Grid Resource Management and Scheduling

Figure: Ian Foster

Cluster Grid  Enterprise Grid  Global Grid
Slide Credits

- Ramin Yahyapour, University of Dortmund, Germany (Europar 2004) – Grid Scheduling Slides
- Jimmy Lin, University of Maryland, 2008 – Cloud Computing Slides
More resource types come into play:

- processing nodes, memory, storage, networks, experimental devices, instruments; data, software, licenses; people
- Distributed and heterogeneous resources, belonging to different administrative domains

More complex notion of ‘job’

- a job may involve different resources and consists of several activities in a workflow with according dependencies

HPC is still key the application for Grids. Consequently, the main resources in a Grid are the previously considered HPC machines with their local RMS
Complexity of Grid Resource Management

- **Security:** authentication, authorization, accounting
  - who has access to a certain resource?
  - what information can be exposed to whom?

- **Lack of global information:**
  - what resources are when available for an activity?

- **Heterogeneous resources:**
  - different RMS in use
  - individual access and usage paradigms
  - administrative policies have to be considered
Different Levels of Scheduling

- **Resource-level scheduler**
  - low-level scheduler, local scheduler, local resource manager
  - scheduler close to the resource, controlling a supercomputer, cluster, or network of workstations, on the same local area network
  - Examples: Open PBS, PBS Pro, LSF, SGE, Maui

- **Enterprise-level scheduler**
  - Scheduling across multiple local schedulers belonging to the same organization
  - Examples: PBS Pro peer scheduling, LSF Multicluster, Maui

- **Grid-level scheduler**
  - also known as super-scheduler, broker, community scheduler
  - Discovers resources that can meet a job’s requirements
  - Schedules across lower level schedulers
  - Examples: Globus
Grid-Level Scheduler

- Discovers & selects the appropriate resource(s) for a job
- If selected resources are under the control of several local schedulers, a meta-scheduling action is performed

- **Architecture:**
  - Centralized: all lower level schedulers are under the control of a single Grid scheduler
    - not realistic in global Grids
  - Distributed: lower level schedulers are under the control of several grid scheduler components; a local scheduler may receive jobs from several components of the grid scheduler
Grid Scheduling

Grid User

Grid-Scheduler

Scheduler

Schedule

Job-Queue

Machine 1

Scheduler

Schedule

Job-Queue

Machine 2

Scheduler

Schedule

Job-Queue

Machine 3
Activities of a Grid Scheduler

GF Document: “10 Actions of Super Scheduling (GFD-I.4)”

Phase One - Resource Discovery
1. Authorization Filtering
2. Application Definition
3. Min. Requirement Filtering

Phase Two - System Selection
4. Information Gathering
5. System Selection

Phase Three - Job Execution
6. Advance Reservation
7. Job Submission
8. Preparation Tasks
9. Monitoring Progress
10. Job Completion
11. Clean-up Tasks

Source: Jennifer Schopf
A Grid scheduler allows the user to specify the required resources and environment of the job without having to indicate the exact location of the resources.

A Grid scheduler answers the question: to which local resource manager(s) should this job be submitted?

Answering this question is hard:
- resources may dynamically join and leave a computational grid
- not all currently unused resources are available to grid jobs:
  - resource owner policies such as “maximum number of grid jobs allowed”
- it is hard to predict how long jobs will wait in a queue
Select a Resource for Execution

- Most systems do not provide advance information about future job execution
  - user information not accurate as mentioned before
  - new jobs arrive that may surpass current queue entries due to higher priority

- Grid scheduler might consider current queue situation, however this does not give reliable information for future executions:
  - A job may wait long in a short queue while it would have been executed earlier on another system.

- Available information:
  - Grid information service gives the state of the resources and possibly authorization information
  - Prediction heuristics: estimate job’s wait time for a given resource, based on the current state and the job’s requirements.
Selection Criteria

- Distribute jobs in order to balance load across resources
  - not suitable for large scale grids with different providers

- Data affinity: run job on the resource where data is located

- Best-fit: select the set of resources with the smallest capabilities and capacities that can meet job’s requirements

- Quality of Service of
  - a resource or
  - its local resource management system
    - what features has the local RMS?
    - can they be controlled from the Grid scheduler?
Scheduling Attributes

Working Group in the Global Grid Forum to: “Define the attributes of a lower-level scheduling instance that can be exploited by a higher-level scheduling instance.”

- Attributes of allocation properties
  - Revocation of an allocation
    - The local scheduler reserves the right to withdraw a given allocation.
  - Guaranteed completion time of allocation,
    - A deadline for job completion is provided by the local scheduler.
  - Guaranteed Number of Attempts to Complete a Job
    - A local scheduler will retry a given job task, e.g. useful for data transfer actions.
  - Allocations run-to-completion
    - A job is not preempted if it has been started
  - Exclusive Allocations
    - A job has exclusive access to the given resources; e.g. no time-sharing is performed
  - Malleable Allocations
    - The given resource set may change during runtime; e.g. a computational job will gain (moldable job) or lose processors
Scheduling Attributes (2)

- Attributes of available information
  - Access to tentative schedule
    - The local scheduler exposes his schedule of future allocations
    - Option: Only the projected start time of a specified allocation is available
    - Option: Only partial information on the current schedule is available
  - Exclusive control
    - The local scheduler is exclusively in charge of the resources; no other jobs can appear on the resources
  - Event Notification
    - The local scheduler provides an event subscription service

- Attributes for manipulating allocation execution
  - Preemption
  - Checkpointing
  - Migration
  - Restart
Scheduling Attributes (3)

- **Attributes for requesting resources**
  - **Allocation Offers**
    - The local system can provide an interface to request offers for an allocation.
  - **Allocation Cost or Objective Information**
    - The local scheduler can provide cost or objective information.
  - **Advance Reservation**
    - Allocations can be reserved in advance.
  - **Requirement for Providing Maximum Allocation Length in Advance**
    - The higher-level scheduler must provide a maximum job execution length.
  - **Deallocation Policy**
    - A policy applies for the allocation that must be met to stay valid.
  - **Remote Co-Scheduling**
    - A schedule can be generated by a higher-level instance and imposed on the local scheduler.
  - **Consideration of Job Dependencies**
    - The local scheduler can deal with dependency information of jobs; e.g. for workflows.
Grid Resource Management System consists of:

- **Local resource management system (Resource Layer)**
  - Basic resource management unit
  - Provide a standard interface for using remote resources

- **Global resource management system (Collective Layer)**
  - Coordinate all Local resource management system within multiple or distributed Virtual Organizations (VOs)
  - Provide high-level functionalities to efficiently use all of resources
    - Job Submission
    - Resource Discovery and Selection
    - Scheduling
    - Co-allocation
    - Job Monitoring, etc.
Grid Middleware

- User/Application
- Higher-Level Services

Resource Broker

- PBS
- LSF
- Grid Resource Manager

- Information Services
- Monitoring Services
- Security Services

Core Grid Infrastructure Services

Local Resource Management

Resource

Grid Middleware
Globus Grid Middleware

- **Globus Toolkit**
  - Open source software to serve as broker between user and Grid software
  - Standards maintained by the Globus Alliance at the Open Grid Forum

- **GRAM (Grid Resource Allocation and Management)**
  - Provides a single interface for requesting and using remote system resources for the execution of jobs.
  - remote job submission and control
  - uniform, flexible interface to local job scheduling systems
  - File and I/O management
  - Job monitoring
Globus Job Execution

- Job is described in the resource specification language
- Discover a Job Service for execution
  - Job Manager in Globus 2.x (GT2)
  - Master Management Job Factory Service (MMJFS) in Globus 3.x (GT3)
- Alternatively, choose a Grid Scheduler for job distribution
  - Grid scheduler selects a job service and forwards job to it
  - A Grid scheduler is not part of Globus
- The Job Service prepares job for submission to local scheduling system
- If necessary, file stage-in is performed
- The job is submitted to the local scheduling system
- If necessary, file stage-out is performed after job finishes.
Grid jobs are described in the resource specification language (RSL)

Can specify number of processes, number of requested nodes, max wall clock time, max CPU time, job queue

Example:

```
& (executable = a.out)
(directory = /home/nobody )
(arguments = arg1 "arg 2")
(count = 1)
```
A simple job submission requiring 2 nodes:

```
globus-job-run -np 2 -s myprog arg1 arg2
```

A multirequest specifies multiple resources for a job

```
globus-job-run -dumprs1 -: host1 /bin/uname -a \
 -: host2 /bin/uname -a
+ ( &resourceManagerContact="host1")
(subjobStartType=strict-barrier) (label="subjob 0")
(executable="/bin/uname") (arguments="-a")
) 
( &resourceManagerContact="host2")
(subjobStartType=strict-barrier)(label="subjob 1")
(executable="/bin/uname") (arguments="-a")
)```
Globus 2 Job Client Interface

- The full flexibility of RSL is available through the command line tool globusrun

- Support for file staging: executable and stdin/stdout

```bash
globusrun -o -r hpc1.acme.com/jobmanager-pbs '&(executable=$(HOME)/a.out) (jobtype=single) (queue=time-shared)'
```
Globus Job States

- Suspended
- Pending
- Stage-in
- Active
- Stage-out
- Failed
- Done
With transition to Web/Grid-Services, the job management becomes
- the Master Managed Job Factory Service (MMJFS)
- the Managed Job Factory Service (MJFS)
- the Managed Job Service (MJS)

The client contacts the MMJFS

MMJFS informs the MJFS to create a MJS for the job

The MJS takes care of managing the job actions.
- interact with local scheduler
- file staging
- store job status
Globus as a toolkit does not perform scheduling and automatic resource selection
Problem: Resource Management Systems Differ Across Each Component

<table>
<thead>
<tr>
<th></th>
<th>Interface Format</th>
<th>Execution Environment</th>
<th>Platform Mix</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>LSF</strong></td>
<td>Has API plus Batch Utilities via “LSF Scripts”</td>
<td>User: Local disk exported</td>
<td>Unix, Windows</td>
</tr>
<tr>
<td></td>
<td></td>
<td>System: Remote initialized (option)</td>
<td></td>
</tr>
<tr>
<td><strong>Grid Engine</strong></td>
<td>GDI API Interface plus Command line interface</td>
<td>System: Remote initialized, with SGE local variables exported</td>
<td>Unix only</td>
</tr>
<tr>
<td><strong>PBS</strong></td>
<td>API (script option)</td>
<td>System: Remote initialized, with PBS local variables exported</td>
<td>Unix only</td>
</tr>
<tr>
<td></td>
<td>Batch Utilities via “PBS Scripts”</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Source: Hrabri Rajic
Condor-G is a Condor system enhanced to manage Globus jobs

Descendant of Condor (University of Wisconsin).

It provides two main features

- **Globus Universe:** interface for submitting, queuing and monitoring jobs that use Globus resources
- **GlideIn:** system for efficient execution of jobs on remote Globus resources

Condor-G runs as a “personal Condor” system

- daemons run as non-privileged user processes
- each user runs her/his Condor-G
Globus is used to run the Condor daemons on Grid resources
- Condor daemons run as a Globus-managed job

GRAM service starts daemons rather than the Condor jobs
- When the resources run these GlideIn jobs, they will join the personal Condor pool

These daemons can be used to launch a job from Condor-G to a Globus resource

Jobs are submitted as Condor jobs and they will be matched and run on the Grid resources
- The daemons receive jobs from the user’s Condor queue

Combines the benefits of Globus and Condor
Using GlideIn

600 Condor jobs

Condor-G

GridManager

Collector

glide-ins

Grid Resource

JobManager

LSF

Startd

Source: ANL/USC ISI
DAGMan allows you to specify the dependencies between your Condor-G jobs, so it can manage them automatically for you.

(e.g., “Don’t run job “B” until job “A” has completed successfully.”)

A DAG is defined by a .dag file, listing each of its nodes and their dependencies:

```
# diamond.dag
Job A a.sub
Job B b.sub
Job C c.sub
Job D d.sub
Parent A Child B C
Parent B C Child D
```

each node will run the Condor-G job specified by its accompanying Condor submit file

Source: Miron Livny
Newer Generation Grids

New Grid Resource Management and Scheduling Approaches
Assume a data-intensive simulation that should be *visualized and steered during runtime!*
Example: Coordinated Simulation and Visualization

- Data
  - Data Access
  - Storing Data

- Network 1
  - Data Transfer

- Computer 1
  - Loading Data
  - Parallel Computation
  - Providing Data
  - Communication for Computation

- Network 2
  - Parallel Computation

- Computer 2
  - Software Usage

- Software License

- Storage
  - Data Storage
  - Communication for Visualization
  - Visualization

- Network 3

- VR-Cave

Reservations are necessary!
Service Oriented Architectures

- Services are used to abstract all resources and functionalities.

- Concept of OGSI and WSRF
  - using WebServices, SOAP, XML to implement the services
  - OGSI idea of GridServices is implemented in GT3
  - transition to WSRF with GT4

- Core service for building a Grid are discussed in the Open Grid Services Architecture (OGSA)
Open Grid Services
Architecture
Users in Problem Domain X

Applications in Problem Domain X

Application & Integration Technology for Problem Domain X

Generic Virtual Service Access and Integration Layer

Job Submission  Brokering  Workflow  Structured Data Integration
Registry  Banking  Authorisation  Structured Data Access
Data Transport  Resource Usage  Transformation

OGSI: Interface to Grid Infrastructure

Web Services: Basic Functionality

Compute, Data & Storage Resources

Distributed

Virtual Integration Architecture
Functional Requirements for Grid Scheduling

Functional Requirements:

- Cooperation between different resource providers
- Interaction with local resource management systems
- Support for reservations and service level agreements
- Orchestration of coordinated resources allocations
- Automatic handling of accounting and billing
- Distributed Monitoring
- Failure Transparency
What are Basic Blocks for a Grid Scheduling Architecture?

- Scheduling Service
- Data Management Service
- Network Management Service
- Information Service

Interfaces of Basic Blocks are still to be defined!
Information Service
Resource Discovery

Relevant for Grid Scheduling:

- Access to static and dynamic information

- Dynamic information include data about planned or forecasted future events
  - e.g.: existing reservations, scheduled tasks, future availabilities
  - need for anonymous and limited information (privacy concerns)

- Information about all resource types
  - including e.g. data and network
    - future reservation, data transfers etc.
Information about the job specifics (what is the job) and job requirements (what is required for the job) including data access and creation.

Need for common workflow description:
- e.g. a DAG formulation
- include static and dynamic dependencies
- need for the ability to extract workflow information to schedule a whole workflow in advance
Reservation Management Agreement and Negotiation

- Interaction between scheduling instances, between resource/agreement providers and agreement initiators (higher-level scheduler)
  - access to tentative information necessary
  - negotiations might take very long
  - individual scheduling objectives to be considered
  - probably market-oriented and economic scheduling needed

- Need for combining agreements from different providers
  - coordinate complex resource requests or workflows

- Maintain different negotiations at the same time
  - probable several levels of negotiations, agreement commitment and reservation
Accounting and Billing

- Interaction to budget information
- Charging for allocations, reservations; preliminary allocation of budgets
- Concepts for reliable authorization of Grid schedulers to spend money on behalf of the user
- Re-funding in terms of resource/SLA failure, re-scheduling etc.

- Reliable monitoring and accounting
  - required for tracing whether a party fulfilled an agreement
Monitoring Services

- Monitoring of
  - resource conditions
  - agreements
  - schedules
  - program execution
  - SLA conformance
  - workflow
  - ...

- Monitoring must be reliable as it is part of accountability
  - fail or fulfillment of a service/resource provider must be clearly identifiable
Brokerage Scheduling Model

Using a Brokerage/Trading strategy:

Submit Grid Job Description

Discover Resources

Query for Allocation Offers

Higher-level scheduling

Coordinate Allocations

Select Offers

Collect Offers

Generate Allocation Offer

Lower-level scheduling

Consider individual user policies

Consider individual owner policies

Analyze Query
Negotiation in Grids

- Multilevel Grid scheduling architecture
  - Lower level local scheduling instance
    - Implementation of owner policies
  - Higher level Grid scheduling instance
    - Resource selection and coordination

- (Static) Interface definition between both instances
  - Different types of resources
  - Different local scheduling systems with different properties
  - Different owner policies

- (Dynamic) Communication between both instances
  - Resource discovery
  - Job monitoring
The mapping of jobs to resources can be abstracted using the concept of Service Level Agreement (SLAs) (Czajkowski, Foster, Kesselman & Tuecke)

SLA: Contract negotiated between
- resource provider, e.g. local scheduler
- resource consumer, e.g., grid scheduler, application

SLAs provide a uniform approach for the client to
- specify resource and QoS requirements, while
- hiding from the client details about the resources,
- such as queue names and current workload
Service Level Agreement Types

- **Resource SLA (RSLA)**
  - A promise of resource availability
    - Client must utilize promise in subsequent SLAs
    - Advance Reservation is an RSLA

- **Task SLA (TSLA)**
  - A promise to perform a task
    - Complex task requirements
    - May reference an RSLA

- **Binding SLA (BSLA)**
  - Binds a resource capability to a TSLA
    - May reference an RSLA (i.e. a reservation)
    - May be created lazily to provision the task
  - Allows complex resource arrangements
Agreement-Based Negotiation

- A client (application) submits a task to a Grid scheduler
  - The client negotiates a TSLA for the task with the Grid Scheduler

- In order to provision the TSLA, the Grid Scheduler may obtain an RSLA with the Grid resource or may use a pre-existing RSLA that the Grid scheduler has negotiated speculatively
  - TSLA that refers to an RSLA assures the jobs gets the reserved resources at a specified time
  - TSLA without an RSLA tells little about when the resources will be available to the job
The job starts execution on the resource according to the TSLA and the RSLA
- For an existing TSLA, the Grid Scheduler may obtain additional RSLAs
- An RSLA is negotiated by the Grid Scheduler with the Resource
- A BSLA binds this RSLA to the corresponding TSLA

BSLAs allow to dynamically provision resources that are either
- not needed for the whole duration of the task or
- not known completely (e.g., the time at which a resource will be needed) before submitting the task
The Grid Scheduler receives requests for two agreements.

- It negotiates with the resources the RSLA1 and RSLA2, and
- in parallel with the agreement initiators about the corresponding TSLA1 and TSLA2.
GGF: GRAAP-WG

- Goal: Defining WebService-based protocols for negotiation and agreement management
- WS-Agreement Protocol:
Towards Market-based Grid Scheduling

Grid Scheduling Methods:
- Support for individual scheduling objectives and policies
- Multi-criteria scheduling models
- Economic scheduling methods to Grids

Architectural requirements:
- Generic job description
- Negotiation interface between higher- and lower-level scheduler
- Economic management services
- Workflow management
- Integration of data and network management
Grid Economics – Different Business Models

- Cost model
  - Use of a resource
  - Reservation of a resource

- Individual scheduling objective functions
  - User and owner objective functions
  - Formulation of an objective function
  - Integration of the function in a scheduling algorithm

- Resource selection
  - The scheduling instances act as broker
    - Collection and evaluation of resource offers
Economic Scheduling

- Market-oriented approaches are a suitable way to implement the interaction of different scheduling layers
  - agents in the Grid market can implement different policies and strategies
  - negotiations and agreements link the different strategies together
  - participating sites stay autonomous

- Needs for suitable scheduling algorithms and strategies for creating and selecting offers
  - need for creating the Pareto-Optimal scheduling solutions

- Performance relies highly on the available information
  - negotiation can be hard task if many potential providers are available.
Several possibilities for market models:
- auctions of resources/services
- auctions of jobs

Offer-request mechanisms support:
- inclusion of different cost models, price determination
- individual objective/utility functions for optimization goals

Market-oriented algorithms are considered:
- robust
- flexible in case of errors
- simple to adapt
- markets can have unforeseeable dynamics
Evaluation with utility functions

- A utility function is a mathematical representation of a user's preference
- The utility function may be complex and contain several different criteria

Example using response time (or delay time) and price:

\[ \text{util} = U_{\text{max}} - (a_1 \cdot \text{latency} + a_2 \cdot \text{price}) \]
The utility depends on the user and provider objectives
Following are some slides to illustrate the scheduling process in the NWIRE project.

NWIRE models a Grid infrastructure without central services

A P2P scheduling model is employed where Grid Scheduler/Managers that reside on each local site are queried for offers.

A Grid scheduler asks known other Schedulers.

Similar to P2P systems, no Grid manager knows the whole grid but keeps a list of “know other Grid managers”

Those schedulers can forward the request to other schedulers

The scope and depths of the query can be parameterized

Individual objective functions are used to evaluate the utility for a user and the resource provider.

The scheduler selects the allocation offer with the highest utility value

The whole model represents a Grid market and auction system.
Evaluation of Economic Approach

- Real Workload Data of Grids is not yet available.
- However, available are workloads of single HPC installations.

Example of evaluation results:

- considered: traces from Cornell Theory Center (430 nodes IBM RS/6000 SP)
- 4 workload adapted from the real traces, each 10000 jobs

<table>
<thead>
<tr>
<th>Name</th>
<th>Configuration</th>
<th>Largest Machine</th>
</tr>
</thead>
<tbody>
<tr>
<td>m64</td>
<td>4<em>64 + 6</em>32 + 8*8</td>
<td>64</td>
</tr>
<tr>
<td>m128</td>
<td>4*128</td>
<td>128</td>
</tr>
<tr>
<td>m256</td>
<td>2*256</td>
<td>256</td>
</tr>
<tr>
<td>m384</td>
<td>1<em>384 + 1</em>64 + 4*16</td>
<td>384</td>
</tr>
<tr>
<td>m512</td>
<td>1*512</td>
<td>512</td>
</tr>
</tbody>
</table>
1) Client sends a request

- **Application** sends a **Request** to the **Grid Manager**.

  - **Request** includes **Requirements**, **Job Attributes**, and **ObjectiveFunction**.

  - The **Grid Manager** then sends the request to **Resource Managers**.

**Diagram:**
- Application → Grid Manager
- Grid Manager → Resource Manager (3)
Example Request

KEY "Hops" {VALUE "HOPS" {2}"
KEY "MaxOfferNumber" {VALUE "MaxOfferNumber" {10}"
KEY "MinMulti-Site" {VALUE "MinMulti-Site" {10}"
...
KEY "Utility" {
    ELEMENT 1 {
        CONDITION{((OperatingSystem EQ "Linux")&& (NumberOfProcessors >= 8))}
        VALUE "UtilityValue" {-EndTime}
        VALUE "RunTime" {5}
    }
    ...
    ELEMENT 2 {
        CONDITION{ ... }
        VALUE "UtilityValue" {-JobCost}
    }
    ...
}
Scheduler requests offers from other Grid Managers/Schedulers.

Search depths and scope can be parameterized.
Selection of allocation according to the utility value
Returned offers are collected.
The user can directly be informed about the schedule for his execution.

The Grid Manager may re-schedule his allocations to optimize the result further.

5) User and Resource Managers are informed about allocations.
AWRT for Economic Scheduling Compared to Conservative Backfilling

Changes in average completion time of market-economic vs conventional scheduler

<table>
<thead>
<tr>
<th>Resource Configuration</th>
<th>Changes on pct</th>
</tr>
</thead>
<tbody>
<tr>
<td>M128</td>
<td>-5,000</td>
</tr>
<tr>
<td>M256</td>
<td>0,000</td>
</tr>
<tr>
<td>M384</td>
<td>-15,000</td>
</tr>
</tbody>
</table>

Conventional algorithms better
Markted-oriented better
Utilization

Changes in Utilization of market-economic vs conventional schedulers

Marketed-oriented better

Conventional better

Changes in pct

M128
M256
M384

Resource Configurations
AWRT for different Machine Utility Functions

Average Weighted Response Time in seconds

Maschinen-Funktion

- MF 1
- MF 2
- MF 3
- MF 4
- MF 5
- MF 6
Results on Economic Models

- Economical models provide results in the range of conventional algorithms in terms of AWRT
- Economic methods leave much more flexibilities in defining the desired resources
- Problems of site autonomy, heterogeneous resources and individual owner and user preferences are solved
- Advantage of reservation of resources ahead of time
- Further analysis needed:
  - Usage of other utility functions
Extending Resource Types

- Data is a key resource in the Grid
  - remote job execution will in most cases include the consideration of data
  - Data is a resource that can be managed
    - replication, caching
    - pre-fetching, post-fetching

- The requirements on network information and QoS features will increase
  - scheduling and planning data and resources requires information about available network bandwidth
  - how long does a reliable data transfer take?
  - is sufficient bandwidth available between two resources (e.g. between a visualization device and the compute resource)

- Both resource types as well as others will need suitable management and scheduling features
  - negotiation and agreement protocols provide already an infrastructure
Most new resource types can be included via individual lower-level resource management systems.

Additional considerations for

- **Data management**
  - Select resources according to data availability
  - But data can be moved if necessary!

- **Network management**
  - Consider advance reservation of bandwidth or SLA
  - Network resources usually depend on the selection of other resources!
  - Problem: no general model for network SLAs.

- Coordinate data transfers and storage allocation
Data Management

- Access to information about the location of data sets
- Information about transfer costs
- Scheduling of data transfers and data availability
  - optimize data transfers in regards to available network bandwidth and storage space
- Coordination with network or other resources

- Similarities with general grid scheduling:
  - access to similar services
  - similar tasks to execute
  - interaction necessary
Functional View of Grid Data Management

**Application**

**Planner:**
Data location, Replica selection, Selection of compute and storage nodes

**Executor:**
Initiates data transfers and computations

**Metadata Service**
Location based on data attributes

**Replica Location Service**
Location of one or more physical replicas

**Information Services**
State of grid resources, performance measurements and predictions

**Security and Policy**

**Data Movement**

**Data Access**

**Compute Resources**

**Storage Resources**

Source: Carl Kesselmann
Replica Location Service In Context

The Replica Location Service is one component in a layered data management architecture.

- Provides a simple, distributed registry of mappings
- Consistency management provided by higher-level services

Source: Carl Kesselmann