Job Scheduling and Management

Colin Perkins

http://csperkins.org/teaching/2004-2005/gc5/

Lecture Outline

- Job Scheduling and Management Concepts
  - Resource Discovery
  - Job Allocation
  - Job and Data Distribution
  - Job Scheduling
  - Middleware
- Implementations
  - Condor, OpenPBS, Sun Grid Engine, Xgrid
  - GRAM, Condor-G
- Tutorial

Copyright © 2004 University of Glasgow

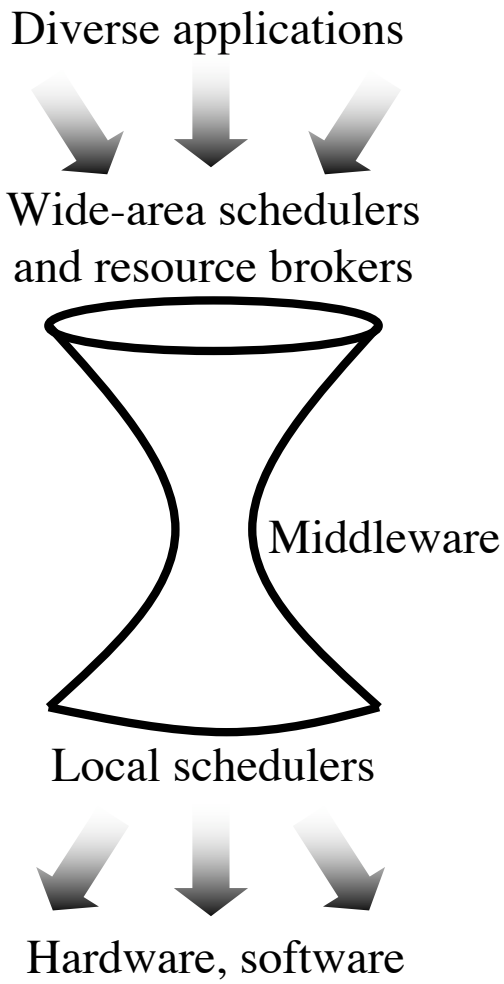
Job Scheduling and Management in a Grid

- Grid computing applications follow two popular patterns:
  - Distributed Computing
    - Exploring a large parameter space, repeating a task across a data set
      - Manageable amounts of data, enormous need for computational cycles
      - Master-worker model
      - Embarrassingly parallel applications
    - SETI@home, particle physics, bioinformatics, movie rendering
  - Remote Resource Access
    - Coordinating execution of small number of tasks, running on distributed resources managed by diverse organizations
    - Remote instrument access (scanning electron microscopes, sensors, etc.), combining large database queries
- Need grid-aware scheduling algorithms to coordinate execution of jobs across multiple sites

Copyright © 2004 University of Glasgow

The Job Scheduling Problem

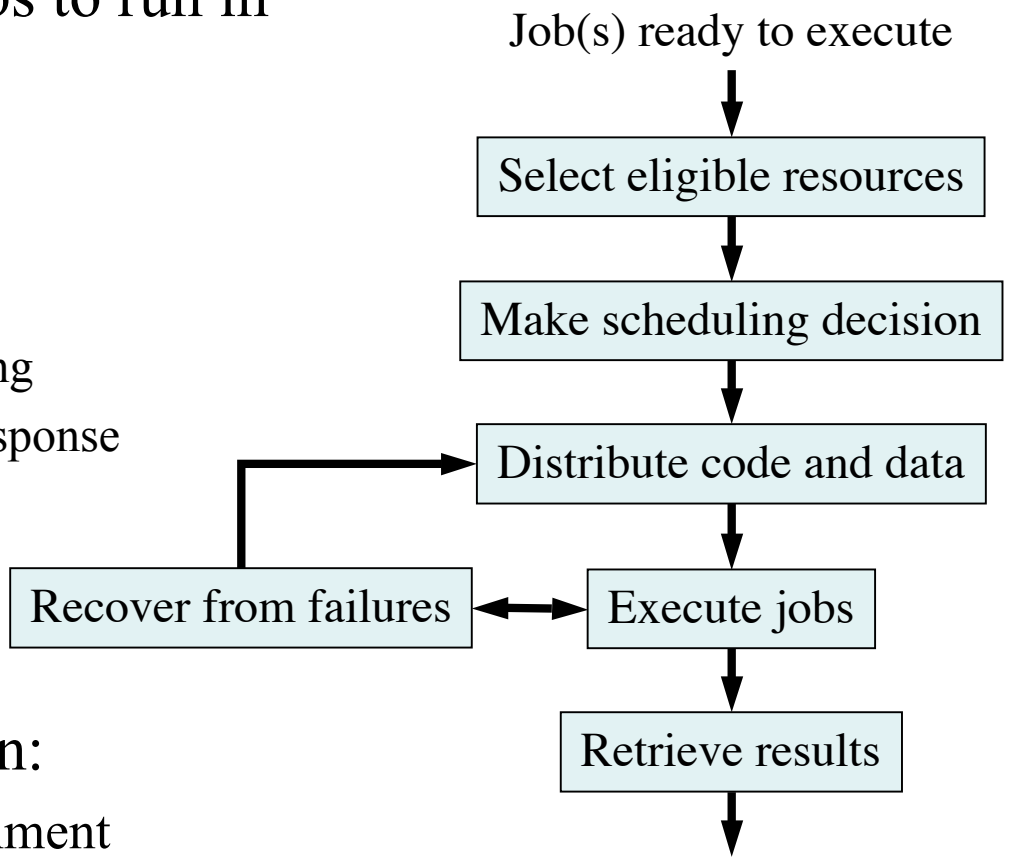
- Job scheduling and management in a Grid is a wide-area scheduling and resource management problem
  - *Discover* remote resources
  - *Allocate* jobs to resources
  - *Distribute* the code and data
  - *Schedule* jobs to run
  - Collect and collate results
- Desirable to hide details of the local infrastructure at each site from the wide-area scheduler
  - Decouple implementations
- Use an hourglass model with *middleware* to isolate the wide-area scheduler from the local schedulers



Copyright © 2004 University of Glasgow

Local Scheduling

- How to schedule multiple jobs to run in parallel across a set of hosts?
  - Where to run jobs:
    - Need for specific resources
    - System load
    - Batch *vs.* interactive scheduling
    - Real time *vs.* non-real time response
  - Co-allocating related jobs
  - How to distribute jobs
  - Failure tolerance
- Occurs within an organization:
  - Can control the system environment
  - Can make informed decisions
    - Full knowledge of the system
  - But often loosely coupled systems; not full control

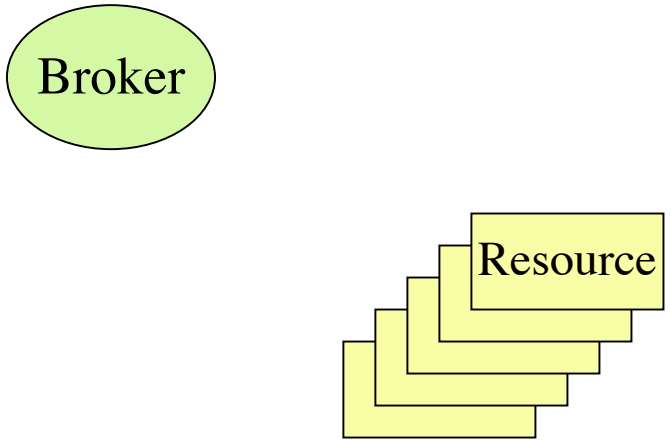


Copyright © 2004 University of Glasgow

Resource Selection

- Resources (hosts) vary in capacity, configuration, policy
- Jobs have difference requirements
- When scheduling a job, need global knowledge to make an informed choice

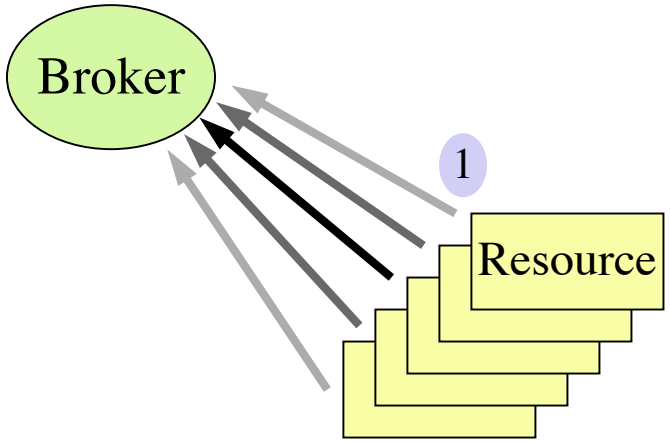
⇒ Resource broker architecture



Copyright © 2004 University of Glasgow

Resource Selection

- Resources advertise status and configuration to broker
  - Is broker centralised or distributed?
  - Do hosts push status information to the broker? Or does it poll periodically?
  - How to scale to many resources?
  - How to specify resource status? Simple statistics vs. complex specification languages?

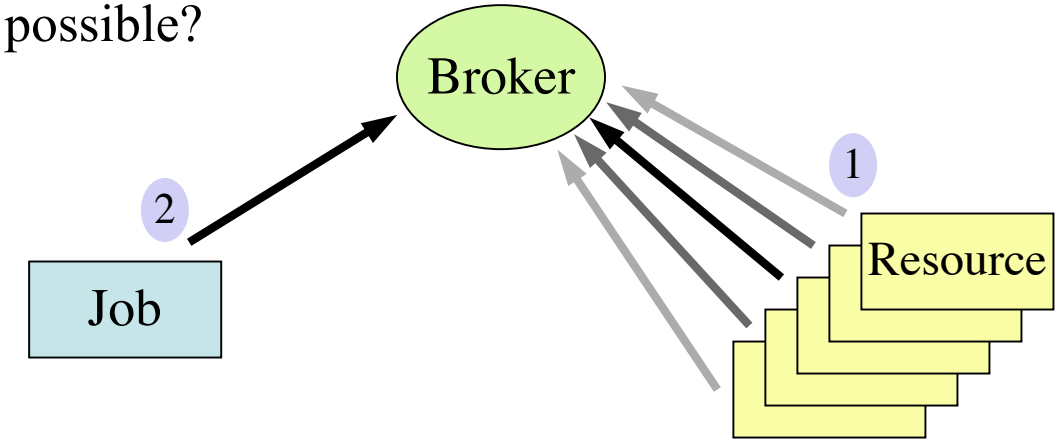


1 Resources advertise status

Copyright © 2004 University of Glasgow

Resource Selection

- Job initiators solicit resources from broker
  - How to contact the broker? What protocol?
  - How to choose between multiple offers from distributed brokers?
  - How to specify resource requirements? Domain specific language?
    - Fixed (complete?) schema and matching rules
    - Variable attributes and matching function
  - How to specify dependencies on other jobs?
  - Are advance reservations possible?



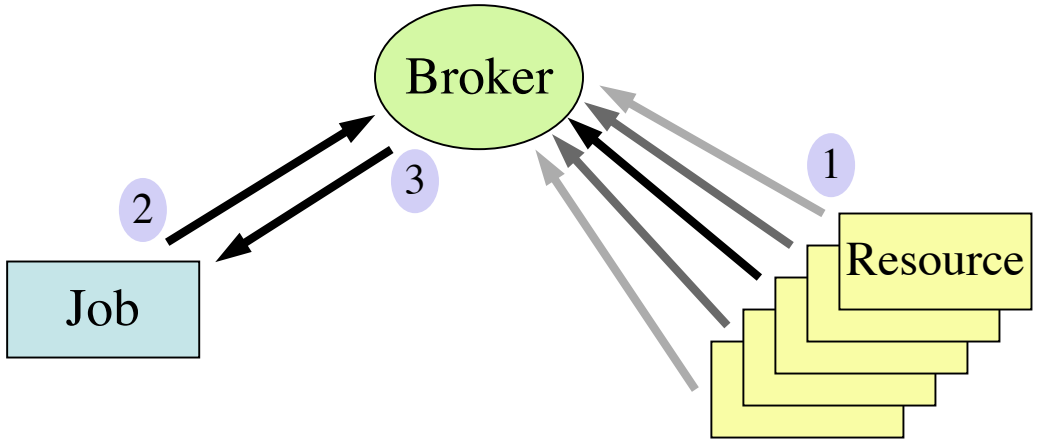
1 Resources advertise status

2 Solicit resources for job

Copyright © 2004 University of Glasgow

The Scheduling Decision

- Broker matches jobs to resources
  - Based on most recent knowledge of resource status
- Set of eligible resources constrained by job requirements
  - Hardware, software, configuration, policy
  - Timing constraints for real-time jobs

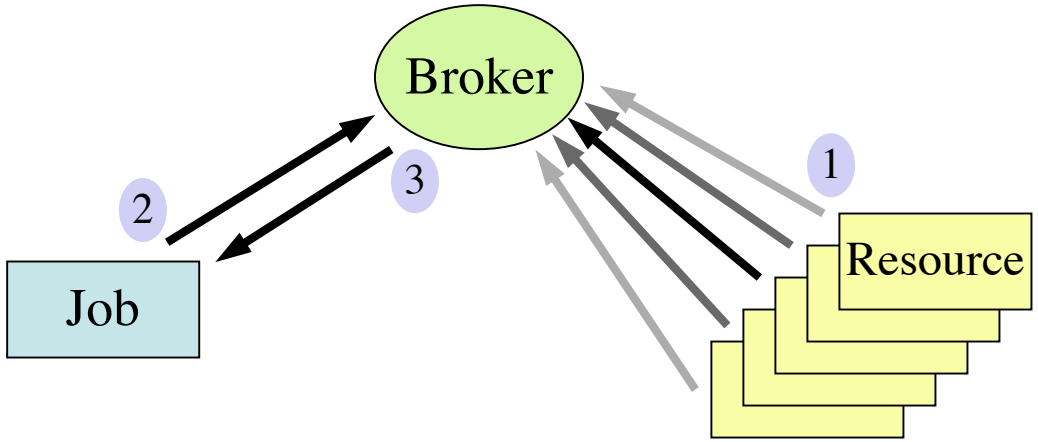


1. Resources advertise status
2. Solicit resources for job
3. Broker matches job with eligible resource

Copyright © 2004 University of Glasgow

The Scheduling Decision

- Choice of resource from those eligible is driven by policy goals
  - Minimize load on a particular resource
  - Minimize execution time
  - Minimize communication
  - Balance load across resources
  - Prefer to use certain resources
  - Co-allocation of dependent jobs
  - Etc.



1 Resources advertise status

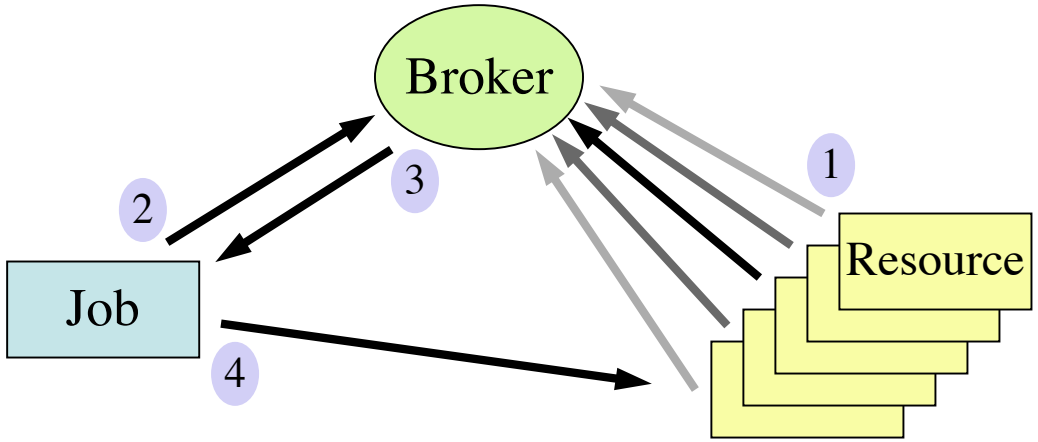
2 Solicit resources for job

3 Broker matches job with eligible resource

Copyright © 2004 University of Glasgow

Distributing Jobs

- Jobs need a predictable execution environment
  - Hardware differences: virtual machine languages like Java, C#, etc.
  - Surprisingly hard to make a binary that'll execute across different versions of an operating system; need standard environment
  - How to ensure code and data integrity and confidentiality?
    - Don't necessarily trust the host on which a job is running…
    - Trusted computing (a.k.a. Palladium) *very* helpful here!



1. Resources advertise status
2. Solicit resources for job
3. Broker matches job with eligible resource
4. Job assigned to resource

Copyright © 2004 University of Glasgow

Distributing Jobs and Data

- Jobs need a predictable execution environment
  - Hardware differences: virtual machine languages like Java, C#, etc.
  - Surprisingly hard to make a binary that'll execute across different versions of an operating system; need standard environment
  - How to ensure code and data integrity and confidentiality?
    - Don't necessarily trust the host on which a job is running…
    - Trusted computing (a.k.a. Palladium) *very* helpful here!

- Jobs need to access data and store results
  - Cannot assume a shared file system
  - Does the scheduler automatically distribute input files and collect results? Is this done manually as part of the job?
  - Does the job have access to a file system? All or part of it?
    - How does constraining access to the file system affect ability to load shared libraries? (e.g. the `jail` facility on FreeBSD)

Copyright © 2004 University of Glasgow

Executing Jobs

- Hosts employ a range of scheduling algorithms
  - Batch, interactive, real-time
- Hosts need resource reservation, for co-allocation
- Hosts may impose policy considerations
  - E.g. only execute jobs when otherwise idle
- Hosts may require authentication, authorisation and accounting

- Execution environment may not trust the job
  - Signed code
  - Sandboxes and virtual machines
    - Complete virtual machine architectures (e.g. Java)
    - Partial virtualisation (e.g. virtual system calls in Condor)

Copyright © 2004 University of Glasgow

Robustness and Fault Tolerance

- Jobs may fail due to programmer error or environmental issues
- Need to ensure consistent results in presence of failure
  - Graceful failure of a system of dependent jobs
  - Checkpoint and retry of jobs
    - May require modification of applications…
  - Atomic operations
  - Two-phase commit protocols to ensure agreement
    1. The commit manager assembles votes on result: *commit* or *abort*
       Hosts that vote *commit* guarantee they can complete action at later date
    2. The commit manager decides on basis of vote, and propagates *commit* or *abort* to other nodes
  - Exception handling

- Critical to understand failure and recovery modes

Copyright © 2004 University of Glasgow

Policy Issues

- User policy issues:
  - When to allow use of your workstation?
  - How much to trust remote jobs?
  - How much state of your host is visible to remote jobs?
  - How much control do you have over jobs running on your host?

- Site policy issues:
  - When to allow use of resources?
  - How much to trust remote jobs?
  - Are users allowed to see jobs executing on their host?
  - How much of site state to expose to the outside world?

Axioms: distrust and privacy…

Copyright © 2004 University of Glasgow

Local Scheduling Examples

- Condor
- OpenPBS
- Sun Grid Engine
- Xgrid

Copyright © 2004 University of Glasgow

Local Scheduling: Condor

- http://www.cs.wisc.edu/condor/
- Network batch queuing system for clusters and cycle-stealing
  - Jobs have dispatch priority; facilities for ordered jobs and master-worker operation
  - Scheduling performed using standard workstation scheduler
- Automatically distributes code, data and retrieves results
- Uses *ClassAds* to match jobs to resources
  - Schema-free requirements specification language
- Robustness via checkpoint and recovery
  - Requires re-linking against a modified `libc`
  - Generally very robust in practice
- Limited security and sandboxing; trusted environment

Copyright © 2004 University of Glasgow

Condor
High Throughput Computing

Local Scheduling: Other Systems

- OpenPBS http://www.openpbs.org/
- Sun Grid Engine http://gridengine.sunsource.net/
- Xgrid http://www.apple.com/acg/xgrid/

More limited systems… generally perform job scheduling; leave job management, data distribution, fault tolerance to the user

---

In general:

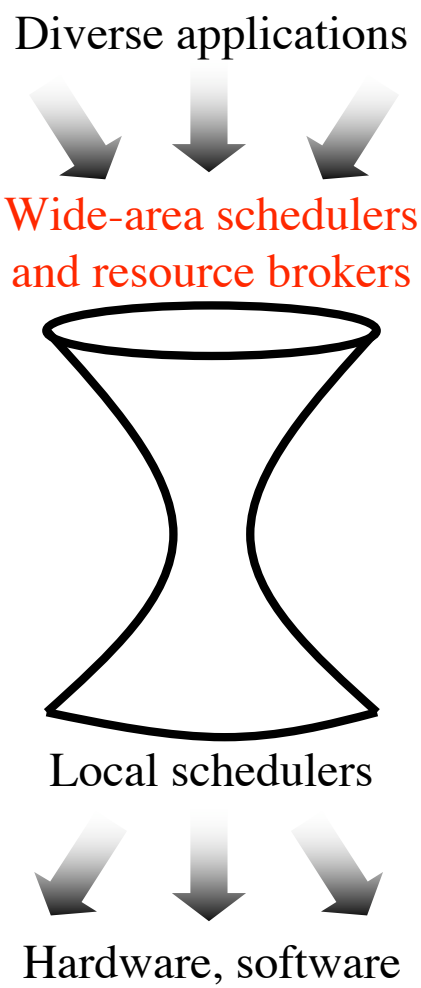
- Security, authentication, authorisation and accounting neglected
- No real facilities for real-time jobs, resource reservation
- Limited robustness and distributed coordination facilities

Copyright © 2004 University of Glasgow

Grid Scheduling

- Planning execution of jobs on particular sites
  - Policy choice based on advertised capabilities of sites, contracts
  - Not concerned with the management of resources within a particular site
  - An issue of planning *vs.* scheduling

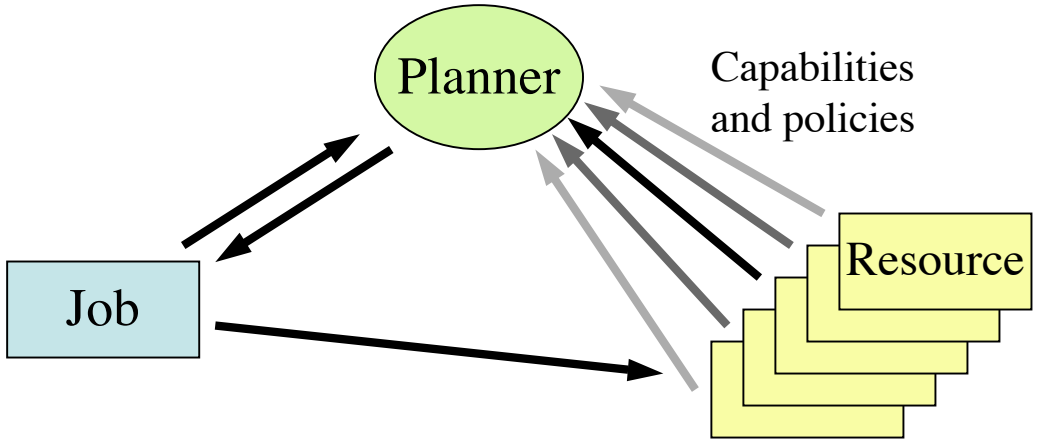
- Sites wish to retain their autonomy and hide details of their operation
  - Grid scheduler operates with partial information, abstract system state



Copyright © 2004 University of Glasgow

Grid Scheduling

- Similar broker-based model to local scheduling
- But operates at a more abstract level
  - Sites advertise capabilities and policies
  - Brokers assign jobs to sites based on those capabilities and policies
  - Brokers and job submitters plan execution, perhaps use contracts to reserve resources
  - Driven by economics and pricing models



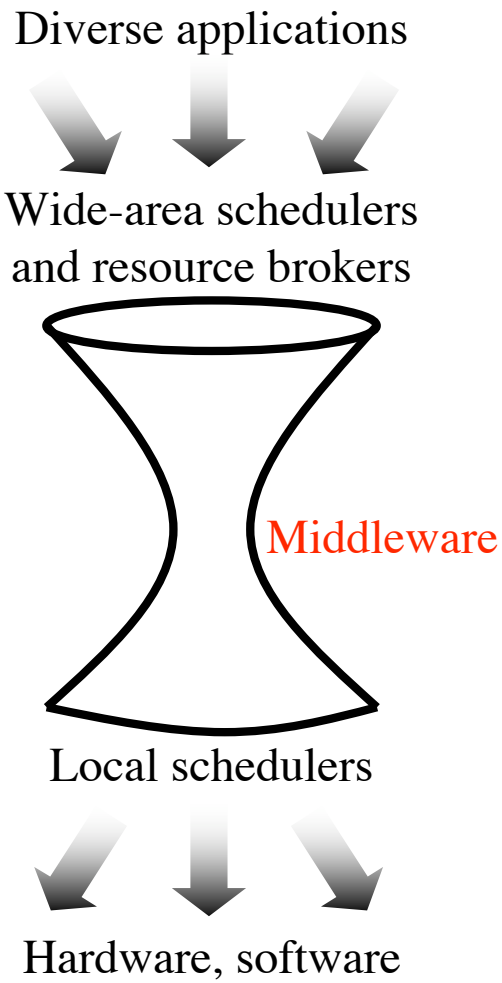
Copyright © 2004 University of Glasgow

The Need For Middleware

- Want to support:
  - Site autonomy
  - Heterogeneous substrates
  - Varied and extensible site policies
  - Co-allocation of resources across sites
  - Online control of jobs

… with a uniform interface that decouples grid applications from implementation details at a particular site

⇒ Middleware to hide the differences and expose common functionality



Copyright © 2004 University of Glasgow

Middleware: GRAM

"The Grid Resource Allocation and Management (GRAM) service provides a single interface for requesting and using remote system resources for the execution of jobs. The most common use of GRAM is remote job submission and control. It is designed to provide a uniform, flexible interface to job scheduling systems." [http://www-unix.globus.org/toolkit/docs/3.2/gram/]

- Abstracts local scheduling functions as middleware, to allow uniform wide-area scheduling
  - Exposes a limited subset of local scheduling functionality, primarily job submission and management
  - Resource brokers are not part of GRAM; must be built above it

Copyright © 2004 University of Glasgow

Middleware: Condor-G

- Condor-G is a version of Condor that uses GRAM to assign jobs to run queues
- Same interface as Condor, but schedules jobs using GRAM rather than native Condor protocols
  - Use Globus for authentication, remote program execution and data transfer
- Somewhat reduced functionality
  - No matchmaking
  - No automatic file transfer, staging
  - No check pointing for fault tolerance

Copyright © 2004 University of Glasgow

Grid Scheduling: Fundamental Issues

- Effects of communication latency
  - Embarrassingly parallel jobs in the master-worker model are effective
    - Rendering
    - Exhaustive search of parameter space
    - Analysis of large data space
  - Communication and coordination are *very* expensive in widely distributed systems; significant performance bottleneck for some applications
- Co-scheduling jobs across organizational boundaries
  - No central authority, so becomes a distributed coordination and resource reservation problem
  - *Very* difficult to guarantee if resources are scarce
- Effects of failure
  - Likelihood of increases with scale
  - Understand failure and recovery modes

Copyright © 2004 University of Glasgow

Tutorial and References

- Tutorial on Thursday: discussion of job scheduling papers
  - D. Thain, T. Tannenbaum & M. Livny, “Distributed Computing in Practice: The Condor Experience”, to appear in *Concurrency and Computation: Practice and Experience*, 2004.
  - K. Czajkowski, I. Foster, N. Karonis, C. Kesselman, S. Martin, W. Smith and S. Tuecke, “A Resource Management Architecture for Metacomputing Systems”, Proceedings of the IPPS/SPDP '98 Workshop on Job Scheduling Strategies for Parallel Processing, Orlando, FL, USA, March 1998.
  - A. Andrieux, D. Berry, J. Garibaldi, S. Jarvis, J. MacLaren, D. Ouelhadj and D. Snelling, “Open Issues in Grid Scheduling”, Proceedings of the workshop held at the e-Science Institute, Edinburgh, October 2003.

Copyright © 2004 University of Glasgow