1. Computing Resources
   1.1. Parallel and Distributed Computing
   1.2. Types of Computing Platforms
   1.3. Local Resource Management Systems

2. Grid Middleware
   2.1. Integration of Different Administrative Domains
   2.2. The Globus Toolkit
   2.3. The GridWay Meta-scheduler

3. A Taxonomy for Grid Scheduling Architectures
   3.1. The Taxonomy
   3.2. Multiple Administrative Domains
   3.3. Multiple Grid Infrastructures
   3.4. From the Cluster to the Grid
1. Computing Resources

1.1. Parallel and Distributed Computing

Goal of Parallel and Distributed Computing

- **Efficient** execution of computational or data-intensive applications

Types of Computing Environments

**High Performance Computing (HPC) Environments**

- Reduce the execution time of a single distributed or shared memory parallel application (MPI, PVM, HPF, OpenMP…)
- Performance measured in floating point operations per second
- Sample areas: CFD, climate modeling…

**High Throughput Computing (HTC) Environments**

- Improve the number of executions per unit time
- Performance measured in number of jobs per second
- Sample areas: HEP, Bioinformatics, Financial models…
1. Computing Resources

1.2. Types of Computing Platforms

- **Centralized**
  - Coupled
  - Network Links
  - Administration
  - Homogeneity

- **Decentralized**
  - Decoupled

- **SMP (Symmetric Multi-processors)**
- **MPP (Massive Parallel Processors)**
- **Clusters**
- **Network Systems**
  - Intranet/Internet

---

**High Performance Computing**

**High Throughput Computing**
1.3. Local Resource Management Systems

Management of Computing Platforms

- Computing platforms are managed by Local Resource Management (LRM) Systems
  1. Batch queuing systems for HPC servers
  2. Resource management systems for dedicated clusters
  3. Workload management systems for network systems
- There aim is to maximize the system performance

<table>
<thead>
<tr>
<th>Independent Suppliers</th>
<th>Open Source</th>
<th>OEM Proprietary</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 Platform Computing</td>
<td>2 Altair</td>
<td>1 IBM Load Leveler</td>
</tr>
<tr>
<td>3 LSF</td>
<td>3 Open PBS</td>
<td></td>
</tr>
<tr>
<td>2 Altair</td>
<td>3 University of Wisconsin Condor</td>
<td></td>
</tr>
<tr>
<td>3 PBS Pro</td>
<td>1 Cray NQE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2 Sun Microsystems SGE</td>
<td></td>
</tr>
</tbody>
</table>
1. Computing Resources

1.3. Local Resource Management Systems

LRM Systems Limitations

- Do not provide a common interface or security framework
- Based on proprietary protocols
- **Non-interoperable computing vertical silos** within a single organization
  - Requires specialized administration skills
  - Increases operational costs
  - Generates over-provisioning and global load unbalance

![Diagram showing fragmented infrastructure and non-interoperable systems]

Only a small fraction of the infrastructure is available to the user.

Infrastructure is fragmented in non-interoperable computational silos.
1. Computing Resources
   1.1. Parallel and Distributed Computing
   1.2. Types of Computing Platforms
   1.3. Local Resource Management Systems

2. Grid Middleware
   2.1. Integration of Different Administrative Domains
   2.2. The Globus Toolkit
   2.3. The GridWay Meta-scheduler

3. A Taxonomy for Grid Scheduling Architectures
   3.1. The Taxonomy
   3.2. Multiple Administrative Domains
   3.3. Multiple Grid Infrastructures
   3.4. From the Cluster to the Grid
"Any problem in computer science can be solved with another layer of indirection… But that usually will create another problem." David Wheeler

A New Abstraction Level

“A (computational) grid offers a common layer to integrate heterogeneous computational platforms (vertical silos) and/or administrative domains by defining a consistent set of abstraction and interfaces for access to, and management of, shared resources”

**Grid Middleware**

- Local Resource Manager 1
- Local Resource Manager 2
- Local Resource Manager 3

**Types of Resources:** Computational, storage and network.

**Common Interface for Each Type of Resources:** User can access a wide set of resources.
2. Grid Middleware

2.1. Integration of Different Administrative Domains

Grid Middleware (a computational view)

- **Services in the Grid Middleware layer**
  - Security
  - Information & Monitoring
  - Data Management
  - Execution
  - Meta-scheduling

- **Open Source Software Distributions**

- **Open Source Software Communities**

  - The Globus Alliance (dev.globus.org)
2. Grid Middleware

2.2. The Globus Toolkit

The Globus Alliance Community

*Open-Source Software Community* =
*Open-Source Software + Open Development Processes*

- **Open Community Project** based on Apache Jakarta model:
  - Control of each individual project is in hands of the committers
  - Public development infrastructure for each project: CVS, bugzilla, mailing list, and Wiki
  - Each project goes through an incubation process before becoming a Globus project

The Globus Toolkit

- Software distribution that integrates a selected group of Globus technologies
- GT provides basic services to allow secure remote operation over multiple administrative domains with different LRM systems and access policies.
2. Grid Middleware

2.2. The Globus Toolkit

Globus Components

Globus Projects
- GridWay

Globus Incubator Projects
- Gridshib
- DDM
- LRMA
- GRAADS
- CoG Workflow
- and many more!

Min. components for a Computational Grid
2. Grid Middleware

2.3. The GridWay Meta-scheduler

Global Architecture of a Computational Grid

- GridWay
  - Application-Infrastructure decoupling
  - DRMAA
    - .C, .java
  - Results
  - $> CLI

- Grid Meta-Scheduler
  - open source
  - job execution management
  - resource brokering

- Grid Middleware
  - Globus
    - Globus services
    - Standard interfaces
    - end-to-end (e.g. TCP/IP)

- Infrastructure
  - PBS, SGE
  - highly dynamic & heterogeneous
  - high fault rate

- Applications
  - Standard API (OGF DRMAA)
  - Command Line Interface

- Results

- Applications
2. Grid Middleware

2.3. The GridWay Meta-scheduler

Benefits

Integration of non-interoperable computational platforms (Organization)

- Establishment of a uniform and flexible infrastructure
- Achievement of greater utilization of resources and higher application throughput

Support for the existing platforms and LRM Systems (Sys. Admin.)

- Allocation of grid resources according to management specified policies
- Analysis of trends in resource usage
- Monitoring of user behavior

Familiar CLI and standard APIs (End Users & Developers)

- High Throughput Computing Applications
- Workflows
2. Grid Middleware

2.3. The GridWay Meta-scheduler

Features

Workload Management

• Advanced (Grid-specific) scheduling policies
• Fault detection & recovery
• Accounting
• Array jobs and DAG workflows

User Interface

• OGF standards: JSDL & DRMAA (C and JAVA)
• Analysis of trends in resource usage
• Command line interface, similar to that found on local LRM Systems

Integration

• Straightforward deployment as new services are not required
• Interoperability between different infrastructures
2. Grid Middleware

2.3. The GridWay Meta-scheduler

GridWay Internals

- GridWay Core
  - DRMAA library
  - CLI
  - GridWay Core
    - Job Pool
    - Host Pool
    - Request Manager
    - Dispatch Manager
    - Scheduler
      - Job Submission
      - Job Monitoring
      - Job Control
      - Job Migration
  - Transfer Manager
    - GridFTP
    - RFT
    - Grid File Transfer Services
  - Execution Manager
    - pre-WS GRAM
    - WS GRAM
    - Grid Execution Services
  - Information Manager
    - MDS2
    - MDS2 GLUE
    - MDS4
    - Grid Information Services
- Job Preparation
- Job Termination
- Job Migration
- Resource Discovery
- Resource Monitoring
2. Grid Middleware

2.3. The GridWay Meta-scheduler

Grid-specific Scheduling Policies

Resource Policies

- Rank Expressions
- Fixed Priority
- User Usage History
- Failure Rate

Grid Scheduling = Job + Resource Policies

Job Policies

- Fixed Priority
- Urgent Jobs
- User Share
- Deadline
- Waiting Time

Matching Resources for each job (user)
2. Grid Middleware

2.3. The GridWay Meta-scheduler

Centralized
Coupled

- Network Links
- Administration
- Homogeneity

Decentralized
Decoupled

SMP (Symmetric Multi-processors)
MPP (Massive Parallel Processors)
Clusters
Network Systems
Intranet/Internet

Grid Infrastructures

High Performance Computing

High Throughput Computing
Contents

1. Computing Resources
   1.1. Parallel and Distributed Computing
   1.2. Types of Computing Platforms
   1.3. Local Resource Management Systems

2. Grid Middleware
   2.1. Integration of Different Administrative Domains
   2.2. The Globus Toolkit
   2.3. The GridWay Meta-scheduler

3. A Taxonomy for Grid Scheduling Architectures
   3.1. The Taxonomy
   3.2. Multiple Administrative Domains
   3.3. Multiple Grid Infrastructures
   3.4. From the Cluster to the Grid
3. A Taxonomy for Grid Architectures

3.1. The Taxonomy

- Multiple Administrative Domains
  - Single Meta-scheduler Instance
  - Multiple Meta-scheduler Instances

- Multiple Grid Infrastructures
  - Single Meta-scheduler Layer
  - Multiple Meta-scheduler Layers

- Grid Interoperability

- Adapters
- Gateways

Admin. Domains | Grid | Middleware | Meta-scheduler
3. A Taxonomy for Grid Scheduling Architectures

3.2. Multiple Administrative Domains

Single Meta-Scheduler Grids

Characteristics

• One meta-scheduler instance with access to resources that may belong to different administrative domains
• Small scale infrastructures (campus or enterprise) that may be geographically distributed in different sites

Goal & Benefits

• Integrate multiple heterogeneous systems and/or administrative domains in an uniform/centralized infrastructure
• Improve return of IT investment
• Cost minimization
• Performance/Usage maximization

Scheduling

• Centralized meta-scheduler that allows the enforcement of Grid-wide policies (e.g. resource usage)
3. A Taxonomy for Grid Scheduling Architectures

3.2. Multiple Administrative Domains

Deploying Single Meta-Scheduler Grids with GridWay

- **Users**
  - GridWay
  - Globus
  - Globus
  - Globus

- **Applications**
  - DRMAA interface
  - Portal and/or CLI access

- **Middleware**
  - SGE Cluster
  - PBS Cluster
  - LSF Cluster
  - Services: MDS, GRAM, GridFTP
  - One scheduling instance
  - Grid-wide policies

- **Infrastructure**
  - Multiple Admin. Domains
  - Multiple Sites
3. A Taxonomy for Grid Scheduling Architectures

3.2. Multiple Administrative Domains

Single Meta-Scheduler Grids: Examples

European Space Astronomy Center

- Data Analysis from space missions (DRMAA)
- Site-level meta-scheduler
- Several clusters
3. A Taxonomy for Grid Scheduling Architectures

3.2. Multiple Administrative Domains

Single Meta-Scheduler Grids: Examples

**AstroGrid-D, German Astronomy Community Grid**

- Collaborative management of supercomputing resources & astronomy-specific resources
- Grid-level meta-scheduler (GRAM interface)
- 22 resources @ 5 sites, 800 CPUs
3. A Taxonomy for Grid Scheduling Architectures

3.2. Multiple Administrative Domains

Single Meta-Scheduler Grids: Examples

UABGrid, University of Alabama at Birmingham

- Bioinformatics applications
- Campus-level meta-scheduler
- 3 resources (PBS, SGE and Condor)
3. A Taxonomy for Grid Scheduling Architectures

3.2. Multiple Administrative Domains

Multiple Meta-Scheduler Grids

Characteristics

- Multiple meta-scheduler instances with access to resources belonging to different administrative domains (different organizations or partners)
- Large scale, loosely-coupled infrastructures (Partner Grids) shared by several Virtual Organizations

Goal & Benefits

- Large-scale, secure and reliable sharing of resources
- Support collaborative projects
- Access to higher computing power to satisfy peak demands

Scheduling

- Decentralized scheduling system that allows the enforcement of organization-wide policies
3. A Taxonomy for Grid Scheduling Architectures

3.2. Multiple Administrative Domains

Deploying Single Meta-Scheduler Grids with GridWay

- Users
- GridWay
- Globus
- SGE Cluster
- PBS Cluster
- LSF Cluster
- (Virtual) Organization
- Users
- GridWay
- Globus
- (V) Organization-wide policies
- DRMAA interface
- Science Gateways
- Services: MDS, GRAM, GridFTP
- Multiple scheduling instances
- Multiple Admin. Domains
- Multiple Organizations
- Infrastructure
- Middleware
- Applications
3. A Taxonomy for Grid Scheduling Architectures

3.2. Multiple Administrative Domains

Multiple Meta-Scheduler Grids: Examples

- **Fusion**
  - Users
  - GridWay
  - gLite
  - SGE Cluster

- **Biomed**
  - Users
  - GridWay
  - gLite
  - PBS Cluster

**CD-HIT workflow**

**Services:**
- BDII, GRAM, GridFTP
- DRMAA interface
- VO Schedulers
- EGEE Resource Broker
3. A Taxonomy for Grid Scheduling Architectures

3.3. Multiple Grid Infrastructures

Single Meta-Scheduler Layer Grids

Characteristics

• Single layer (one ore more meta-schedulers) with plain access to the underlying Grids
• (Virtual) Organizations involved in different Grid infrastructures

Goal & Benefits

• Integrate multiple Grids based on different middleware stacks
• Collaboration between trans-grid VOs

Scheduling

• Enforcement of organization-wide Grid-aware policies
• Adapters to interface different middleware stacks
3. A Taxonomy for Grid Scheduling Architectures

3.3. Multiple Grid Infrastructures

Deploying Single Meta-Scheduler Layer Grids with GridWay

- Trans-Grid VOs
- Multiple Middlewares
- Global names (DN's)
- Middleware adapters

GridWay

Users

(Virtual)
Organization

Applications

GridWay

Users

Middleware

• Multiple Middlewares
• Global names (DN’s)
• Middleware adapters

Infrastructure

Grid Infrastructure

SGE Cluster

PBS Cluster

Globus

gLite

Applications
3. A Taxonomy for Grid Scheduling Architectures

3.3. Multiple Grid Infrastructures

Single Meta-Scheduler Layer Grids: Example

- Different Middlewares (e.g. WS and pre-WS)
- Different Data/Execution architectures
- Different Information models
- Integration through adapters
- Global DN's
- See it running at TeraGrid07!
3. A Taxonomy for Grid Scheduling Architectures

3.3. Multiple Grid Infrastructures

Multiple Meta-Scheduler Layer Grids

Characteristics

• Multiple meta-scheduler layers in a hierarchical structure
• Resource provision in a utility fashion (provider/consumer)

Goal & Benefits

• Supply resources on-demand, making resource provision more adaptive
• Access to \textit{unlimited} computational capacity
• Transform IT costs from fixed to variable
• Seamless integration of different Grids (The Grid)

Scheduling

• Each Grid is handled as any other resource
• Characterization of a Grid as a single resource
• Use standard interfaces to virtualize a Grid infrastructure
3. A Taxonomy for Grid Scheduling Architectures

3.3. Multiple Grid Infrastructures

Deploying Multiple Meta-Scheduler Layer Grids with GridWay

globus-job-run, Condor/G, Nimrod/G …

Access to Outsourced Resources

Virtualization of a Grid

GRID-GATEWAY
3. A Taxonomy for Grid Scheduling Architectures

3.3. Multiple Grid Infrastructures

Multiple Meta-Scheduler Layer Grids: Example

- Access to different infrastructures with the same adapters
- EGEE managed as other resource

- Delegate identity/ "VO" certificates
- In-house/provider gateway

- Regional infrastructure
3.4. From the Cluster to the Grid

Interfaces Provided by Existing Grid Infrastructures

**Grid specific commands & API’s**
- Applications must be ported to the Grid
- Process (submission, monitoring…) must be adapted to the Grid
- New interfaces (e.g. portal) to simplify Grid use

**LRMS-like commands & API’s**
- A familiar environment to interact with a computational platform
- Some systems provide LRMS-like environment for Computational Grids
- Process still need to be adapted
- Applications would greatly benefit from standards (DRMAA)

*Transfer Queues: Seamless access to the Grid*
3. A Taxonomy for Grid Scheduling Architectures

3.4. From the Cluster to the Grid

Transfer Queues: Seamless access to the Grid

- Communicate LRM systems with meta-schedulers (the other way)
- Users keep using the same interface, even applications (e.g. DRMAA)

![Diagram showing the flow from the cluster to the grid]

Job submitted to the cluster but executed in the Grid
Thank you for your attention!