Building a Fault-Aware Computing Environment for High End Computing

Zhiling Lan
Illinois Institute of Technology (Dept. of CS)

In collaboration with
Xian-He Sun, Illinois Institute of Technology

Reliability Concerns

- Systems are getting bigger
  - 1024-4096 processors is today's "medium" size (>54% on the recent TOP500 List)
  - O(10,000)~ O(100,000) processor systems are being designed/deployed
- Even highly reliable HW can become an issue at scale
  - 1 node fails every 10,000 hours
  - 6,000 nodes fail every 1.6 hours
  - 64,000 nodes fail every 5 minutes

☞ Needs for fault management!
Losing the entire job due to one node's failure is costly in time and CPU cycles!

ILLINOIS INSTITUTE OF TECHNOLOGY

The Big Picture

- Checkpoint/restart is widely used for fault tolerance
  - Simple
  - IO intensive, may trigger a cycle of deterioration
  - Reactively handle failures through rollbacks
- Newly emerging proactive methods
  - Good at preventing failures and avoiding rollbacks
  - But, relies on accurate prediction of failure

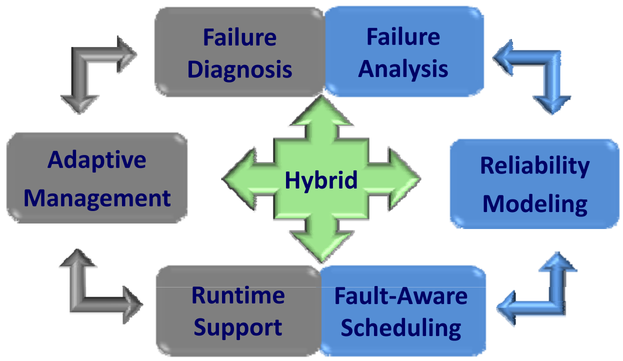
FENCE: Fault awareness ENabled Computing Environment

- **A "fence" to protect system and appl. from severe failure impact**
- **Exploit the synergy between various methods to advance fault management**

FENCE Overview

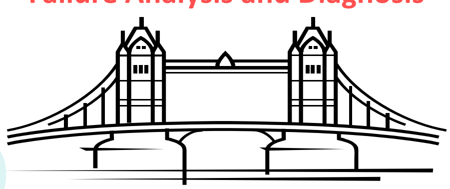
- Adopt a hybrid approach:
  - *Long-term* reliability modeling and scheduling enables intelligent mapping of applications to resources
  - *Runtime* fault resilience support allows applications to avoid imminent failures
- Explore runtime adaptation:
  - *Proactive actions* prevent applications from anticipated failures
  - *Reactive actions* minimize the impact of unforeseeable failures
- Address fundamental issues
  - Failure analysis & diagnosis
  - Adaptive management
  - Runtime support
  - Reliability modeling & scheduling

Key Components



What Is Essential?

Failure Analysis and Diagnosis



Health/perf monitoring:

- Hardware sensors
- System monitoring tools
- Error checking services,

e.g. Blue Gene series and Cray XT series

Fault tolerance technologies

- Checkpointing

(open MPI, MPICH-V, BLCR, ….)

- Process/object migration
- Other resilience supports

Failure Analysis & Diagnosis

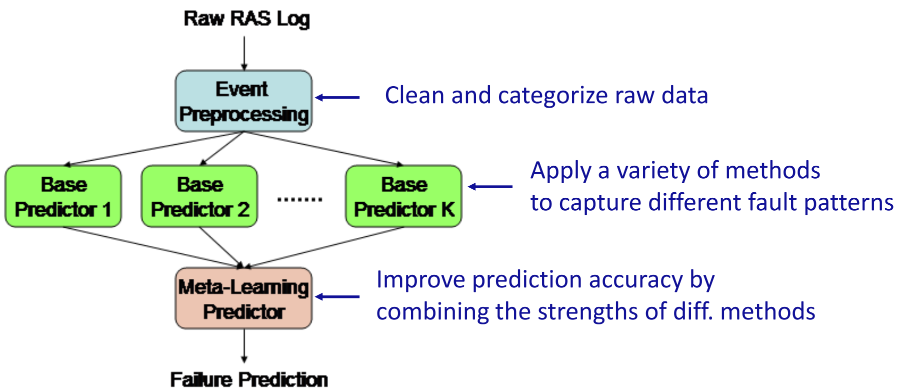
- Goal:
  - To DISCOVER *failure patterns and trends* from data
  - To PROVIDE timely *alerts* regarding *"when and where"* failures are likely to occur
- Challenge:
  - Potentially overwhelming amount of information collected by error checking and monitoring tools
  - Fault patterns and root causes are often buried like needles in a haystack!

➢ How to capture a variety of fault patterns?
➢ How to achieve better diagnosis ?

Failure Analysis & Diagnosis

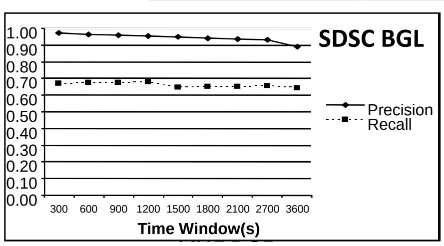
- Our approach:
  - Integrate multiple data sources: RAS log, perf data, sensor readings, …
  - Coordinate data-driven methods: statistical learning, data mining, pattern recognition, ensemble learning (meta-learning)
- The "when" question
  - Ensemble learning based prediction
- The "where" question
  - PCA (Principal component analysis) based localization

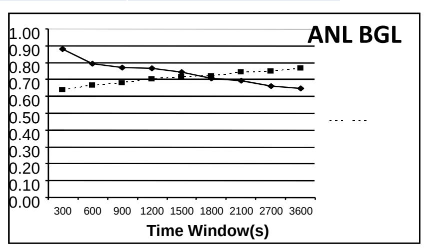
Ensemble Learning Based Prediction



Prediction Results

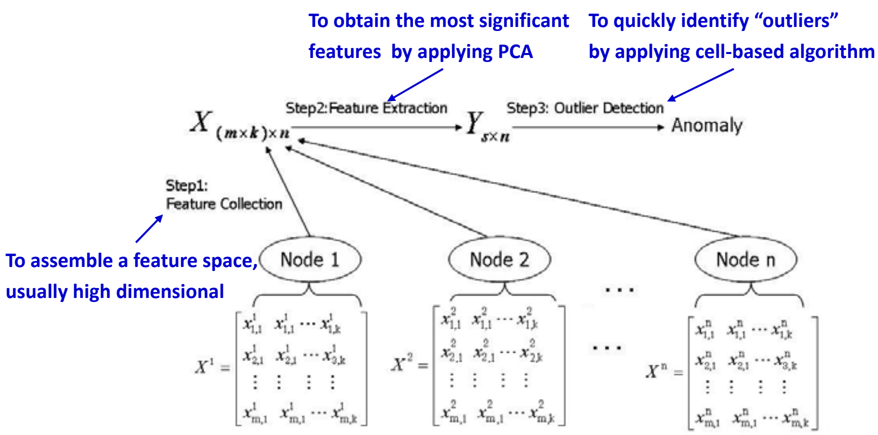
	SDSC BGL	ANL BGL
Start Date	12/6/04	1/21/05
End Date	2/21/06	4/28/06
No. of Records	428,953	4,172,359
Log Size	540MB	5GB





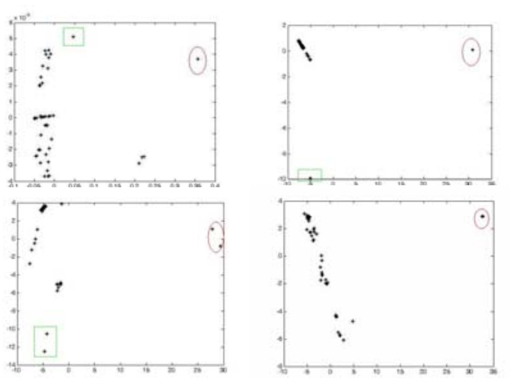
- Captures 65+% of failures, with the false alarm rate less than 35%
- The pattern generation process varies from 35 seconds to 167 seconds; and the matching process is trivial.

PCA based Localization



- Three interrelated steps, with a linear complexity
- A reduced feature space after PCA, e.g. ~97% reduction

Localization Results

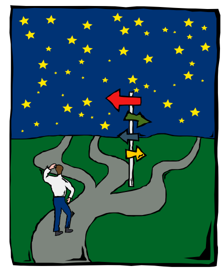


- The ones in the ELLIPSE are correct predictions, and the ones in the RECTANGLE are false alarms
- For 256-node systems, the location method took less than 1.0 second

Faults	Recall	Precision
Memory leaking	1	0.98
Unterminated CPU intensive threads	1	0.80
High frequency IO	1	0.94
Network volume overflow	1	0.85
Deadlock	1	0.94

Adaptive Fault Management

- Runtime adaptation:
  - *SKIP*, to remove unnecessary overhead
  - *CHECKPOINT*, to mitigate the recovery cost in case of unpredictable failures
  - *MIGRATION*, to avoid anticipated failures
- Challenge:
  - Imperfect prediction
  - Overhead/benefit of different actions
  - The availability of spare resources



Adaptive Fault Management

- MIGRATION: $E_{pm} = (2I + C_r + C_{pm}) * f_{appl} + (I + C_{pm}) * (1 - f_{appl})$

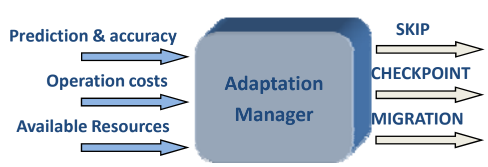
$$\text{where } f_{appl} = \begin{cases} 1 - \prod_{i=1}^{N_W^f - N_S^h} f_p & \text{if } N_W^f > N_S^h \\ 0 & \text{if } N_W^f \le N_S^h \end{cases}$$

- CHECKPOINT: $E_{ckp} = (2I + C_r + C_{ckp}) * f_{appl} + (I + C_{ckp}) * (1 - f_{appl})$

$$\text{where } f_{appl} = 1 - \prod_{i=1}^{N_W^f} f_p$$

- SKIP: $E_{skip} = (C_r + (2 + l_{current} - l_{last}) * I) * f_{appl} + I * (1 - f_{appl})$

$$\text{where } f_{appl} = 1 - \prod_{i=1}^{N_W^f} f_p$$



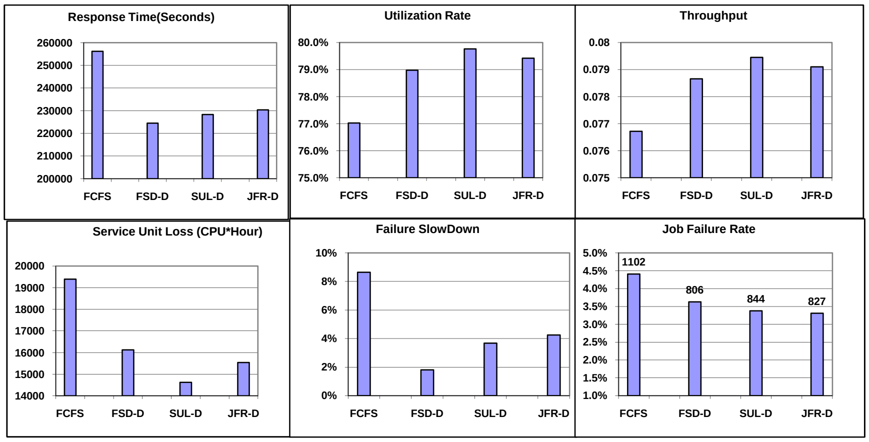
Adaptation Results

- Fluid Stochastic Petri Net (FSPN) modeling
  - Study the impact of computation scales, number of spare nodes, prediction accuracies, and operation costs
- Case studies
  - Implemented with MPICH-VCL
  - Test applications: ENZO, Gromacs, NPB
  - Platform: TeraGrid/ANL IA32 Linux Cluster
- Results:
  - Outperforms periodic checkpointing as long as recall and precision are higher than 0.30
  - A modest allocation of spare nodes (i.e. <5%) is sufficient
  - Lower than 3% overhead

Runtime Support

- Development /optimization of fault tolerance techniques
  - Live migration support
  - Dynamic virtual machine
  - Fast fault recovery
- System-wide node allocation strategy
  - Nodes for regular scheduling vs. spare nodes for failure prevention
- Job rescheduling strategy
  - Selection of jobs for rescheduling in case of multiple simultaneous failures

Results: Runtime Support

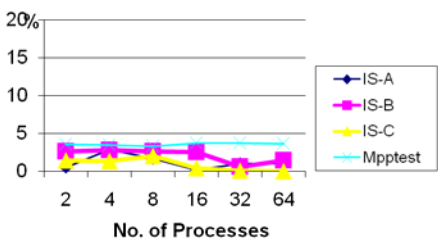


Positive improvement on system productivity, if the failure predictor can capture 20% of failure events with a false alarm rate lower than 80%

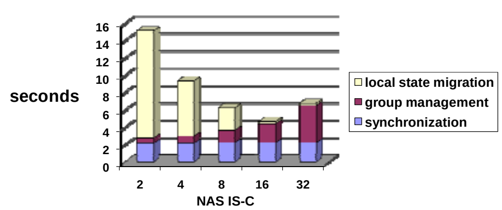
Results: Live Migration

Preliminary results with NAS Parallel Benchmarks and mpptest

– Less than 4% overhead

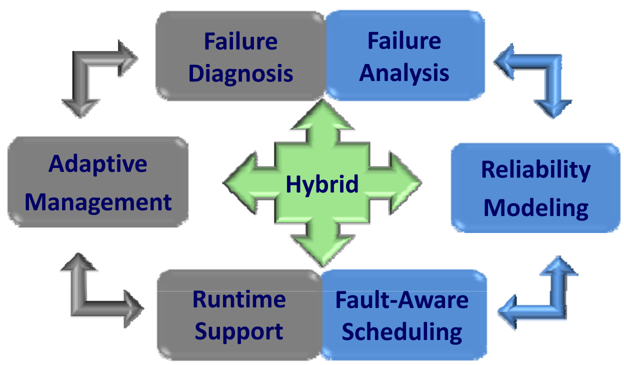


Overhead



Migration Time

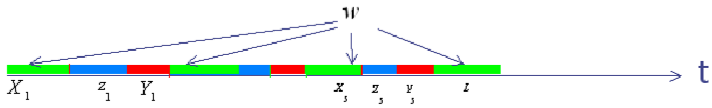
Key Components



Reliability Modeling & Scheduling

- FENCE long-term support:
  - Investigate long-term failure modes, e.g. failure distributions
  - Analyze application performance under failures
  - Apply reliability models for fault-aware scheduling
- SC07 paper: "Performance under Failure of High-end Computing" (Thur. 2:00-2:30pm A2/A5)

Performance Modeling under Failures



The completion time of the application:

$$T = X_1 + Y_1 + Z_1 + X_2 + Y_2 + Z_2 + \ldots + X_s + Y_s + Z_s + L$$

The whole system can be considered as M/G/1 queuing system. We can derive the mean and variance of T, application execution time for single node as:

$$E(T) = \left(\frac{1}{1-\lambda_f \mu_f} + \lambda_f \mu_c\right) w$$

$$V(T) = \left(\frac{\mu_f^2 + \sigma_f^2}{(1-\lambda_f \mu_f)^3} + \mu_c^2 + \sigma_c^2 + 2\frac{\mu_f \mu_c}{1-\lambda_f \mu_f}\right)\lambda_f w$$

Fault-aware Task Partition and Scheduling

Assumption: a parallel task can be partitioned into any size of subtasks. Each subtask will be assigned to a machine respectively.
Objective: scheduling a parallel task heuristically to reach a semi-optimal performance

Begin

List a set of idle machines in the order of their reliability over an observed time period, $M = \{m_1, m_2, \ldots m_q\}$;

Sort the list of idle machines in an decreasing order with $\frac{(1-\rho_{c,k})\tau_k}{1+\rho_{c,k}-\rho_{c,k}\rho_{f,k}}$,

$M' = \{c_1, c_2, \ldots c_q\}$;

$a = 1$, $b = \min\{|M'|, \frac{w}{4*(\mu_{f,k}+\mu_{c,k})}\}$;

Repeat
- $c = \lfloor (a+b)/2 \rfloor$
- /* $f(x)$ denotes $E(T_{C(x)})(1+Coe.(T_{C(x)}))$ where $C(x) = \{c_1, c_2, \ldots c_x\}$ */
- **If** $f(a) = \min\{f(a), f(b), f(c)\}$ **then** $b = c$
- **Else If** $f(b) = \min\{f(a), f(b), f(c)\}$ **then** $a = c$
- **Else If** $f(c) < f(c+1)$ **then** $b = c$
- **Else** $a = c$

Until $a + 1 = b$
If $f(a) < f(b)$ **then**
  Assign parallel task to the machine set $C(a)$;
Else Assign parallel task to the machine set $C(b)$;
End

Figure 7. A heuristic fault-aware task scheduling algorithm

Work In Progress

- Complete prototype systems
  - Failure analysis & diagnosis toolkit
  - Adaptive fault management library for HEC applications
  - Job scheduling/rescheduling support
- Investigate advanced predictive methods
- Provide better integration and coordination support
- Conduct extensive assessment

Conclusions

- FENCE (Fault awareness ENabled Computing Environment) to advance fault management
  - Potential for better failure analysis and diagnosis
    - Captures 65+% of failures, with the false alarm rate less than 35%
  - Up to 50% improvement in system productivity
  - Up to 43% reduction in application completion time

"Adaptation is key" (D. Reed)

"It is not cost-effective or practical to rely on a single fault tolerance approach for all applications and systems" (Scarpazza, Villa, Petrini, Nieplochar, …)

Questions?

FENCE Project Website:

http://www.cs.iit.edu/~zlan/fence.html

SCS Lab Website:

http://www.cs.iit.edu/~scs

THANK YOU To our sponsor: National Science Foundation