Performance Prediction Engineering

Francine Berman
U. C. San Diego

Rich Wolski
U. C. San Diego and University of Tennessee
The Computational Grid

**Computer** = ensemble of resources

“**Computer**” may consist of

- computational sites
- dist. databases
- remote instruments
- visualization
- distinct networks
Grid Programs

Grid programs

– may couple distributed and dissimilar resources
– may incorporate tasks with different implementations
– may adapt to dynamic resource load
Performance Models for Grid Programs

• Grid applications may couple dissimilar resources
  – models must accommodate heterogeneity

• Grid applications may incorporate tasks with different implementations
  – model must accommodate multiple task models

• Grid applications may adapt to dynamic resource load
  – models must allow for dynamic parameters
Compositional Models

• Grid programs can be represented as a composition of tasks

• “Tasks” consist of relevant performance activities

• Model parameters may reflect performance variations of grid
  – may be parameterized by time
Using Grid Performance Models

• Compositional models particularly useful for grid application scheduling

• Application schedulers use performance prediction models to
  – select resources
  – estimate potential performance of candidate schedules
  – compare possible schedules
AppLeS = Application-Level Scheduler

- NWS
- User Pref
- App Perf Model

- Resource Selector
- Planner
- Application
- Act.

Grid infrastructure and resources
Partitionings

- Block partitioning
- Compile-time non-uniform strip partitioning
- AppLeS dynamic strip partitioning
Application Scheduling Jacobi2D

Dynamic information key to leveraging deliverable performance from the Grid environment

Comparison of Execution Times

Execution Time (seconds)

Problem Size

HPF Uniform/Blocked
Non-uniform Strip
AppLeS
Performance is Time-Dependent

Jacobi2D AppLeS (strip) vs. Block partitioning
Schedulers and Performance Models

• Predictions may be used at different levels of accuracy
  – predictions can be “engineered”

• Knowing something about a prediction can make it more useful
  – performance range of predictions may provide additional information
  – meta-information about predictions can improve schedules
Performance Prediction Engineering

- *Performance Prediction Engineering (PPE) System* is a methodology for modeling performance in dynamic Grid environments

- **3 Components:**
  - Structural performance prediction models
  - Quantitative meta-information
  - Dynamic Forecasting
Structural Models

- **Top-level Model** = performance equation
  - describes composition of application within a specific time frame (*performance grammar*)

- **Component models**
  - represent application performance activities (*nonterminals*)

- **Model parameters**
  - represent system or application values (*terminals*)
Example: Modeling the Performance of SOR

- Regular, iterative computation
- 5 point stencil
- Divided into a red phase and a black phase
- 2D grid of data divided into strips
- Targeted to WS cluster
SOR Structural Model

**SOR performance equation**

\[
\text{ExecTime}(t_0) = \sum_{i=0}^{n} \text{IterTime}(t_i)
\]

\[
\text{IterTime}(t_i) = \max_p \{ \text{RComp}(p, t_i) + \text{RComm}(p, t_i + \Delta_1) + \text{BComp}(p, t_i + \Delta_2) + \text{BComm}(p, t_i + \Delta_3) \}
\]

**SOR component models**

\{ \text{RComp}(p,t), \text{RComm}(p,t), \text{BComp}(p,t), \text{BComm}(p,t) \}
SOR Component Models

\[ R_{Comp}(p,t) = \frac{\text{NumElts}(p) \times \text{Benchmark}(p, \text{Elt})}{\text{FracAvailCPU}(p,t)} \]

\[ R_{Comm}(p,t) = \frac{\text{ColumnSize} \times \text{Size}(\text{Elt})}{\text{BWAvail}(p, p + 1, t)} \]

\[ + \frac{\text{ColumnSize} \times \text{Size}(\text{Elt})}{\text{BWAvail}(p, p - 1, t')} \]

Dynamic Parameters

\[ \text{FracAvailCPU}(p,t), \text{ BWAvail}(x,y,t) \]
Single-User Experiments

• **Question**: How well does the SOR model predict performance in a single-user cluster?

• **Platform**
  - heterogeneous Sparc cluster
  - 10 Mbit ethernet connection
  - quiescent machines and network

• **Prediction within 3% before memory spill**
Dedicated Platform Experiments

What happens when other users share the system?
Non-dedicated SOR Experiments
Improving Predictions

• Many parameters represent values which vary over time

• Range of behavior of time-dependent parameters represented by distributions

• Structural models can be extended to accommodate stochastic parameters and render stochastic predictions
Stochastic Predictions

Stochastic predictions capture range of possible behavior
Stochastic Structural Models

- Stochastic and point-valued parameters
- Component models

“Quality” of performance prediction (lifetime, accuracy, overhead)

Stochastic predictions
Stochastic SOR Performance Model

- $\text{FracAvailCPU, BWAvail}$ given by stochastic parameters

- **Network Weather Service** improved to provide better performance information

- **First cut:** consider stochastic parameters which can adequately be represented by normal distributions
  
  - normal distributions make math tractable
Experiments with Multi-user Systems

• Platform
  – Sun workstation cluster
  – 10Mbit ethernet
  – experiments run in lab environment with additional generated load

• Experiments run back-to-back for multiple trials
SOR Stochastic Parameters

Bandwidth Histogram

Normal PDF

Percentage of values equal to X

Bandwidth (Mbits/sec)

BWAvail

Number of values equal to X

Available CPU

FracAvailCPU
Data stays within single mode

Data changes modes
“Single-mode” Experiments

• All values captured by stochastic predictions
• Maximum absolute error between means and actual values is 10%
“Multiple Mode” Experiments

- 80% of actual values captured by stochastic prediction
- Max discrepancy between stochastic prediction and actual values is 14%
- Max absolute error between means and actual values is 39%
The Next Step

What if performance range of parameters cannot be adequately represented by normal distributions?

- Can we identify distributions for model parameters?
- Can we combine non-normal distributions efficiently? Is the math tractable?
- Can we use empirical data to determine performance ranges if distributions cannot be identified?
Using PPE for Application Scheduling

Basic Strategy:

- Develop **structural model** for application
- Use **stochastic parameters** to provide information about performance range
- Use profiling to determine desired level of accuracy for component models
- Use stochastic prediction and **meta-information** to develop application schedule
Scheduling with Meta-Information

• Stochastic predictions provide information about range of behavior

• Stochastic predictions and meta-information provide additional information for schedulers
Quality of Information

• Meta-information = Quality of Information

• SOR stochastic predictions provide a measure of accuracy

• Other qualitative measures are possible
  – lifetime
  – overhead
  – complexity

• Quality of Information attributes can be used to improve scheduling
Preliminary Experiments: Application Scheduling with PPE

Simple scheduling scenario:

- SOR with strip decomposition
- Scheduling strategies adjust strip size to minimize execution time
- Multi-user cluster
  - machines connected by 10 Mbit ethernet
  - available CPU on at least half of the machines is multi-modal with data changing between modes frequently
Adjusting Strip Size

- **Time balancing** used to determine strip size
- Set all $T(p,t)$ equal and solve for $NumElts(p,t')$

$$T(p,t) = RComp(p,t) + RComm(p,t + \Delta_1)$$
$$+ BComp(p,t + \Delta_2) + BComm(p,t + \Delta_3)$$
$$= A(p,t) \times NumElts(p) + B(p,t)$$

$$\sum_{p} NumElts(p) = n^2$$
Scheduling Strategies

• **Mean**
  – data assignments determined using *mean* (point-valued) application execution estimates

• **Conservative**
  – data adjusted so that machines with high-variance application execution estimates receive less work \((\mu + 2\sigma)\)
  – goal is to reduce penalty of being wrong
Preliminary Scheduling Results

- Conservative scheduling strategy misses big spikes, but is sometimes too conservative.
Research Directions

• **Quality of Information (QoIn)**
  – How can we develop useful mechanisms for obtaining and quantifying performance meta-information?
  – How do we combine different QoIn measures?
  – How can QoIn measures enhance scheduling?

• **Contingency Scheduling**
  – Can we develop schedules which adapt dynamically *during* execution?
More Research Directions

• **Performance-enhanced Tools**
  – *Netsolve* enhanced with *NWS* and *AppLeS* scheduling methodology

• **Performance contracts**
  – How should performance information be exchanged and brokered in grid systems?
  – How can we develop “grid-aware” programs?
Project Information

• Thanks to Dr. Darema and DARPA for support and very useful feedback.

• **Performance Prediction Engineering Home Page:**
  
  [http://www-cse.ucsd.edu/groups/hpcl/apples/PPE/index.html](http://www-cse.ucsd.edu/groups/hpcl/apples/PPE/index.html)

• **PPE team:** Jennifer Schopf, Neil Spring, Alan Su, Fran Berman, Rich Wolski
Up Next: Rich Wolski

Dynamic Forecasting for Performance Prediction Engineering with the Network Weather Service