Failure avoidance techniques for HPC systems based on failure prediction

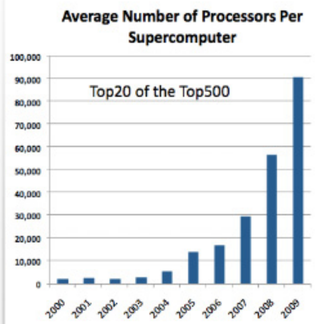
ANA GAINARU

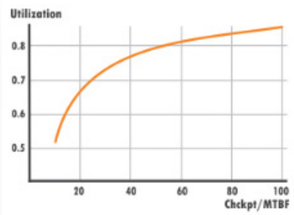
www.ana-gainaru.com
againaru@illinois.edu
University of Illinois
at Urbana-Champaign

Advisor: **Marc Snir**

Motivation

Fear: Exaflop/s system will fail so frequently that no useful work will be possible



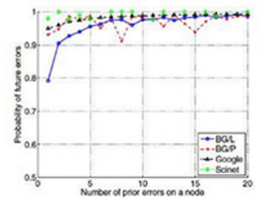


Percentage of wasted time due to checkpointing, as the time to take a checkpoint approaches the MTBF.

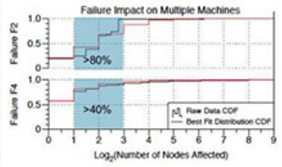
Moving forward:
- More complex application codes - more user errors
- More complex system codes - more "logic" system errors
- Larger system - more "performance" system errors
- More hardware - more hardware errors
- More failure-prone hardware - more hardware errors
- Smaller feature size - more variance, faster aging
- Sub-threshold logic - more bit and multiple-bit upsets

Possible solutions for Exascale:
Multilevel checkpointing - decrease the checkpointing time
Preventive and proactive checkpoiting - increase the MTBF for unpredicted failures

Why prediction



Probability of a node developing memory errors as a function of the number of prior memory errors
(Schroeder et al. ASPLOS12)



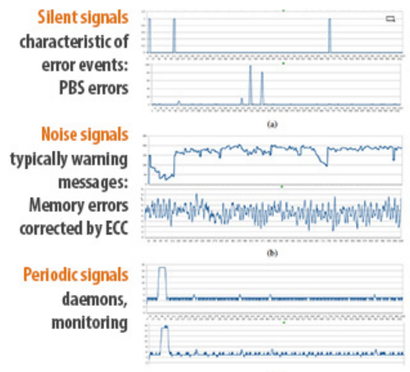
Cumulative Failure Distribution of the number of nodes affected by a failure

Contribution

1. Failure prediction methodology and results on smaller systems
   - Introducing signal processing concepts in the context of log analysis
   - Data mining offers 20% better predictions when combined with signal analysis
2. Combining failure prediction with multi-level checkpointing
   - Recall more important than precision
   - 29% benefit for small systems
3. Analysis of failures and prediction results on Blue Waters
   - Specific predictors needed
   - ~10% benefit for hybrid checkpointing

Failure prediction

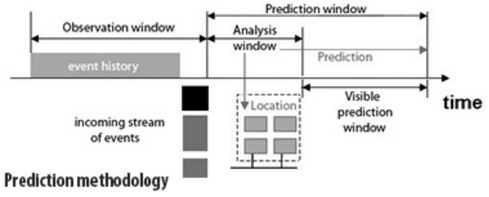
Transforming event logs into signals



ELSA (Event Log Signal Analyzor) - hybrid method combining signal analysis with data mining

Detect outliers
- Changes in frequency and intensity (Wavelet functions)
- Using only signal analysis concepts

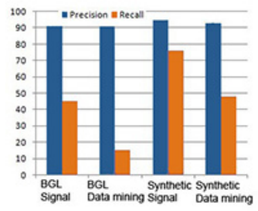
Detect correlations between signals
- Data mining algorithms based on outliers in the signals



Prediction methodology

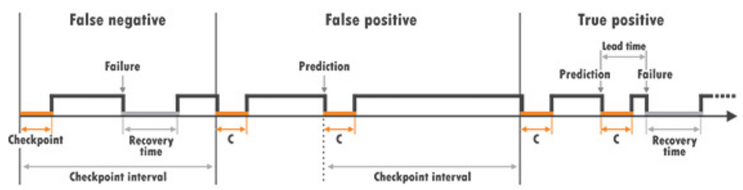
20% of recall increase between ELSA and pure data mining

Overall best precision : 91% with 43% recall
Best recall value : 61% with 36% precision



Preventive and proactive checkpointing

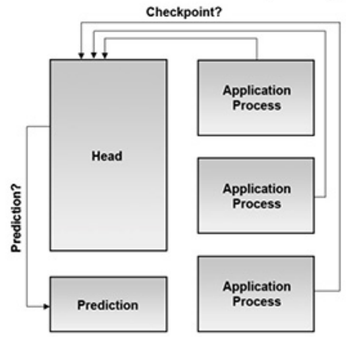
Coupling failure prediction, proactive and preventive checkpoint



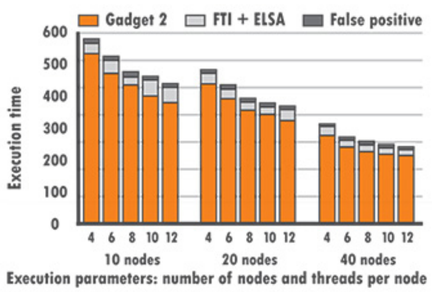
Checkpoints
- Fixed checkpoint interval
- Established by - the recall value (MTB unpredicted F)
  - and checkpointing time
- Resets after each prediction
- Additional checkpoints for each prediction

Depending on the moment of the prediction:
- Math model to decide when take a checkpoint

Framework for combining prediction with FTI (multi-level checkpoiting)



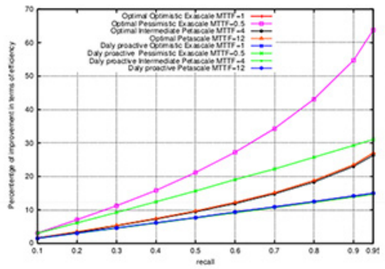
Results



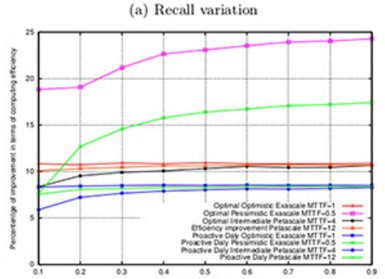
Overhead increase of 2% - 6% compared to FTI alone
False positives - Triggers un-necessary checkpoints
- Adds aditional 1% - 2% overhead

Current Blue Gene/L results show 29% waste reduction With around 12% total overhead (4% over FTI alone)

Paramters	Petascale Jaguar, 10PF	Intermediate 100PF	Exascale Optimistic	Exascale Pessimistic
MTTF	24h to 6h	6h to 4h	2h to 1h	30 min
Preventive Checkpoint time	30 min	10 min	2.5 min	10 min
Proactive Checkpoint time	10 to 5 sec	5 to 1 sec	5 to 1 sec	5 to 1 sec



(a) Recall variation



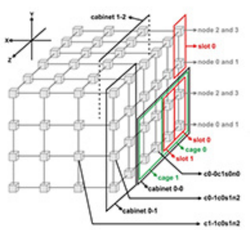
(b) Precision variation

Blue Waters

Blue Waters generates two order of magnitude more events per second

Blue Waters	Events/Day	Total Event types
Syslog	8GB (50 mil events)	3,852
HPSS	1MB (900,000 events)	358
Sonexion	3.5GB (10 mil events)	3,112
Moab	500MB (15 mil events)	725
ESMS	3GB (12 mil events)	2,452
BW Total	~15GB	>10,000
System	**Events/Day**	**Total Event types**
BlueGene/L	5.76MB (25,000 events)	186
BlueGene/P	8.12MB (120,000 events)	252
Mercury	152.4MB (1.5 mil events)	563

Topology aware failure prediction

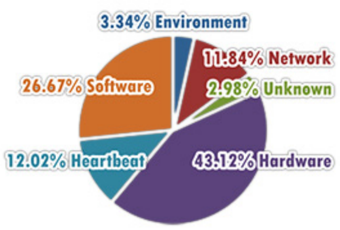


Some cabinets are more error prone than others
- The values are correlated to a couple of performance metrics

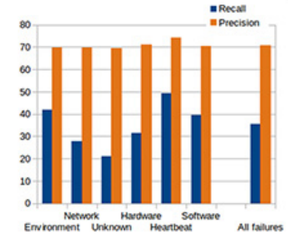
Recall/node
- Not influenced by the count of failures
- Not influenced by usage

After filtering scheduler and filesystem failures
- Overall recall 40%
- Around 50% recall for software failures
- With perfect FS prediction: recall >60%

The majority of failures do not crash applications
- Different effects depending on the type



Percentage of failures for different types



Precision / Recall for each type of failure

Overall total 35% recall
Over prediction of <96 nodes in average

Over prediction affects more Software and environment failures

Future work

Application's perspective
- Around 40% of failures crash application
- Improvement of 5% in recall
  - Luster failures - degrade performance
- Preventive action
  - Should depend on the type of failure
  - Lead time

Performance metrics
- Not all failures are related to the same metrics
  - High temperature are precursors for any-type failures
  - "Moab load" values related to only a subset of scheduler failures
- Different behavior prior to an anomaly
  - Temperature / power follow a linear regression
  - Write accesses present spikes

Specific predictors
- Luster is the main focus
- Preliminary results
  - Correlation between application performance degradation and failures
  - Darshan logs used to detect performance degradation

Location over-prediction
- The over prediction has different cost. Math model :
  - Trade-off between recall and over prefiction
  - Depending on the failure type take different proactive actions