Achieving Application Performance on the Computational Grid

Francine Berman
University of California, San Diego
The Computational Grid

- The Computational Grid
  - ensemble of heterogeneous, distributed resources
  - emerging platform for high-performance and resource-intensive computing

How do we write programs for the Grid?
Programming the Grid I

• Basics
  – Need way to login, authenticate in different domains, transfer files, coordinate execution, etc.
Programming the Grid II

• Performance-oriented programming

  – Need way to develop and execute performance-efficient programs

  – Program must achieve performance in an environment which is
    • heterogeneous
    • dynamic
    • shared by other users with competing resource demands

  This can be extremely challenging.

  – Adaptive application scheduling is a fundamental technique for achieving performance
• **Why scheduling?**
  - Experience with parallel and distributed codes shows that careful coordination of tasks and data required to achieve performance

• **Why application scheduling?**
  - No centralized scheduler which controls all Grid resources, applications are on their own
  - Resource and job schedulers prioritize utilization or throughput over application performance
- Why *adaptive* application scheduling?
  - Heterogeneity of resources and dynamic load variations cause performance characteristics of platform to vary over time and with load
  - To achieve performance, application must adapt to deliverable resource capacities
Adaptive Application Scheduling

**Fundamental components:**

- **Application-centric performance model**
  - Provides quantifiable measure of system components in terms of their potential impact on the application

- **Prediction of deliverable resource performance at execution time**

- **User’s performance criteria**
  - Execution time
  - Convergence
  - Turnaround time

**These components form the basis for AppLeS.**
What is AppLeS?

- **AppLeS = Application Level Scheduler**
  - Joint project with Rich Wolski (U. of Tenn.)

- **AppLeS is a methodology**
  - Project has investigated *adaptive application scheduling* using dynamic information, application-specific performance models, user preferences.
  - AppLeS approach based on real-world scheduling.

- **AppLeS is software**
  - Have developed multiple AppLeS-enabled applications and templates which demonstrate the importance and usefulness of adaptive scheduling on the Grid.
How Does AppLeS Work?

AppLeS + application
= self-scheduling application

Resources

NWS

Grid Infrastructure

- Schedule Deployment
  - "best” schedule
- Decision Model
  - evaluated schedules
- Schedule Planning and Performance Modeling
  - feasible resource sets
- Resource Selection
  - accessible resources
- Resource Discovery
Network Weather Service (Wolski, U. Tenn.)

- The NWS provides dynamic resource information for AppLeS

- NWS is stand-alone system

- NWS
  - monitors current system state
  - provides best forecast of resource load from multiple models
AppLeS Example: Simple SARA

- **SARA = Synthetic Aperture Radar Atlas**
  - application developed at JPL and SDSC

- **Goal: Assemble/process files for user’s desired image**
  - Radar organized into tracks
  - User selects track of interest and properties to be highlighted
  - Raw data is filtered and converted to an image format
  - Image displayed in web browser
Simple SARA

- AppLeS focuses on **resource selection problem**: Which site can deliver data the fastest?
- Code developed by Alan Su
Simple SARA

- Simple Performance Model

\[
    \text{FileTransferTime} = \frac{\text{DataSize}}{\text{AvailableBandwidth}}
\]

- Prediction of available bandwidth provided by Network Weather Service

- User’s goal is to optimize performance by minimizing file transfer time

- **Common assumptions:** (> = performs better)
  - vBNS > general internet
  - geographically close sites > geographically far sites
  - west coast sites > east coast sites
Experimental Setup

• Data for image accessed over **shared** networks
• Data sets 1.4 - 3 megabytes, representative of SARA file sizes
• Servers used for experiments
  
  via vBNS
  - `lolland.cc.gatech.edu`
  - `sitar.cs.uiuc`
  - `perigee.chpc.utah.edu`

via general internet
  
  - `mead2.uwashington.edu`
  - `spin.cacr.caltech.edu`
Preliminary Results

• Experiment with larger data set (3 Mbytes)
• During this time-frame, farther sites provide data faster than closer site
9/21/98 Experiments

- Clinton Grand Jury webcast commenced at trial 25
- At beginning of experiment, general internet provides data faster than vBNS
Supercomputing ’99

- From Portland SC’99 floor during experimental timeframe, UCSD and UTK generally “closer” than Oregon Graduate Institute (OGI) in Portland.
What if File Sizes are Larger?

**Storage Resource Broker (SRB)**

- SRB provides access to distributed, heterogeneous storage systems
  - UNIX, HPSS, DB2, Oracle, ..
  - *files can be 16MB or larger*
  - resources accessed via a common SRB interface
Predicting Large File Transfer Times

NWS and SRB present distinct behaviors

NWS probe is 64K, SRB file size is 16MB

Adaptive approach: Use adaptive linear regression on sliding window of NWS bandwidth measurements to track SRB behavior

SRB Performance model being developed by Marcio Faerman
Challenges for AppLeS

• **AppLeS-enabled applications perform well in multi-user environments**
  – Have developed AppLeS for
    • Stencil codes (Jacobi2D, magnetohydrodynamics, LU Decomposition …)
    • Distributed data codes (SARA, SRB, …)
    • Master/Slave codes (DOT, Ray Tracing, Mandelbrot, Tomography, …)
    • Parameter Sweep codes (MCell, INS2D, CompLib, …)

• **Methodology is right on target but …**
  – AppLeS must be integrated with application — labor-intensive and time-intensive
  – You generally can’t just take an AppLeS and plug in a new application
AppLeS Templates

- Current thrust is to develop **AppLeS templates** which
  - target structurally similar classes of applications
  - can be instantiated in a user-friendly timeframe
  - provide good application performance
Case Study: Parameter Sweep Template

- **Parameter Sweeps** = class of applications which are structured as multiple instances of an “experiment” with distinct parameter sets

- Independent experiments may share input files

- Examples:
  - MCell
  - INS2D
Example Parameter Sweep Application: MCell

- **MCell** = General simulator for cellular microphysiology
- Uses Monte Carlo diffusion and chemical reaction algorithm in 3D to simulate complex biochemical interactions of molecules
  - Molecular environment represented as 3D space in which trajectories of ligands against cell membranes tracked
- Researchers plan huge runs which will make it possible to model entire cells at molecular level.
  - Would like to perform execution-time computational steering, data analysis and visualization
PST AppLeS

- Template being developed by Henri Casanova and Graziano Obertelli

- **Resource Selection:**
  - For **small** parameter sweeps, can dynamically select a performance efficient number of target processors [Gary Shao]
  - For **large** parameter sweeps, can assume that all resources may be used
Scheduling Parameter Sweeps

• **Contingency Scheduling:** Allocation developed by dynamically generating a Gantt chart for scheduling unassigned tasks between scheduling events

• **Basic skeleton**
  1. Compute the next scheduling event
  2. Create a Gantt Chart $G$
  3. For each computation and file transfer currently underway, compute an estimate of its completion time and fill in the corresponding slots in $G$
  4. Select a subset $T$ of the tasks that have not started execution
  5. **Until each host has been assigned enough work,** heuristically assign tasks to hosts, filling in slots in $G$
  6. Implement schedule
Parameter Sweep Heuristics

• Currently studying scheduling heuristics useful for parameter sweeps in Grid environments

• HCW 2000 paper compares several heuristics
  – Min-Min [task/resource that can complete the earliest is assigned first]
  – Max-Min [longest of task/earliest resource times assigned first]
  – Sufferage [task that would “suffer” most if given a poor schedule assigned first, as computed by max - second max completion times]
  – Extended Sufferage [minimal completion times computed for task on each cluster, sufferage heuristic applied to these]
  – Workqueue [randomly chosen task assigned first]

• Criteria for evaluation:
  – How sensitive are heuristics to location of shared input files and cost of data transmission?
  – How sensitive are heuristics to inaccurate performance information?
Preliminary PST/MCell Results

- Comparison of the performance of scheduling heuristics when it is up to 40 times more expensive to send a shared file across the network than it is to compute a task.
- “Extended sufferage” scheduling heuristic takes advantage of file sharing to achieve good application performance.
Preliminary PST/MCell Results with “Quality of Information”

(a) one single scheduling event

(b) scheduling events every 500 sec

(c) scheduling events every 250 sec

(d) scheduling events every 125 sec
Work-in-Progress: Half-Baked AppLeS

- **Quality of Information**
  - Stochastic Scheduling
  - AppLePilot / GrADS

- **Resource Economies**
  - Bushel of AppLeS
  - UCSD Active Web

- **Application Flexibility**
  - Computational Steering
  - Co-allocation
  - Target-less computing
Quality of Information

• How can we deal with imperfect or imprecise predictive information?

• Quantitative measures of qualitative performance attributes can improve scheduling and execution
  – lifetime
  – cost
  – accuracy
  – penalty
Using Quality of Information

- **Stochastic Scheduling:** Information about the variability of the target resources can be used by scheduler to determine allocation
  - Resources with more performance variability assigned slightly less work
  - Preliminary experiments show that resulting schedule performs well and can be more predictable
Quality of Information and “AppLePilot”

- **AppLePilot** combines **AppLeS** adaptive scheduling methodology with fuzzy logic decision making mechanism from **Autopilot**
  - Provides a framework in which to negotiate Grid services and promote application performance
  - Collaboration with Reed, Aydt, Wolski

- Builds on the software being developed for **GrADS**
GrADS — Grid Application Development and Execution Environment

- Prototype system which facilitates end-to-end “grid-aware” program development
- Based on the idea of a performance economy in which negotiated contracts bind application to resources

- Joint project with large team of researchers
  - Ken Kennedy
  - Jack Dongarra
  - Dennis Gannon
  - Dan Reed
  - Lennart Johnsson
  - Andrew Chien
  - Rich Wolski
  - Ian Foster
  - Carl Kesselman
  - Fran Berman

Grid Application Development System
Summary

• Development of AppLeS methodology, applications, templates, and models provides a careful investigation of adaptivity for emerging Grid environments.

• Goal of current projects is to use real-world strategies to promote dynamic performance:
  – adaptive scheduling
  – qualitative and quantitative modeling
  – multi-agent environments
  – resource economies
• **Thanks** to NSF, NASA, NPACI, DARPA

• **AppLeS Home Page:**
  [http://apples.ucsd.edu](http://apples.ucsd.edu)

• **AppLeS Corps:**
  – *Fran Berman, UCSD*
  – *Rich Wolski, U. Tenn*
  – *Henri Casanova*
  – *Walfredo Cirne*
  – *Holly Dail*
  – *Marcio Faerman*
  – *Jim Hayes*
  – *Graziano Obertelli*
  – *Gary Shao*
  – *Otto Sievert*
  – *Shava Smallen*
  – *Alan Su*