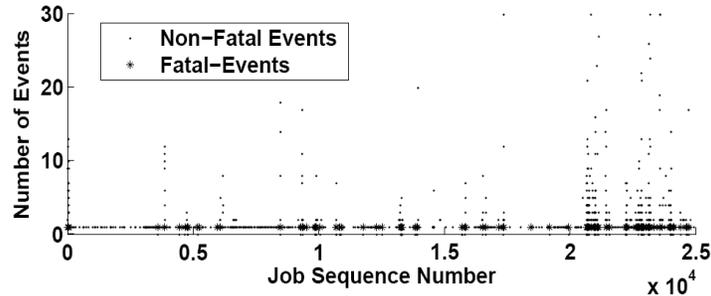


Figure 6. The number of non-fatal and fatal events for each job. On the x-axis, we obtain the job sequence number by subtracting the minimum JOB_ID from the original JOB_ID field of the raw log.

- Counted the number of events each job has encountered (a job has at most one fatal event)

- Large burst of non-fatal events are likely to occur followed by fatal failures

PURDUE
UNIVERSITY

---

## Exploring the Correlation Between Non-Fatal Events and Fatal Events

| $n$ | no. of jobs with $n$ non-fatal events $(x)$ | no. of failures within a window of 5 jobs after these jobs $(y)$ | $y/x$ (%) |
|---|---|---|---|
| $[40, \infty)$ | 4 | 1 | 25 |
| $[20, 40)$ | 9 | 2 | 22.22 |
| $[10, 20)$ | 30 | 8 | 26.67 |
| $[2, 10)$ | 257 | 53 | 20.62 |
| 1 | 1543 | 74 | 4.70 |

(a) The correlation between non-fatal events and fatal events

If a job reports more than 40 non-fatal events, there is a chance of 25% that a fatal failure will occur (in a windows of 5 jobs after it)

- On average, if a job experiences 2 or more non-fatal events, there is a change of 21.33% that a fatal failure will follow
- Prediction Strategy:
  * If a job has observed two non-fatal events, a fatal failure may occur to this job or the following four jobs
  * Results: predicted 65 out of 168 fatal failures

PURDUE
UNIVERSITY

# Summary

- This paper has tackled the challenges of predicting failures at the level of very complex supercomputing systems such as the IBM BlueGene/L

- It has been collected event logs from BlueGene/L over a period of 100 days

- The paper finds strong correlations between the occurrence of a failure and factors such as:
    * Timestamps of other failures
    * Location of other failures
    * The occurrence of non-fatal events

- Three simple predictions schemmes has been proposed for failure prediction
    * Because of their simplicity, this schemes can be implemented a at low runtime cost

PURDUE
U N I V E R S I T Y