Tools and Techniques for Measuring and Improving Grid Performance

Rupak Biswas
NASA Ames Research Center
Moffett Field, California, U.S.A.

rbiswas@nas.nasa.gov

Joint work with:
- M. Frumkin
- W. Smith
- R. Van der Wijngaart
- P. Wong

Overview

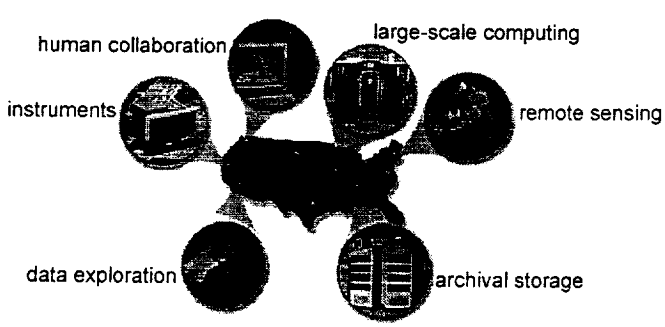
- Motivation and Objectives
- NASA's Information Power Grid
- Grid Benchmarking
- Grid Performance Monitoring
- User-Level Grid Scheduling
- System-Level Scheduling

Motivation and Objectives

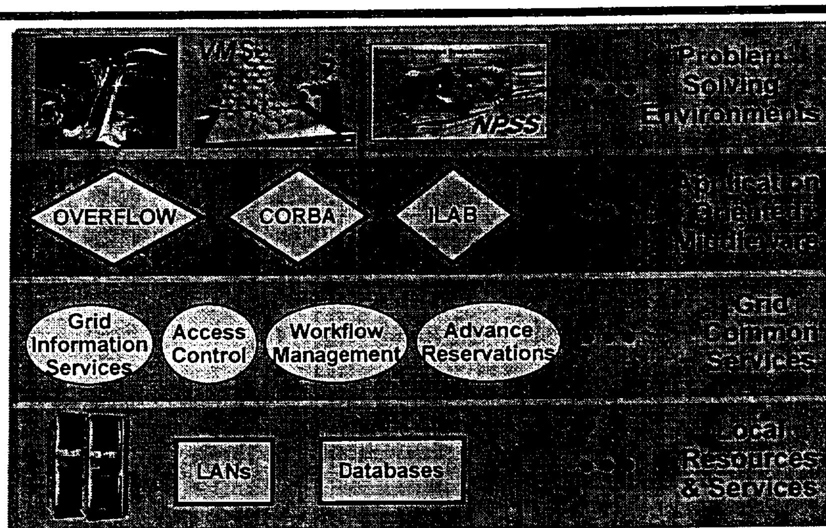
- Large-scale science and engineering accomplished through interaction of geographically-dispersed people, heterogeneous computing resources, information systems, and instruments
- Overall goal is to facilitate the routine interactions of these resources to reduce NASA mission-critical design cycle time
- Many facilities around the world are moving toward making resources available on a "Grid" (grid computing)
- The Information Power Grid (IPG) is NASA's push for a persistent, secure, and robust implementation of a Grid
- Investigate techniques and develop tools to measure and improve performance of a broad class of applications when run on a Grid

Information Power Grid

- Involves linking NASA's vast disperse resources to create an intelligent, scalable, adaptive, and transparent computational, communication, data analysis, and storage environment



IPG Layered Architecture



APART-2001

5

Grid Benchmarking

- Deficiencies of current Grid performance measurement technology
  - Simulation tools idealized, unclear Grid model assumptions, static (WARMstones, Bricks, MicroGrid)
  - Superposition principle of probes may not hold (Globus HBM, NWS, NetLogger)
- Existing techniques useful for
  - Users debugging Grid application performance
  - Developers of Grid and communication software
- But does not provide metric for comparing Grid performance on actual distributed applications
- Goal:
  - Determine Grid functionality and application performance objectively
  - Use representative set of distributed applications

APART-2001

6

Ames Research Center

Grid Benchmark Requirements

- Tests computational aspects of environment
- Is representative of scientific computing tasks
- Uses basic Grid services
- Is not intrusive (no throughput stress testing)
- Contains communicating processes
- Does significant communication
- Is verifiable (deterministic, not interactively steered)
- Needs no initialization data files
- Is fair

Ames Research Center

NAS Grid Benchmarks (NGB)

- Provide paper-and-pencil specifications of small set of complete but representative distributed applications
- For convenience, also provide reference implementations (Globus, Legion, Condor, Java, ksh)
- Focus on computational aspects of Grids
  - Use mesh-based NAS Parallel Benchmarks (NPB) as building blocks (well understood, calibrated, deterministic, portable, allow communication, parallel, no input required but output of one can be input for another)
    - MG (multigrid for Poisson eqn): post-processing (data smoother)
    - FT (spectral method for Laplace eqn): visualization (spectral analysis)
    - BT (ADI, block tridiagonal):
    - SP (ADI, scalar pentadiagonal):
    - LU (lower-upper sym Gauss-Seidel):

    } Scientific computations (flow solvers)

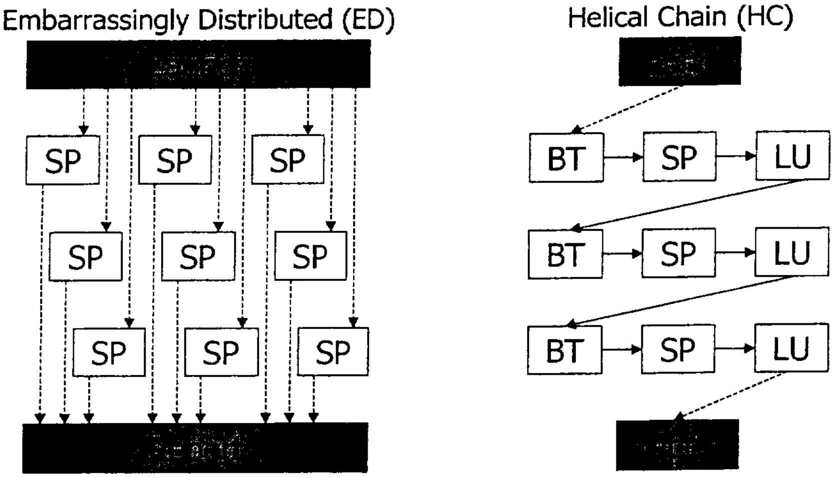
NGB Construction

- Construct synthetic Grid applications for scientific computing
- Data Flow Graph coupling NPB codes
- Provide wide range of problem sizes (classes): S, A, B, C, ...
- Benchmarks non-converging, but numerically stable
- Limit number of verification values
- Specify abstract services: authenticate, create task, communicate
- Do not specify mapping, scheduling, fault tolerance, data security
- Report turnaround time and the resources used

APART-2001 9

NGB Data Flow Graphs (Class S)

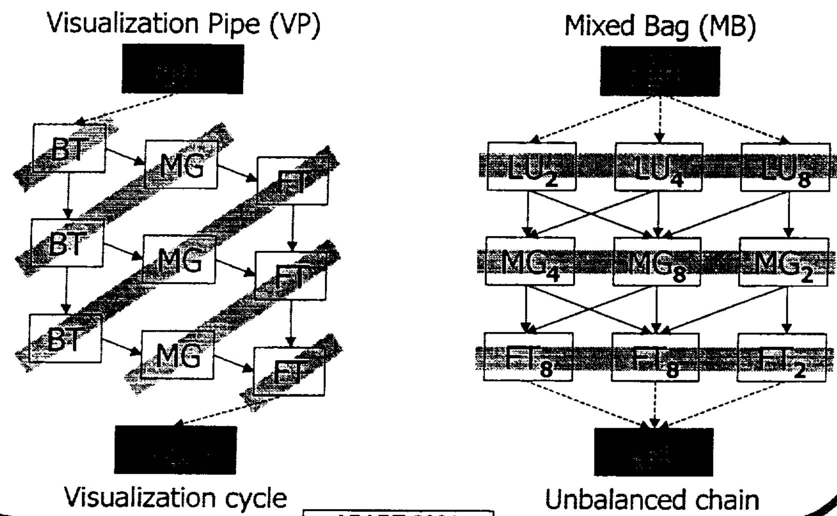
Embarrassingly Distributed (ED) — Helical Chain (HC)



Parameter study — Cyclic process (restart)

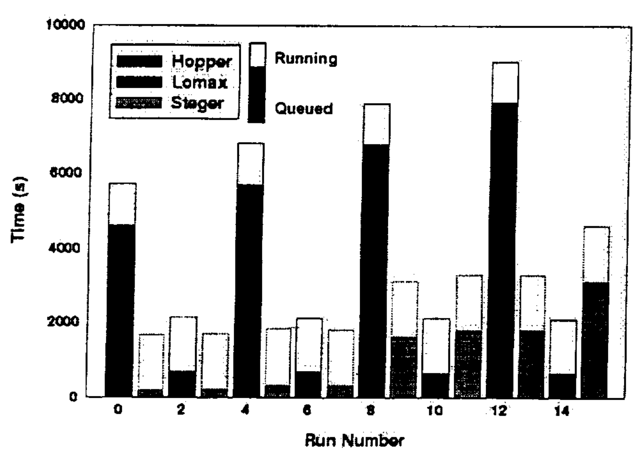
APART-2001 10

NGB Data Flow Graphs (Class S)



Preliminary Results

- ED.A (16 SP.A NPBs) run on three Origin systems under Globus



NGB Issues

- Are proposed Data Flow Graphs representative of scientific apps?
- What other classes of apps should be used?
- Is turnaround time the best measure?
- Do we need to consider a Grid currency (G$)?
- How to interpret the results?
  - Primitive Grid services (functionality, consistency among runs)
  - Reservation of resources (variation of single resource)

APART-2001

13

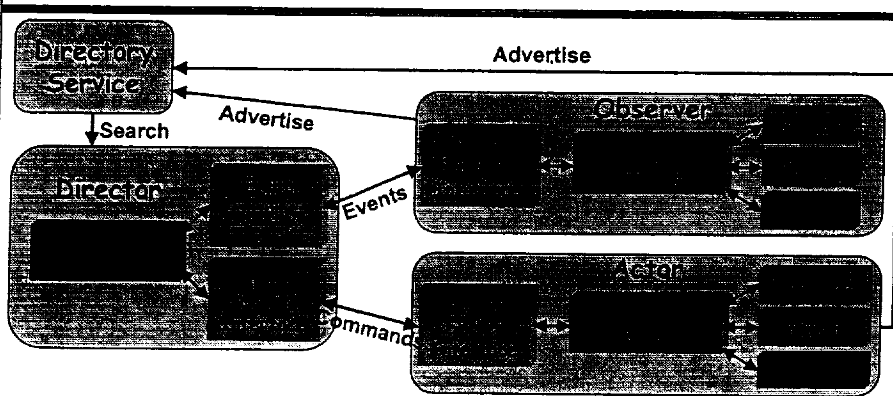
Grid Performance Monitoring

- IPG a large distributed set of resources, services, and applications
  - Will be failures; needs to be monitored
  - Must be managed
- Develop general framework for observation and control
  - Observe and control variety of resources, services, and applications
  - Scalable, secure, and compatible with emerging GGF standards
  - Extensible to observe new events, perform new actions, and manage
- Deficiencies of existing monitors
  - Cannot be embedded in tools or apps (AIMS, Big Brother)
  - Limited fault detection functionality (Globus HBM, NWS)
  - System- or app-specific information, but not both (SNMP-based tools, MPICH profiling)
  - Lack of extensible data forwarding and gathering mechanisms (Netlogger)
  - Incompatibility with IPG security and authentication requirements

APART-2001

14

CODE: Control and Observation of Distributed Environments



- Directory Service contain info about Observers & Actors for Director
- Sensor Manager manages sensors, subscriptions, queries
- Actuator Manager handles requests for actions
- Expert System + User Rules instead of Management Logic in Director

APART-2001

15

CODE Implementation

- In C++ to be modular and extensible
- Uses pthreads
- Communicates using TCP, UDP, or SSL
- OpenSSL for authentication and security
- XML encoding of messages
- Data in Directory Service compatible with LDAP schemas
- CLIPS expert system available as alternative in Director
- Initially targeting IRIX, Solaris, Linux
- Ported Director code to Java for GUI

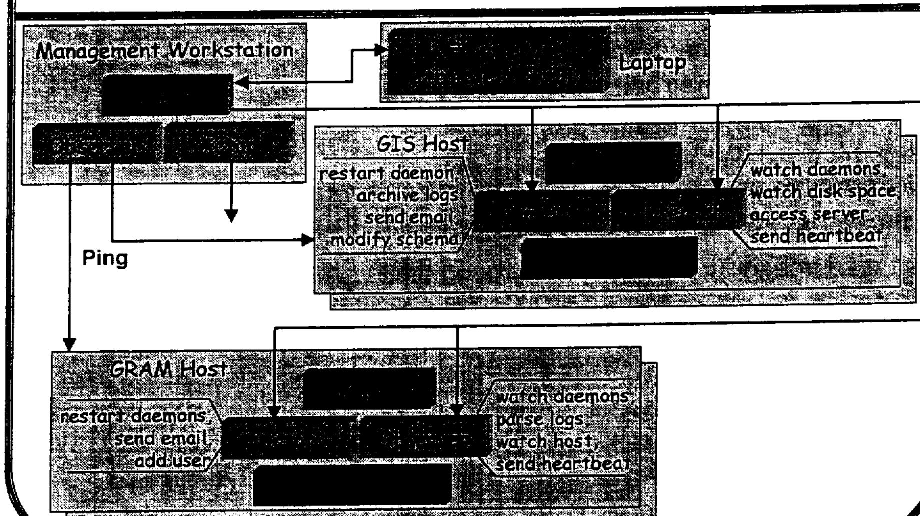
APART-2001

16

Grid Management System Using CODE

- Observe and control a Globus-based computational Grid like IPG
  - Becomes difficult as Grids get larger
- Things to observe
  - Globus Resource Allocation Manager (GRAM) reporter daemons
  - Grid Information Service (GIS) servers
  - Log files
  - Resource status and usage
- Things to control
  - Restarting GRAM daemons
  - Restarting / configuring GIS servers
  - Add / remove user mapping
  - Send appropriate e-mail
- Provide a GUI

Grid Control System Using CODE



User-Level Grid Scheduling

- Grids have lots of different computers
- Where should a user submit his application?
  - Which machines can user access?
  - Which machines have sufficient resources?
  - How much do machines cost to use?
  - When will the application finish?
    - Time to pre-stage input files
    - Time waiting in scheduler queue
    - Time to execute
    - Time to post-stage output files
- Currently ignore time to stage files

APART-2001

19

Approach

- Develop execution time prediction technique
  - Instance-based learning using historical information
- Develop queue wait time prediction technique
  - Simulate scheduling algorithms
  - Use execution time predictions
- Add the two predicted times to obtain application turnaround time
- Select resources with minimum turnaround time

APART-2001

20

Instance-Based Learning

- Aka: locally-weighted learning, memory-based learning, lazy learning
- Maintain a database of experiences
  - Each experience has set of input and output features
- Calculate an estimate for a query using relevant experiences
  - Relevance measured with a distance function
  - Calculation can be an average, distance weighted average, locally weighted regression
  - Use only nearest experiences (nearest neighbors) or all experiences
- Local learning: not one equation that fits all data points
- No learning phase as in neural networks

Distance Functions

- Minkowski $D(x,y)=\left(\sum_f |x_f - y_f|^r\right)^{1/r}$
  - Manhattan $D(x,y)=\sum_f |x_f - y_f|$
  - Euclidean $D(x,y)=\sqrt{\sum_f (x_f - y_f)^2}$
  - Only works where features are linear
- Heterogeneous Euclidean Overlap metric
  - Handles features that are linear or nominal

$$d_f(x,y)=\begin{cases}1, & \text{if } x_f \text{ or } y_f \text{ is unknown,}\\ overlap_f(x,y), & \text{if } f \text{ is nominal,}\\ rn\_diff_f(x,y), & \text{otherwise}\end{cases} \qquad overlap_f(x,y)=\begin{cases}0, \text{if } x_f = y_f\\ 1, \text{otherwise}\end{cases}$$

$$D(x,y)=\sqrt{\sum_f d_f(x,y)^2} \qquad rn\_diff_f(x,y)=\frac{|x_f - y_f|}{\max_f - \min_f}$$

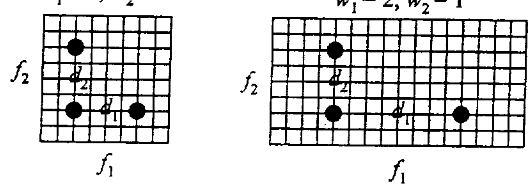
Feature Scaling

- Warp input space by scaling features in distance function

$$D(x,y) = \sqrt{\sum_f w_f d_f(x,y)^2}$$

- Larger weight implies more relevant feature

$w_1 = 1, w_2 = 1$ $\quad\quad$ $w_1 = 2, w_2 = 1$



$d_1 = 4, d_2 = 4$ $\quad\quad$ $d_1 = 4, d_2 = 8$

Kernel Regression

- Estimate is distance weighted average of experiences
- Weighting also called kernel function

$$E_f(q) = \frac{\sum_e K(D(q,e))V_f(e)}{\sum_e K(D(q,e))}$$

- Want weight->C as d->0, and weight->0 as d->∞
- Gaussian an example of kernel function: $K(d) = e^{-d^2}$
- Kernel width k to scale distances: $K(d) = e^{-(d/k)^2}$
- Can also incorporate nearest neighbors

Parameter Selection

- What configuration to use for prediction?
  - Number of nearest neighbors
  - Feature weights
  - Kernel width
- Search techniques to find the best
  - Genetic algorithms
  - Simulated annealing
  - Hill climbing
  - Evaluate configuration using trace data
- Currently, genetic algorithms show best performance

APART-2001

25

Execution Prediction Performance

- Use IBL techniques on experience base of 2000 entries
- Predict application runtime & compare against user estimate
- Genetic algorithm search for configuration over a month's data from steger
- Evaluate using 6 months of data
- Average error of prediction technique 4.6X less than user estimate

Machine	IBL Prediction		User Estimate		Mean Runtime (mins)
	Mean Error (mins)	% of Mean Runtime	Mean Error (mins)	% of Mean Runtime	
Steger	30.31	32.81	78.00	84.43	92.38
Hopper	16.95	44.37	103.36	270.58	38.20
Lomax	23.00	46.06	126.25	252.85	49.93

APART-2001

26

Queue Wait Time Predictions

- Predict how long an application will wait in a scheduling queue before starting execution
- Perform a scheduling simulation
  - Simulate scheduling of all waiting and running applications
  - Use execution time predictions in simulation
  - Developed event-driven simulator
  - Implemented a NAS PBS simulator
- Validated NAS PBS simulator
  - For 6 months of data, 64% matched actual start times of ~20K jobs
  - Some mismatches due to dedicated time and machine crashes
- No systematic analysis of prediction accuracy yet

APART-2001

27

User-Level Scheduling

- Each user has their own grid scheduler
  - No bottleneck or single point of failure
- Many potential goals for user-level schedulers
  - Minimize turnaround time of individual applications, parameter study, DAG of applications
  - Minimize cost
- Minimize turnaround time of individual applications
  - User or scheduler identifies potential resources
    - Cannot consider all grid resources for every application
  - Scheduler selects from potential set of resources using minimum predicted turnaround time
  - Scheduler sends application to selected resource
  - Scheduler monitors application progress and periodically checks if application should be moved to different resources

APART-2001

28

Implementation at NAS

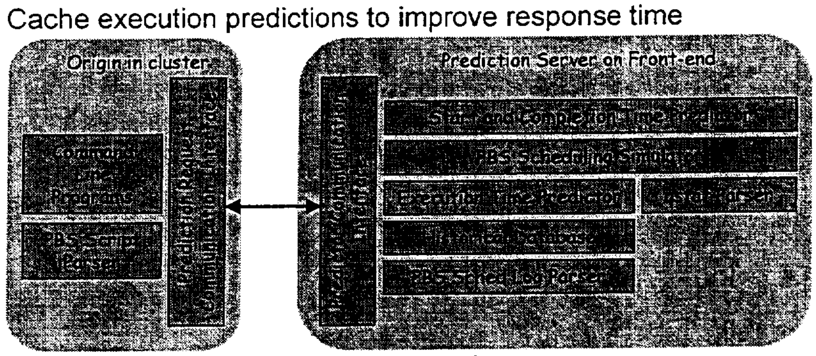
- Predict for three SGI Origins from NAS workstations
- Command line programs for predictions of execution times, start times, and completion times when given PBS script or PBS job ID
- Command line program to suggest which Origin to use
- Experience base for each Origin
- Use NAS Parallel Benchmarks to compute scaling factors between machines
- Predict for machine using it's experience base, or a scaled prediction from other experience bases, depending on confidence
- Cache execution predictions to improve response time

APART-2001

29

Execution Prediction Implementation

- Predict for Steger, Hopper, and Lomax from any machine in cluster
- Separate experience base for each machine
- Use NPBs to compute scaling factors between machines
- Cache execution predictions to improve response time



APART-2001

30