Miron Livny Seminar

Professor Miron Livny
Condor Project
Computer Sciences Department
University of Wisconsin - Madison
Thursday, November 3, 2005,
11:00 a.m.
Johnston Hall, Room 338
Reading

- Chapter 18 “Resource and Service Management”, The Grid 2
- Grid Resource Management: State of the Art and Future Trends
Recap

- **Globus GRAM:**
  - Grid Resource Allocation and Management
  - data staging, delegation of proxy credentials, and computation monitoring and management

- **Grid Application Toolkit:**
  - Abstract high level API to resource management
GRAM

http://www.globus.org/toolkit/docs/4.0/execution/key/WSGRAM_Approach.html
GAT

APIs to ...

- Form software and hardware descriptions
- Discover resources
- Submit and handle jobs

Interface to any grid resource management system.
Basic Problem

- **Resource Management:**
  - Discover resources
  - Allocate resources
  - Utilize resources
  - Monitor use of resources

- **Resources are:**
  - Computational services provided by hardware
  - Application services provided by software
  - Bandwidth on a network
  - Storage space on a storage system
Local Resource Management

- Much work has already been done:
  - Batch schedulers
  - Workflow engines
  - Operating systems

- Characteristics:
  - Have complete control of a resource and can implement mechanisms and policies for effective use.
Grid Resource Management

- Characterised by:
  - Different administrative domains
  - Heterogeneity of interfaces
  - Differing policies

- Need
  - Standard protocols
  - Standard semantics for resources and task requirements
GRM: Requirements

- Task submission
- Workload management
- On demand access (advance reservation)
- Co-scheduling
- Resource brokering

- Transactions and QoS
General Resource Management Framework
Service Level Agreements

- Resource providers “contract” in some way with a client through negotiations to provide some capability with a certain QoS
- SLAs state the terms of the agreement between the resource provider and resource user
  - Abstracts external grid usage from local use and policies
- Three different types of agreement:
  - Task TSLA: performance of activity or task e.g. a TSLA is created by submitting a job description to a queuing system
  - Resource RSLA: right to consume a resource without specifying what it will be used for (eg advance reservation)
  - Binding BSLA: application of a resource to a task (eg binding network bandwidth to a socket, or a number of nodes to a parallel job). BSLA associated a task defined by a TSLA to a RSLA.
Job/Task Description

- For example, JSDL (GGF working group), (also ClassAds, RSL)

- The Job Submission Description Language (JSDL) is a language for describing the requirements of computational jobs for submission to resources. The JSDL language contains a vocabulary and XML Schema that facilitate the expression of those requirements as a set of XML elements.

- Motivations:
  - Grids accommodate a variety of job management systems, where each system has its own language for describing job submission requirements, makes interoperability difficult.
  - Descriptions may be passed between systems or instantiated on resources matching the resource requirements for that job. All these interactions can be undertaken automatically, facilitated by a standard language that can be easily translated to each system’s own language.

- Additional JSDL consumers include: accounting systems, security systems, archiving systems, and provenance (auditing) systems.

- JSDL 1.0 provides the core vocabulary for describing a job for submission to Grid environments.
Job/Task Description

From JSDL specification
JSDL

- Description of jobs only at submission time:
  - No information about job identifiers or job status (job management)
  - Does not describe relationships between jobs (workflow)
  - Does not describe policies
  - Does not describe scheduling
  - etc
JSDL

- Contains types for
  - Processor architecture (sparc, powerpc, ...)
  - File system (swaom temporary, spool, normal, ...)
  - Operating system (WINNT, LINUX, ...)
  - File creation (overwrite, append, ...)
  - Ranges

- Core elements (just a sample)
  - Application name, version, description, resource requirements (filesystem, architecture, CPU speed, CPU time, virtual memory, diskspace), candidate hosts, required filesystems, delete file on terminate
Resource Description

- Need a language for
  - Clients to request resources
  - Servers to describe their resources

- Resource Description Languages
  - Some resources are configurable
  - Need to include lifetimes

- Examples:
  - RSL
  - ClassAds
  - CIM
RSL

- Globus Resource Specification Language
ClassAds

- **Condor Classified Advertisement Language**
- Represents characteristics of hosts (and jobs)
- A ClassAd is a set of uniquely named expressions --- called “attributes”

```plaintext
MyType = "Machine"
TargetType = "Job"
Name = "alfred.cs.wisc.edu"
Machine = "alfred.cs.wisc.edu"
StartIpAddr = "<128.105.83.11:32780>"
Arch = "INTEL"
OpSys = "SOLARIS251"
UidDomain = "cs.wisc.edu"
FileSystemDomain = "cs.wisc.edu"
State = "Unclaimed"
EnteredCurrentState = 892191963
Activity = "Idle"
EnteredCurrentActivity = 892191062
VirtualMemory = 185264
Disk = 35259
KFlops = 19992
Mips = 201
LoadAvg = 0.019531
CondorLoadAvg = 0.000000
KeyboardIdle = 5124
ConsoleIdle = 27592
Cpus = 1
Memory = 64
AFSCell = "cs.wisc.edu"
START = LoadAvg - CondorLoadAvg <= 0.300000 && KeyboardIdle > 15 * 60
Requirements = TRUE
Rank = Owner == "johndoe" || Owner == "friendofjohn"
CurrentRank = - 1.000000
LastHeardFrom = 892191963
```
CIM

- Common Information Model (CIM)
  - Standard designed by the Distributed Management Task Force
- Information about systems, networks, applications and services
- Becoming widely adopted in industry
Resource Discovery and Selection

- **Discovery**: query to identify resources where characteristics and state match those required (no commitment)
- **Selection**: choose based on different criteria (commit)
Task Management

- Monitor task status while executing, and status of managed resource (SLA status)
- Task status: queued, pending, running, terminated, etc.
- Change state of current SLAs or negotiate additional agreements
  - Terminate an SLA (application error, better resource found, unsatisfactory performance)
  - Extend SLA lifetime (task taking longer than expected)
  - Change SLA details (less/more disk space, QoS requirements change)
  - Create a new SLA
Grid Resource Management Systems

- Thus far these are pretty limited, one reason for this is because the local resource management systems are limited (e.g. no advance reservations)

- Examples:
  - Globus GRAM
  - General-purpose architecture for Reservation and Allocation (GARA)
  - Condor
  - Sun Grid Engine, PBS, Load Leveller, LSF.
For any system ...

- What platforms does it support?
- What is the architecture?
- Open source or cost?
- Job life cycle
- Security
- Job management (queues, job types (MPI, batch, interactive), checkpointing, file transfer)
- Resource management (tracking, reservations)
- Scheduling policies
- Resource matching
GRAM

- Does not implement local resource management functionality itself
- Interfaces to e.g. SGE, PBS, LSF, Load Leveller, Condor.
- Does not have advance reservation, but coallocation is supported via DUROC broker (eg need dedicated queues or exclusive access)
- Uses RSL both as a resource description language and for task descriptions.
- (GARA generalized GRAM to provide advanced reservation)
Condor

- High throughput scheduler
- Research project from University of Wisconsin-Madison
- Uses ClassAds for discovery and matching of resources with tasks
- Resources are organized into “condor pools” with one central manager (master) and arbitrary number of execution hosts (workers)
- Handles file transfer to and from submission host
- Has job priorities and various scheduling algorithms
Condor Pool

Condor

Submission host

Central manager

Execute host

Execute host

Execute host
Condor

- **Execution hosts:**
  - Advertise information about resource to central manager
  - Enforce policies of the resource owners
  - Start and monitor jobs

- **Submission hosts:**
  - Advertise job requests
  - Manage jobs running in the Condor pool
  - Handle checkpoints etc

- **Manager:**
  - Monitors information about the Condor pool
  - Matches resources with job requests
  - Handles cycle scavenging (watches for idle machines)
Condor and Globus

- Can use Globus GRAM to submit to Condor pool
  - Use condor job manager
- Can use Condor to submit to a Globus machine
  - Condor-G
  - Job descriptions are converted to RSL
Resource Brokers/Metaschedulers

- Virtualize interface to sets of resources
- Broker acts as an intermediary between VO and a set of resources
  - To date mainly for computational jobs and workflows
- Provides:
  - Simplified view of resources
  - Policy enforcement (community based)
  - Protocol conversion
- Metascheduling algorithms (usually for Grid the resource broker does not actually control the resources)
- Examples:
  - Sun Grid Engine, Platform LSF, PBSPro, GRMS
Grid Scheduling

- Three main paradigms:
  - Centralized, hierarchical, and distributed

- Centralized:
  - Central machine acts a resource manager. Scheduler has all necessary and up to date information. Does not scale well. Single point of failure

- Distributed:
  - Better scaling properties, fault tolerance and reliability. Lack of current information usually leads to suboptimal decisions.
  - Can be Direct or Indirect communication.

- Hierarchical:
  - Can have scalability and communication bottlenecks. Global and local schedulers can have different policies.
The Community Scheduler Framework 4.0 (CSF4.0) is a WSRF-compliant Grid level meta-scheduling framework built upon the Globus Toolkit. CSF provides interface and tools for Grid users to submit jobs, create advanced reservations and define different scheduling policies at the Grid level. Using CSF, Grid users are able to access different resource managers, such as LSF, PBS, Condor and SGE, via a single interface.
GRMS

- Grid Resource Management System
- From GridLab project
- Built on top of Globus
- Multicriteria matchmaking
- Workflow management
Future directions for grid resource management:
- Service oriented architecture
- General services (not just hardware)
- Provisioned rather than best-effort

Suggest a SLA-based resource management model independent of particular services
- Protocols used to negotiate these SLAs: Service Negotiation and Acquisition Protocol (SNAP)

Grid Resource Allocation Agreement Protocol working group of GGF.