Resource Availability Prediction in Fine-Grained Cycle Sharing Systems

Xiaojuan Ren, Seyong Lee, Rudolf Eigenmann, Saurabh Bagchi
School of ECE, Purdue University

Presented by: Saurabh Bagchi

Work supported by
National Science Foundation

Greetings come to you from …

What are Cycle Sharing Systems?

- Systems with following characteristics
  - Harvests idle cycles of Internet connected PCs
  - Enforces PC owners' priority in utilizing resources
  - Resource becomes unavailable whenever owners are "active"
- Popular examples: SETI@Home, protein folding

What are Fine-Grained Cycle Sharing Systems?

- Cycle Sharing systems with following characteristics
  - Allows foreign jobs to coexist on a machine with local ("submitted by owner") jobs
  - Resource becomes unavailable if slowdown of local jobs is observable
  - Resource becomes unavailable if machine fails or is intentionally removed from the network

Fine-Grained Cycle Sharing: FGCS

Trouble in “FGCS Land”

- Uncertainty of execution environment to remote jobs
- Result of fluctuating resource availability
  - Resource contention and revocation by machine owner
  - Software-hardware faults
  - Abrupt removal of machine from network
- Resource unavailability is not rare
  - More than 400 occurrences in traces collected during 3 months on about 20 machines

How to handle fluctuating resource availability?

- Reactive Approach
  - Do nothing till the failure happens
  - Restart the job on a different machine in the cluster
- Proactive Approach
  - Predict when resource will become unavailable
  - Migrate job prior to failure and restart on different machine, possibly from checkpoint
- Advantage of proactive approach: Completion time of job is shorter

IF, prediction can be done accurately and efficiently

Our Contributions

Prediction of Resource Availability in FGCS

- Multi-state availability model
  - Integrates general system failures with domain-specific resource behavior in FGCS
- Prediction using a semi-Markov Process model
  - Accurate, fast, and robust
- Implementation and evaluation in a production FGCS system

Outline

- Multi-State Availability Model
  - Different classes of unavailability
  - Methods to detect unavailability
- Prediction Algorithm
  - Semi-Markov Process model
- Implementation Issues
- Evaluation Results
  - Computational cost
  - Prediction accuracy
  - Robustness to irregular history data

Two Types of Resource Unavailability

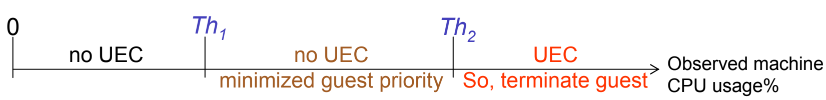
- ***UEC*** – Unavailability due to Excessive Resource Contention
  - Resource contention among one ***guest*** job and ***host*** jobs (CPU and memory)
  - Policy to handle resource contention: Host jobs are sacrosanct
    - Decrease the guest job's priority if host jobs incur *noticeable slowdown*
    - Terminate the guest job if slowdown still persists
- ***URR*** – Unavailability due to Resource Revocation
  - Machine owner's intentional leave
  - Software-hardware failures

Detecting Resource Unavailability

- UEC
  - *Noticeable slowdown* of host jobs cannot be measured directly
  - Our detection method
    - Quantify slowdown by reduction of host CPU usage (> 5%)
    - Find the correlation between observed machine CPU usage and effect on host job due to contention from the guest job
- URR
  - Detected by the termination of Internet sharing services on host machines

Empirical Studies on Resource Contention

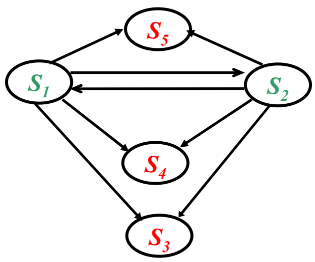
- CPU Contention
  - Experiment settings
    - CPU-intensive guest process
    - *Host group*: Multiple host processes with different CPU usages
    - Measure CPU reduction of host processes for different sizes of host group
    - 1.7 GHz Redhat Linux machine
  - Observation
    - UEC can be detected by observing machine CPU usage on Linux systems



Empirical Studies on Resource Contention (Cont.)

- Evaluate effect of CPU and Memory Contention
- Experiment settings
  - Guest applications: SPEC CPU2000 benchmark suite
  - Host workload: Musbus Unix benchmark suite
  - 300 MHz Solaris Unix machine with 384 MB physical memory
  - Measure host CPU reduction by running a guest application together with a set of host workload
- Observations
  - Memory thrashing happens when processes desire more memory than the system has
  - Impacts of CPU and memory contention can be isolated
  - The two thresholds, Th_1 and Th_2 can still be applied to quantify CPU contention

Multi-State Resource Availability Model



S_1: Machine CPU load is $[0\%, Th_1]$

S_2: Machine CPU load is $(Th_1, Th_2]$

S_3: Machine CPU load is $(Th_2, 100\%]$ -- UEC

S_4: Memory thrashing -- UEC

S_5: Machine unavailability -- URR

For guest jobs, S_3, S_4, and S_5 are unrecoverable failure states

Resource Availability Prediction

- Goal of Prediction
  - Predict temporal reliability (TR)

    The probability that resource will be available throughout a future time window
- Semi-Markov Process (SMP)
  - States and transitions between states
  - Probability of transition to next state depends only on current state and amount of time spent in current state (independent of history)
- Algorithm for TR calculation:
  - Construct an SMP model from history data for the same time windows on previous days

    Daily patterns of host workloads are comparable among recent days
  - Compute TR for the predicted time window

Why SMP?

- Applicability – fits the multi-state failure model
  - Bayesian Network models
- Efficiency – needs no training or model fitting
  - Rules out: Neural Network models
- Accuracy – can leverage patterns of host workloads
  - Rules out: Last-value prediction
- Robustness – can accommodate noises in history data

Background on SMP

- Probabilistic Models for Analyzing Dynamic Systems

  S : state

  Q : transition probability matrix

  $Q_{i,j}$ = Pr { the process that has entered S_i will enter S_j on its next transition };

  H : holding time mass function matrix

  $H_{i,j}(m)$ = Pr { the process that has entered S_i remains at S_i for m time units before the next transition to S_j }

- Interval Transition Probabilities, P

  $P_{i,j}(m) = \Pr\{S(t_0+m)=j \mid S(t_0)=i\}$

Solving Interval Transition Probabilities

- Continuous-time SMP
  - Backward Kolmogorov integral equations

$$P_{i,j}(m) = \sum_{k \in S} \int_0^m Q_i(u) \times H'_{i,k}(u) \times P_{k,j}(m-u)du$$

Too inefficient for online prediction

- Discrete-time SMP
  - Recursive equations

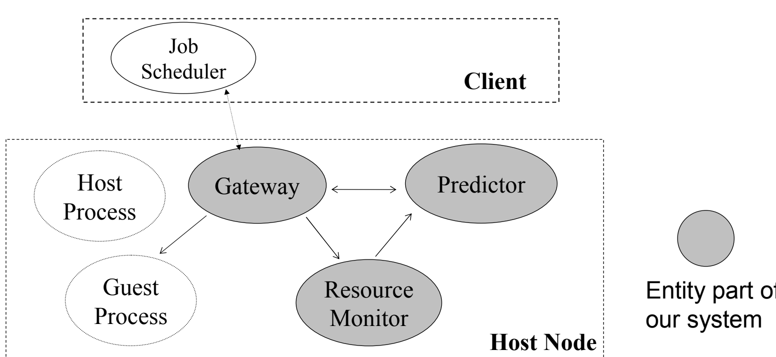
$$P_{i,j}(m) = \sum_{l=1}^{m-1} P^1_{i,k}(l) \times P_{k,j}(m-l) = \sum_{l=1}^{m-1} \sum_{k \in S} Q_i(k) \times H_{i,k}(l) \times P_{k,j}(m-l)$$

- Availability Prediction

  TR(W): the probability of not transferring to S_3, S_4, or S_5 within an arbitrary time window, W of size T

$$TR(W) = 1 - [P_{init,3}(T/d) + P_{init,4}(T/d) + P_{init,5}(T/d)]$$

System Implementation



Non-intrusive monitoring of resource availability

- UEC – use lightweight system utilities to measure CPU and memory load of host processes in non-privileged mode
- URR – record timestamp for recent resource measurement and observe gaps between measurements

Evaluation of Availability Prediction

- Testbed
  - A collect of 1.7 GHz Redhat Linux machines in a student computer lab at Purdue
    - Reflect the multi-state availability model
    - Contain highly diverse host workloads
  - 1800 machine-days of traces measured in 3 months
- Statistics on Resource Unavailability

Categories	Total amount	UEC		URR
		CPU contention	Memory contention	
Frequency	405-453	283-356	83-121	3-12
Percentage	100%	69-79%	19-30%	0-3%

Evaluation Approach

- Metrics
  - Overhead: monitoring and prediction
  - Accuracy
  - Robustness
- Approach
  - Divide the collected trace into training and test data sets
  - Parameters of SMP are learnt based on training data
  - Evaluate the accuracy by comparing the prediction results for test data
  - Evaluate the robustness by inserting noise into training data set

Reference Algorithms:
Linear Time Series Models

– Widely used for CPU load prediction in Grids: Network Weather Service*

– Linear regression equations**

– Application in our availability prediction

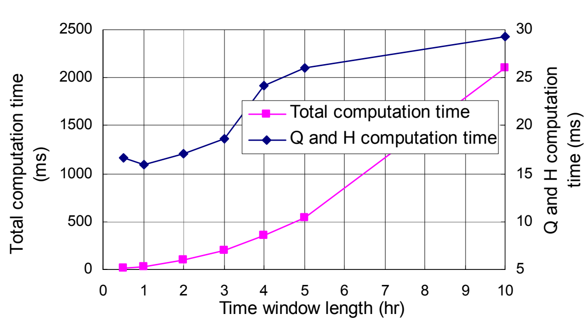
- Predict future system states after observing training set
- Compare the observed TR on the predicted and measured test sets

*R. Wolski, N. Spring, and J. Hayes, The Network Weather Service: A Distributed Resource Performance Forecasting Service for Metacomputing, *JFGCS*, 1999

** Toolset from P. A. Dinda and D. R. O'Halaron. "An evaluation of linear models for host load prediction". In Proc. Of HPDC'99.

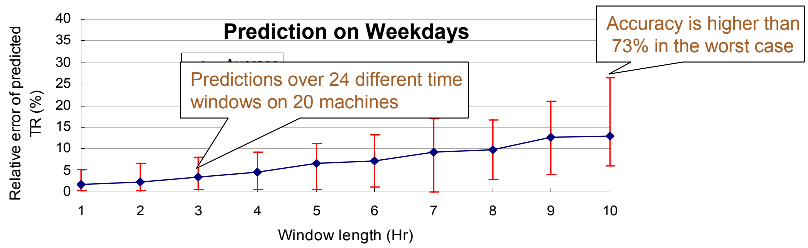
Overhead

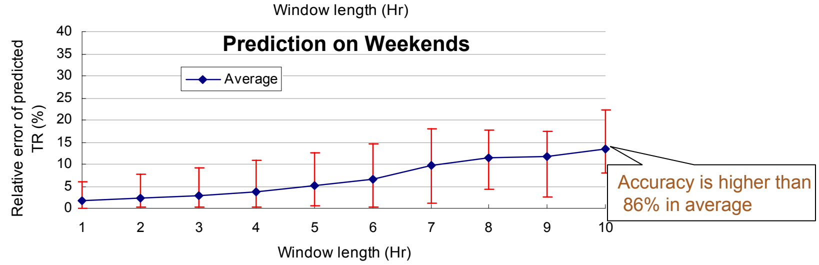
- Resource Monitoring Overhead: CPU 1%, Memory 1%
- Prediction Overhead



Less than 0.006% overhead to a remote job

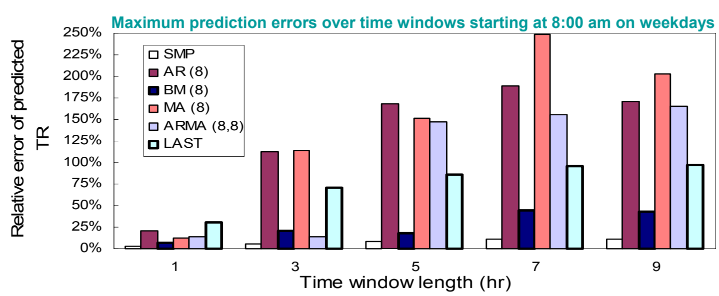
Prediction Accuracy





Relative error = abs($TR_{predicted}$-$TR_{empirical}$)/$TR_{empirical}$

Comparison with Linear Time Series Models

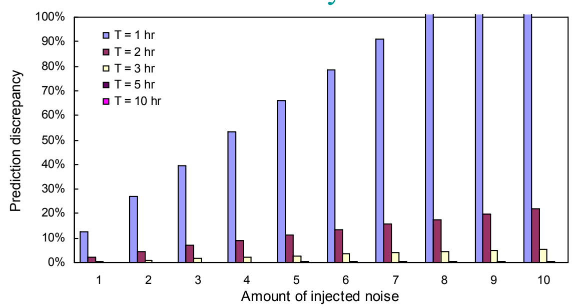


Resource Prediction System: http://www.cs.cmu.edu/~cmcl/remulac/remos.html

Model	Description
AR(p)	Purely autoregressive models with p coefficients
BM(p)	Mean over the previous N values (N < p)
MA(p)	Moving average models with p coefficients
ARMA(p,q)	Autoregressive moving average models with p+q coefficients
LAST	Last measured values

Prediction Robustness

Randomly insert unavailability occurrences between 8:00-9:00 am on a weekday trace



1) Predictions on smaller time windows are more sensitive
2) On large time windows (> 2 hours), intensive noise (10 occ urrences within one hour) causes less than 6% disturbance in the prediction

Summary on Related Work

- Fine-grained cycle sharing with OS kernel modification Ryu and Hollingsworth, *TPDS*, 2004
- Critical event prediction in large-scale clusters Sahoo, et. al., *ACM SIGKDD*, 2003
- CPU load prediction for distributed compute resources Wolski, et. al., *Cluster Computing*, 2000
- Studies on CPU availability in desktop Grid systems Kondo, et. al., *IPDPS*, 2004

Conclusion

- For practical FGCS systems, runtime prediction of resource unavailability is important
- Resource unavailability may occur due to resource contention or resource revocation
- Our prediction system based on an SMP model is
  - Fast: < 0.006% overhead
  - Accurate: > 86% accuracy in average
  - Robust: < 6% difference caused by noise
- Generality
  - Testbed contains highly diverse host workloads
  - Accuracy was tested on workloads for different time windows on weekdays/weekends

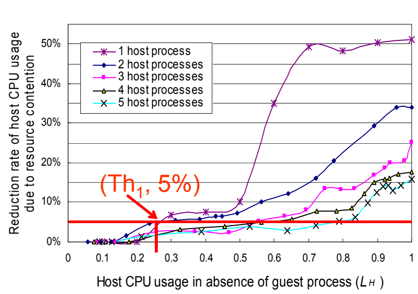
Thanks!

Backup Slides

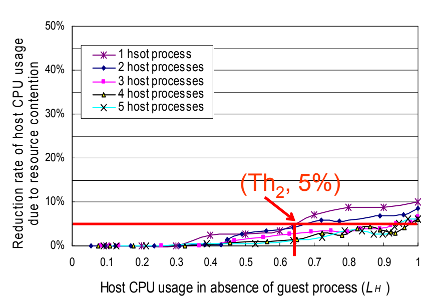
- Resource contention studies, 27-29
- Linux scheduler, 30
- Details on reference algorithms for failure prediction, 31

Empirical Studies on Resource Contention

- CPU Contention
  - CPU-intensive guest applications
  - host groups consisting of multiple processes with diverse CPU usage
  - 1.7 GHz Redhat Linux machine

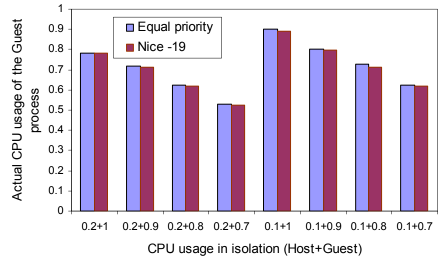


All processes have the same priority

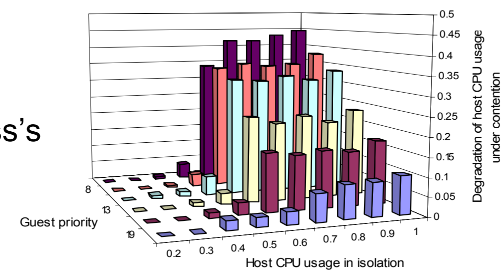


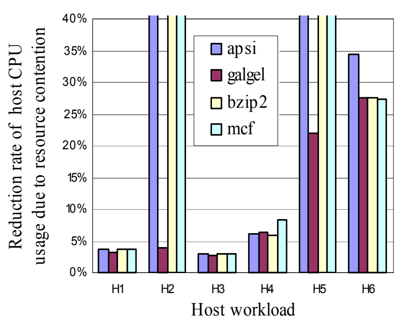
Guest process takes the lowest priority

Restrict resource contention by minimizing guest process's priority from its creation

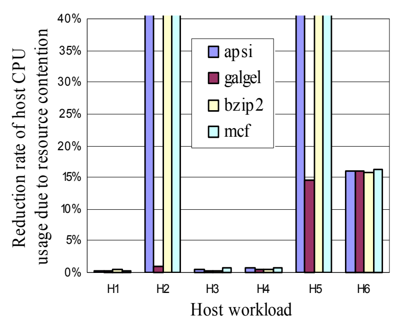


Restrict resource contention by finely tuning guest process's priority





Guest process with priority 0



Guest process with priority 19

- Memory thrashing happens when processes desire more memory than the system has
- Impacts of CPU and memory contention can be isolated
- The two thresholds, Th_1 and Th_2, can still be applied to quantify CPU contention

```
While (1) {
   If exists p such that p.state = RUNNABLE
      Foreach process p
         p.quanta = 20 + p.niceLevel + 1/2 * p.quanta;
   While exists a process p
      such that (p.state = RUNNABLE) and (p.quanta > 0)
         Select p with largest p.quanta;
         Decrement p.quanta;
         Run p;
}
```

Linux CPU scheduler

Details on Reference Algorithms

- AR(p) – An autoregressive model is simply a linear regression of the current value of the series against one or more prior values of the series. p is the order of the AR model. Linear least squares techniques (Yule-Walker) are used for model fitting.
- BM(p) – Average on previous N values. N is chosen to minimize the squared error
- MA(p) - A moving average model is conceptually a linear regression of the current value of the series against the white noise or random shocks of one or more prior values of the series. Iterative non-linear fitting procedures (Powell's methods) need to be used in place of linear least squares.
- ARMA(p,q) - a model based on both previous outputs and their white noise
- LAST – the previous observations from the last time window of the same length are used for prediction