Grid Resource Management and Scheduling

Core Grid Services

- ***Security***: Grid Security Infrastructure
- ***Resource Management***: Grid Resource Allocation Management
- ***Information Services***: Grid Resource Information
- ***Data Transfer***: Grid File Transfer

Grid systems

- Classification: (depends on the author)
  - **Computational grid**:
    - *distributed supercomputing* (parallel application execution on multiple machines)
    - *high throughput* (stream of jobs)
  - **Data grid**: provides the way to solve large scale data management problems
  - **Service grid:** systems that provide services that are not provided by any single local machine.
    - *on demand:* aggregate resources to enable new services
    - *Collaborative:* connect users and applications via a virtual workspace
    - *Multimedia:* infrastructure for real-time multimedia applications

Taxonomy of Applications

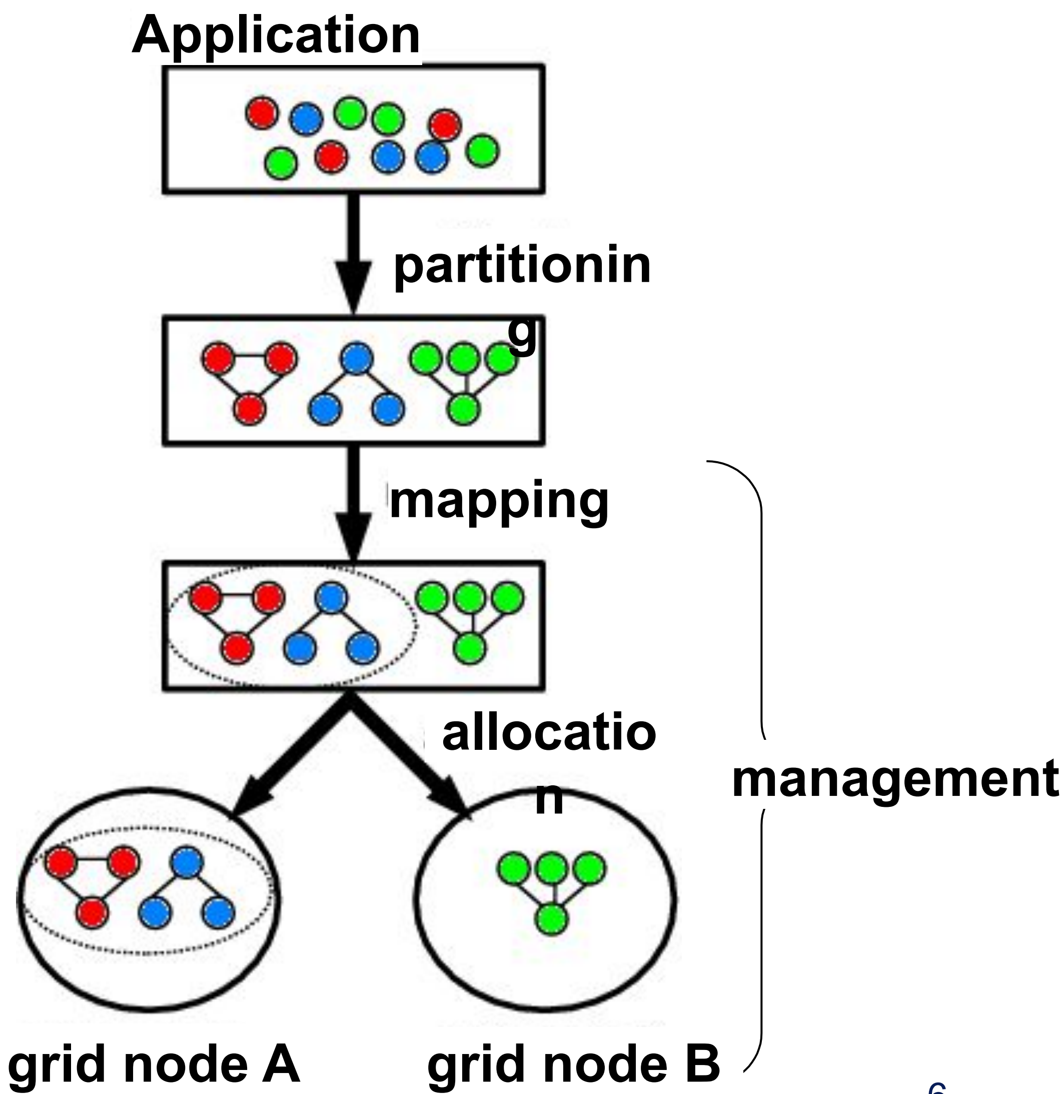
- ***High-Performance Computing (HPC):*** large amounts of computing power for short periods of time; tightly coupled parallel jobs
- ***High-Throughput Computing (HTC):*** large number of loosely-coupled tasks; large amounts of computing, but for much longer times (months and years); unused processor cycles
- ***On-Demand Computing*** meet short-term requirements for resources that cannot be cost-effectively or conveniently located locally
- ***Data-Intensive Computing*** processing large volumes of data
- ***Collaborative Computing*** enabling and enhancing human-to-human interactions (eg: CAVE5D system supports remote, collaborative exploration of large geophysical data sets and the models that generated them)

Alternative classification

- **independent tasks**
- **loosely-coupled tasks**
  - **loosely coupled** system is one in which each of its components has, or makes use of, **little or no knowledge** of the definitions of other separate components
- **tightly-coupled tasks**
  - Components are highly dependent on one another

Application Management

- Description
- Partitioning
- Mapping
- Allocation



Grid and HPC

- We all know what "the Grid" is…
  - one of the many definitions:

    "*Resource sharing & coordinated problem solving in dynamic, multi-institutional virtual organizations*" (Ian Foster)

  - however, the actual scope of "the Grid" is still quite controversial

- Many people consider High Performance Computing (HPC) as the main Grid application.
  - today's Grids are mostly Computational Grids or Data Grids with HPC resources as building blocks
  - thus, Grid resource management is much related to resource management on HPC resources (our starting point).
  - we will return to a broader Grid scope and its implications later

Resource Management on HPC Resources

- HPC resources are usually parallel computers or large scale clusters
- The local resource management systems (RMS) for such resources includes:
  - configuration management
  - monitoring of machine state
  - job management

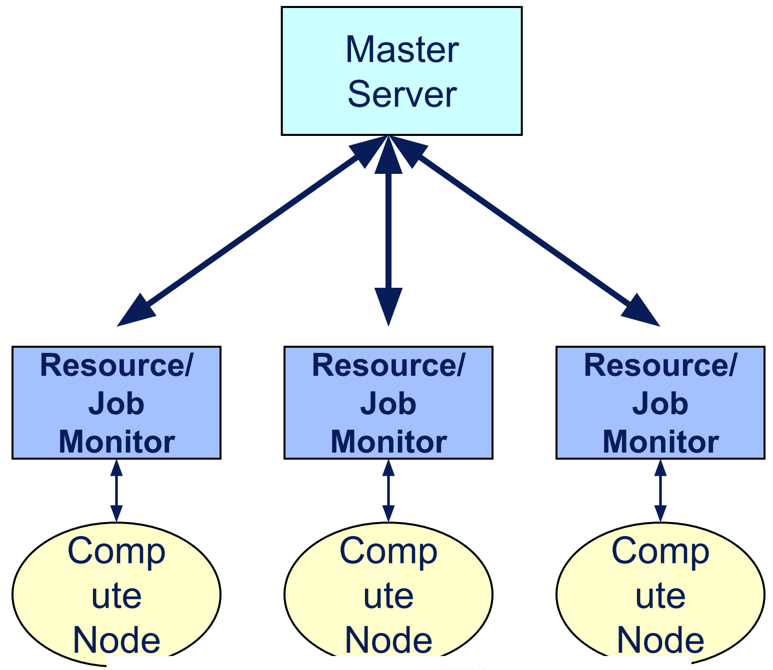
- There is no standard for this resource management.
- Several different proprietary solutions are in use.
- Examples for job management systems:
  - PBS, LSF, NQS, LoadLeveler, Condor

HPC Management Architecture in General

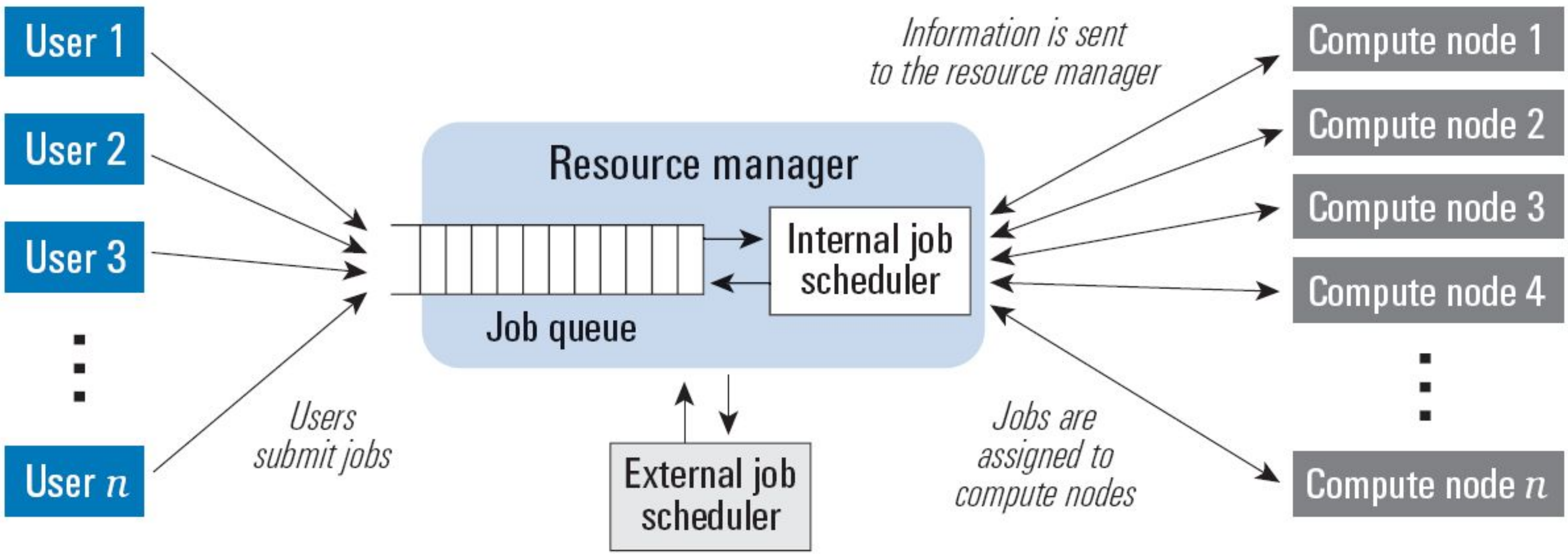
Control Service
Job Master

Resource and Job
Monitoring and Management Services

Compute Resources/
Processing Nodes



Typical cluster resource management



Computational Job

- A job is a computational task
  - that requires processing capabilities (e.g. 64 nodes) and
  - is subject to constraints (e.g. a specific other job must finish before the start of this job)

- The job information is provided by the user
  - resource requirements
    - CPU architecture, number of nodes, speed
    - memory size per CPU
    - software libraries, licenses
    - I/O capabilities
  - job description
  - additional constraints and preferences

- The format of job description is not standardized, but usually very similar

Example: PBS Job Description

- Simple job script:

```
#!/bin/csh

# resource limits: allocate needed nodes
#PBS -l nodes=1

#
# resource limits: amount of memory and CPU time
([[h:]m:]s).

#PBS -l mem=256mb
#PBS -l cput=2:00:00

# path/filename for standard output
#PBS -o master:/mypath/myjob.out


./my-task
```

whole job file is a shell script

information for the RMS are comments

the actual job is started in the script

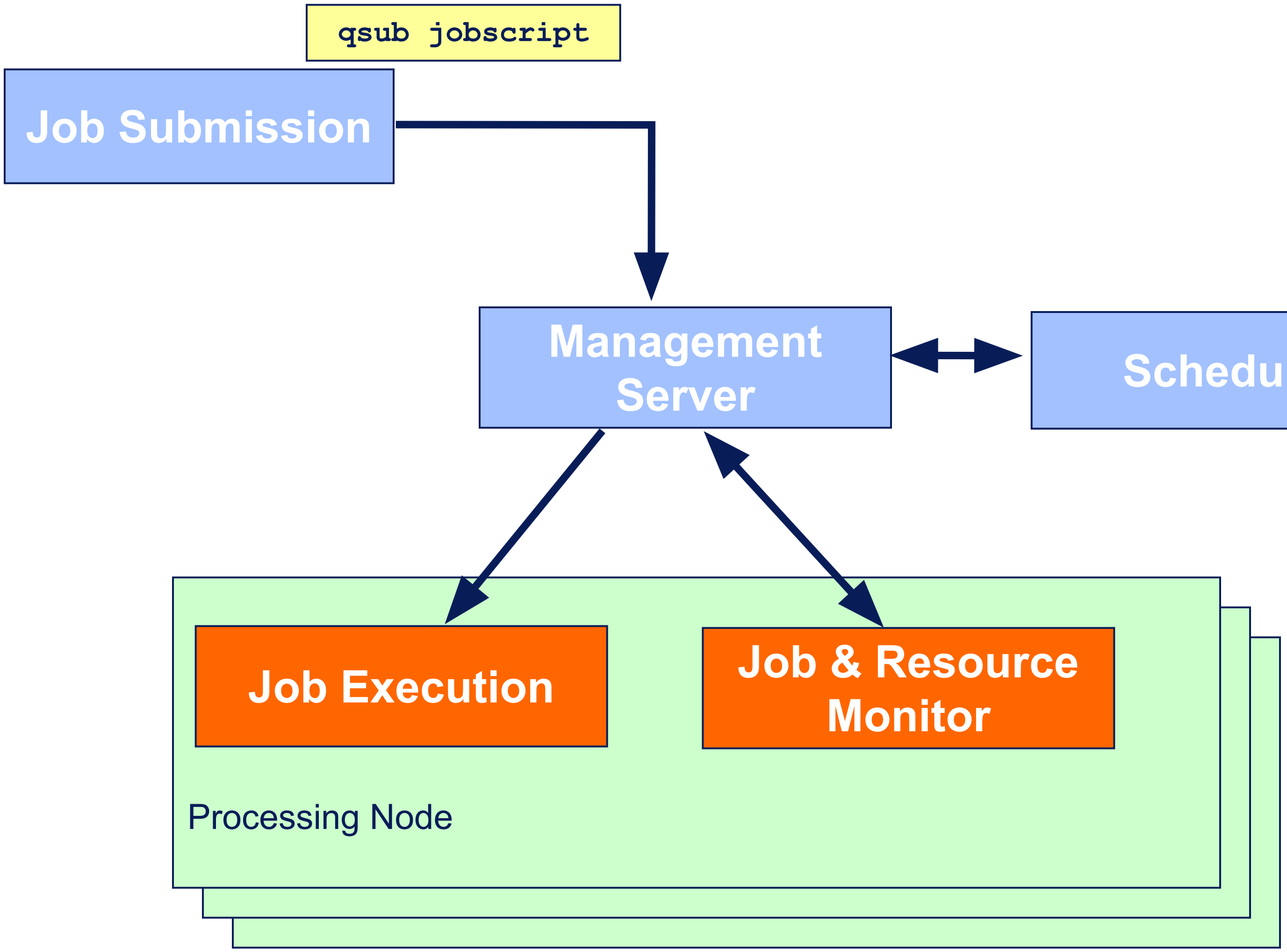
Job Submission

- The user "submits" the job to the RMS
  e.g. issuing "**`qsub jobscript.pbs`**"

- The user can control the job
  - qsub: submit
  - qstat: poll status information
  - qdel: cancel job

- It is the task of the resource management system to start a job on the required resources
- Current system state is taken into account

PBS Structure



Execution Alternatives

Time sharing:

- The local scheduler starts multiple processes per physical CPU with the goal of increasing resource utilization.
  - multi-tasking
- The scheduler may also suspend jobs to keep the system load under control
  - preemption

Space sharing:

- The job uses the requested resources exclusively; no other job is allocated to the same set of CPUs.
  - The job has to be queued until sufficient resources are free.

Job Classifications

- Batch Jobs vs interactive jobs
  - batch jobs are queued until execution
  - interactive jobs need immediate resource allocation
- Parallel vs. sequential jobs
  - a job requires several processing nodes in parallel

- the majority of HPC installations are used to run batch jobs in space-sharing mode!
  - a job is not influenced by other co-allocated jobs
  - the assigned processors, node memory, caches etc. are exclusively available for a single job.
  - overhead for context switches is minimized
  - important aspects for parallel applications

Preemption

- A job is preempted by interrupting its current execution
  - the job might be on hold on a CPU set and later resumed; job still resident on that nodes (consumption of memory)
  - alternatively a checkpoint is written and the job is migrated to another resource where it is restarted later

- Preemption can be useful to reallocate resources due to new job submissions (e.g. with higher priority)
- or if a job is running longer then expected.

Job Scheduling

- A job is assigned to resources through a scheduling process
  - responsible for identifying available resources
  - matching job requirements to resources
  - making decision about job ordering and priorities

- HPC resources are typically subject to high utilization
- therefore, resources are not immediately available and jobs are queued for future execution
  - time until execution is often quite long (many production systems have an average delay until execution of >1h)
  - jobs may run for a long time (several hours, days or weeks)

Typical Scheduling Objectives

- Minimizing the Average Weighted Response Time

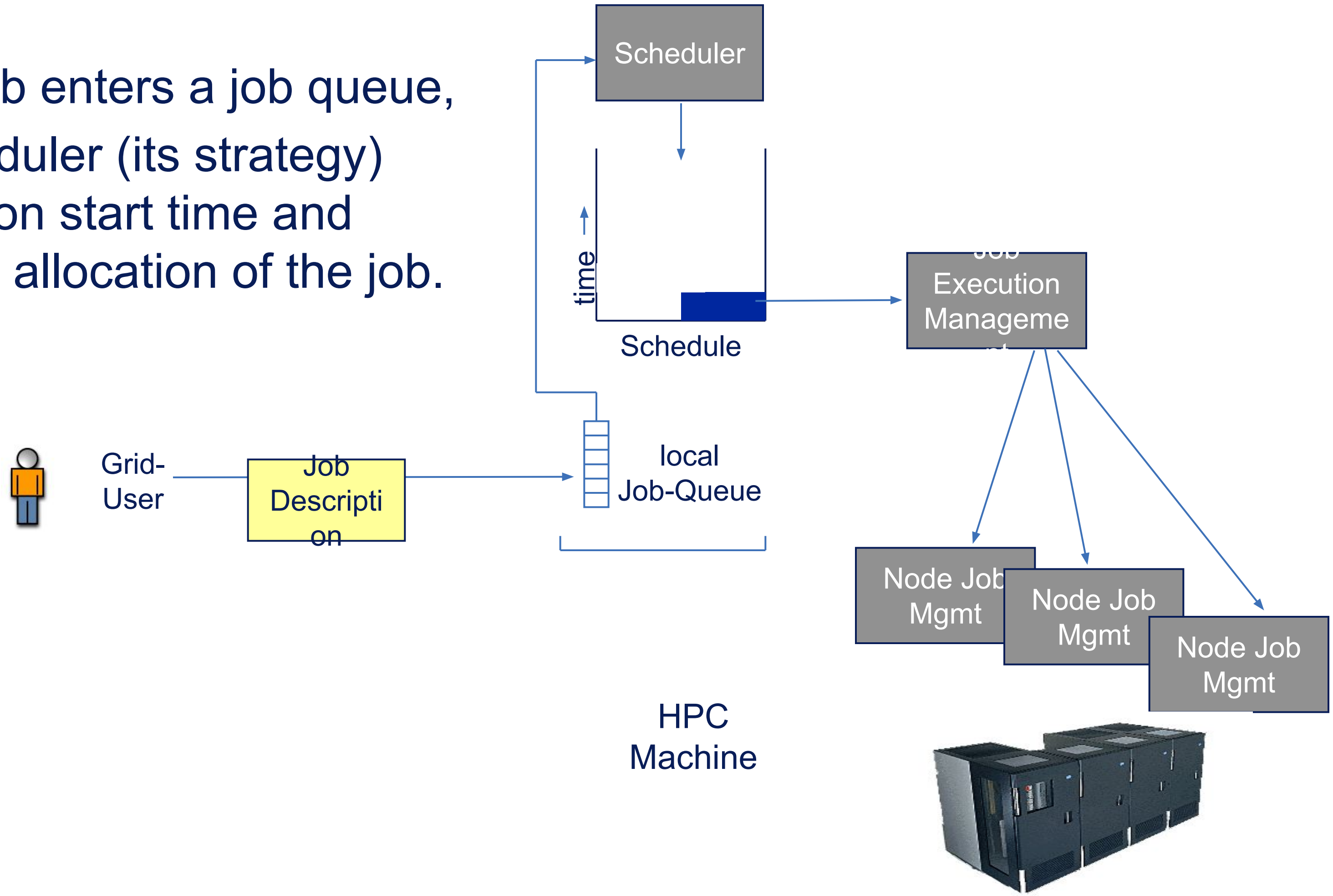
$$AWRT = \frac{\sum_{j \in Jobs} w_j \cdot (t_j - r_j)}{\sum_{j \in Jobs} w_j}$$

  - *r* : submission time of a job
  - *t* : completion time of a job
  - *w* : weight/priority of a job

- Maximize machine utilization/minimize idle time
  - conflicting objective
  - criteria is usually static for an installation and implicit given by the scheduling algorithm

Job Steps

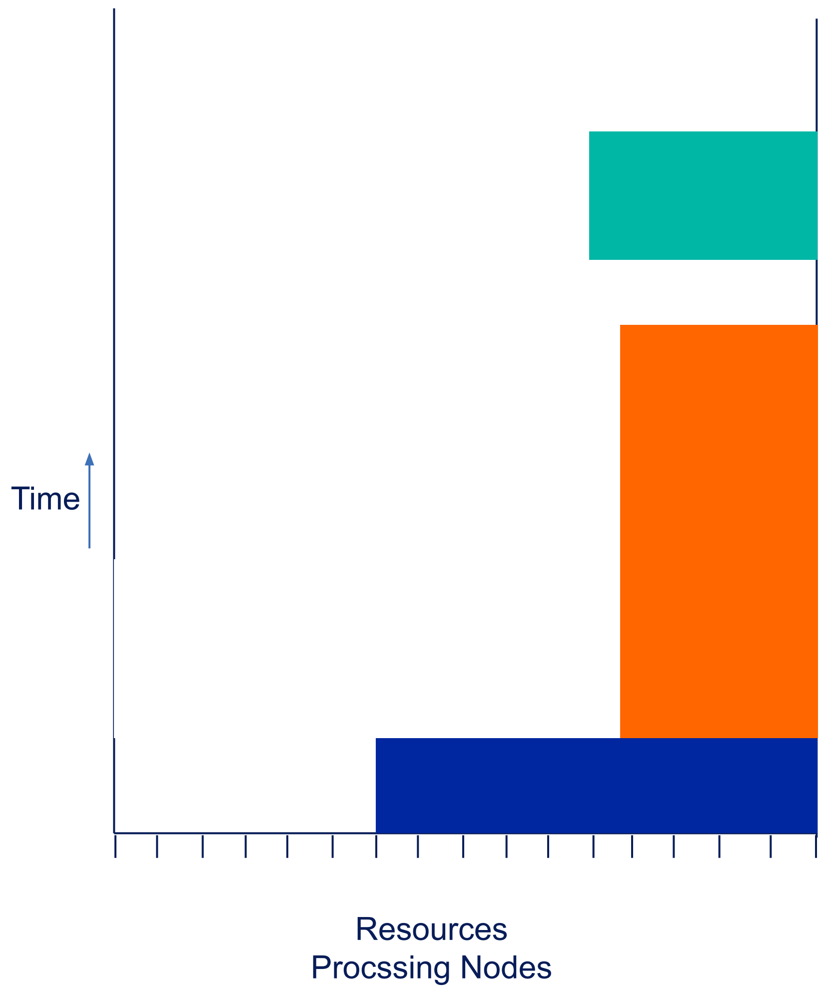
- A user job enters a job queue,
- the scheduler (its strategy) decides on start time and resource allocation of the job.

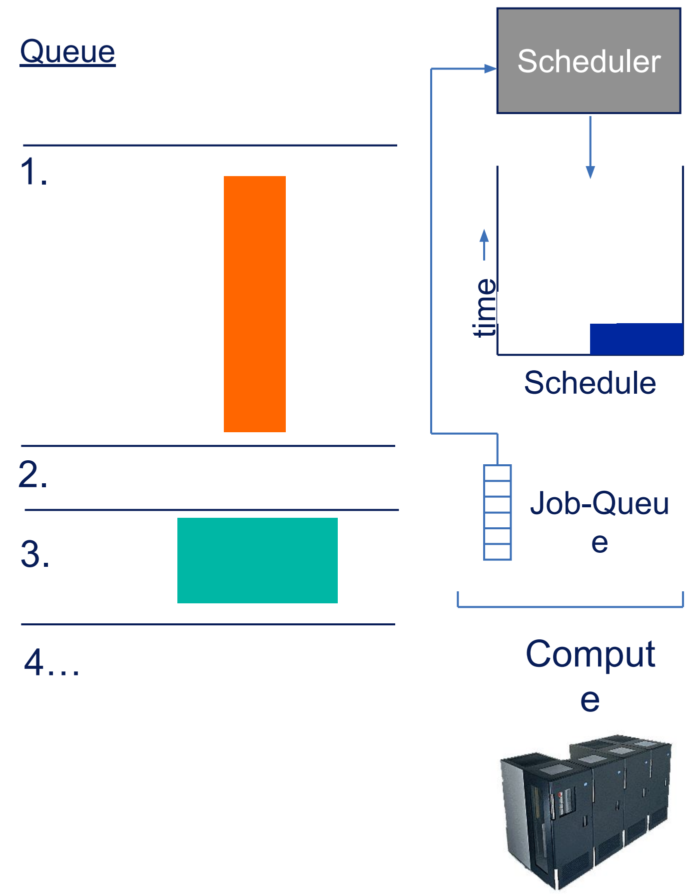


Scheduling Algorithms: FCFS

- Well known and very simple: First-Come First-Serve
- Jobs are started in order of submission
- Ad-hoc scheduling when resources become free again
  - no advance scheduling
- Advantage:
  - simple to implement
  - easy to understand and fair for the users
    (job queue represents execution order)
  - does not require a priori knowledge about job lengths
- Problems:
  - performance can extremely degrade; overall utilization of a machine can suffer if highly parallel jobs occur, that is, if a significant share of nodes is requested for a single job.

FCFS Schedule





Scheduling Algorithms: Backfilling

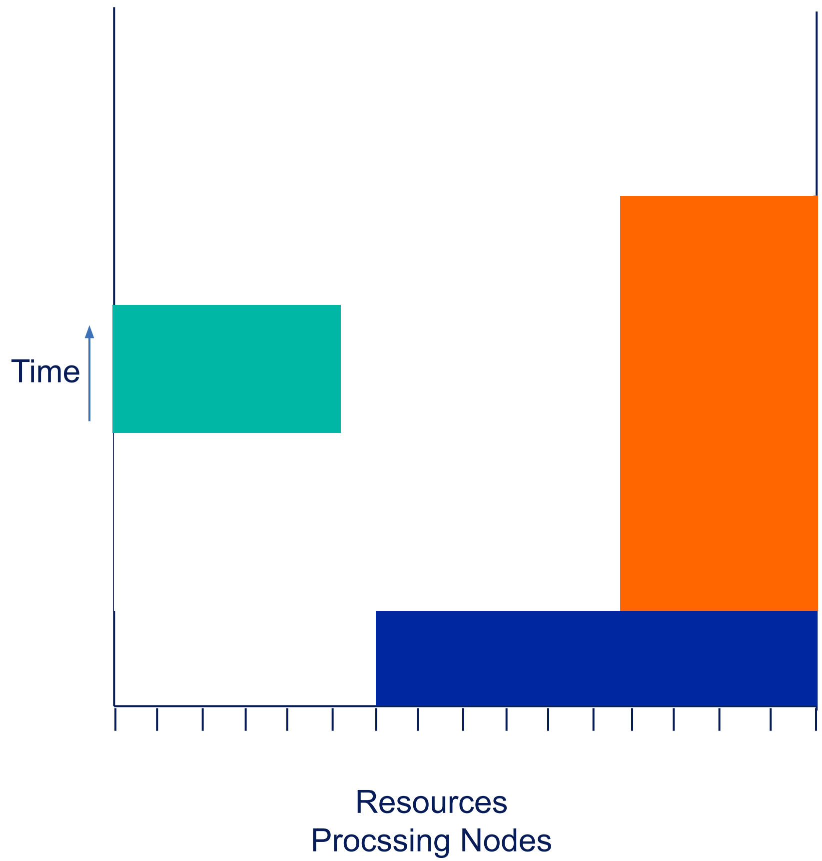
- Improvement over FCFS
- A job can be started before an earlier submitted job if it does not delay the first job in the queue
  - may still cause delay of other jobs further down the queue

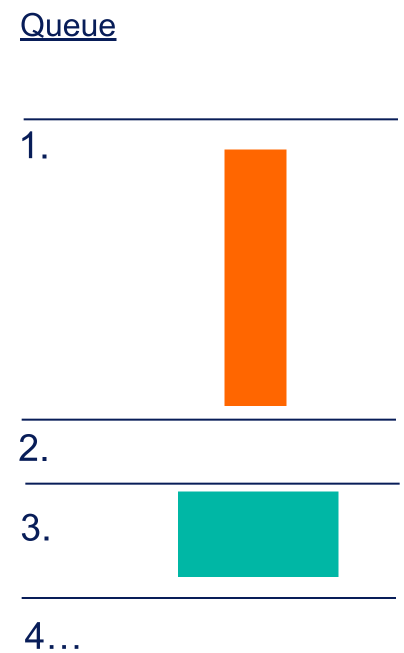
- Some fairness is still maintained
- Advantage:
  - utilization is improved

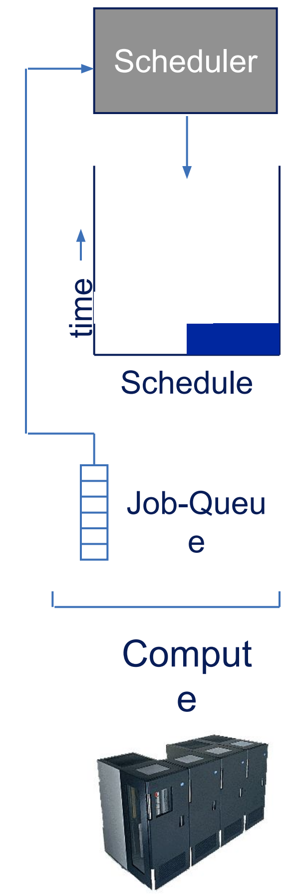
- Information about the job execution length is needed
  - sometimes difficult to provide
  - user estimation not necessarily accurate
  - Jobs are usually terminated after exceeding its allocated execution time;
  - otherwise users may deliberately underestimate the job length to get an earlier job start time

Backfill Scheduling

- Job 3 is started before Job 2 as it does not delay it



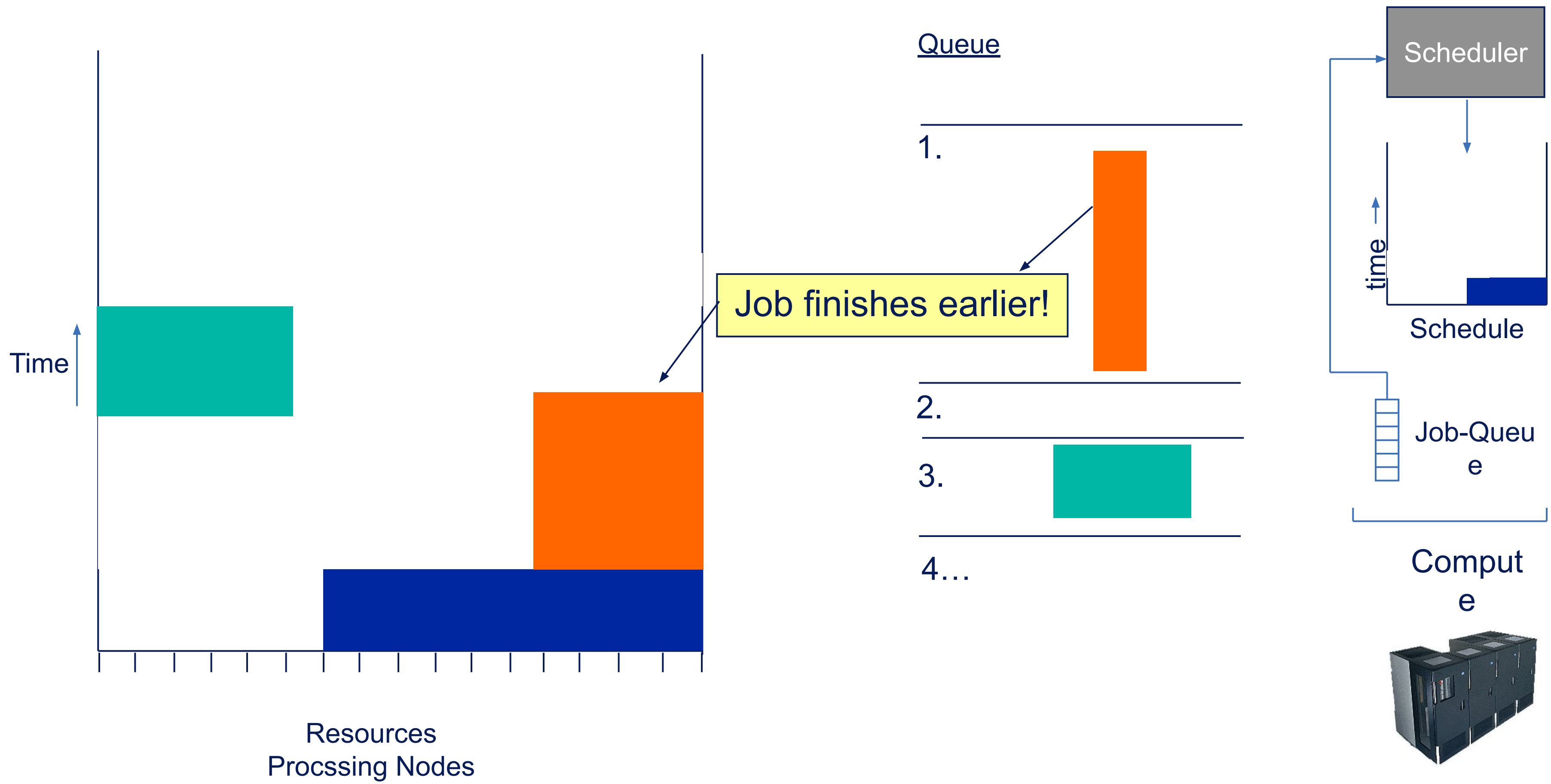




Backfill Scheduling

However, if a job finishes earlier than expected, the backfilling causes delays that otherwise would not occur

- need for accurate job length information (difficult to obtain)



Job Execution Manager

- After the scheduling process,
  the RMS is responsible for the job execution:
  - sets up the execution environment for a job,
  - starts a job,
  - monitors job state, and
  - cleans-up after execution (copying output-files etc.)
  - notifies the user (e.g. sending email)

Scheduling Options

- Parallel job scheduling algorithms are well studied; performance is usually acceptable

Real implementations may have addition requirements instead of need of more complex theoretical algorithms:

- Prioritization of jobs, users, or groups while maintaining fairness
- Partitioning of machines
  - e.g.: interactive and development partition vs. production batch partitions
- Combination of different queue characteristics

For instance, the Maui Scheduler is often deployed as it is quite flexible in terms of prioritization, backfilling, fairness etc.

Transition to Grid Resource Management and Scheduling

Current state of the art

Transition to the Grid

More resource types come into play:

- Resources are any kind of entity, service or capability to perform a specific task
  - processing nodes, memory, storage, networks, experimental devices, instruments
  - data, software, licenses
  - people
- The task/job/activity can also be of a broader meaning
  - a job may involve different resources and consists of several activities in a workflow with according dependencies
- The resources are distributed and may belong to different administrative domains
- 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

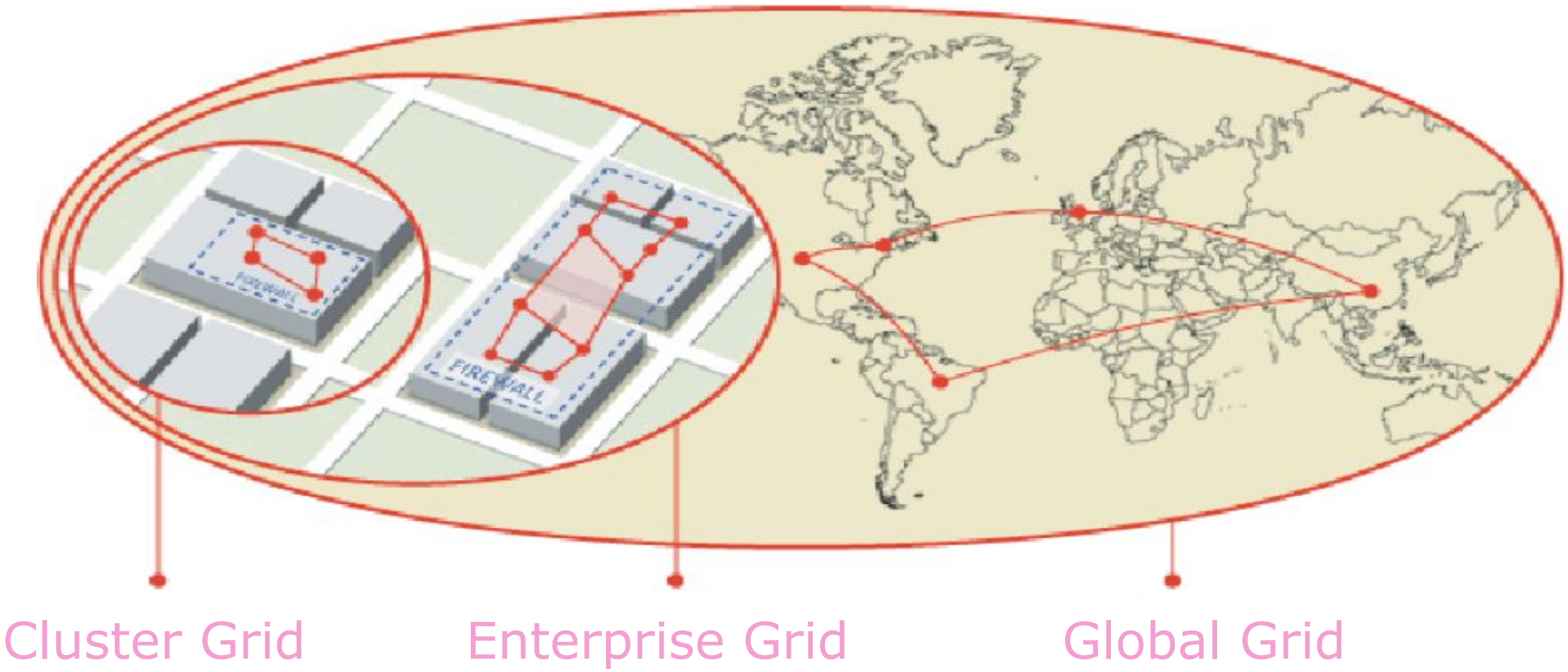
Implications to Grid Resource Management

- Several security-related issues have to be considered: authentication, authorization,accounting
  - who has access to a certain resource?
  - what information can be exposed to whom?

- There is lack of global information:
  - what resources are when available for an activity?

- The resources are quite heterogeneous:
  - different RMS in use
  - individual access and usage paradigms
  - administrative policies have to be considered

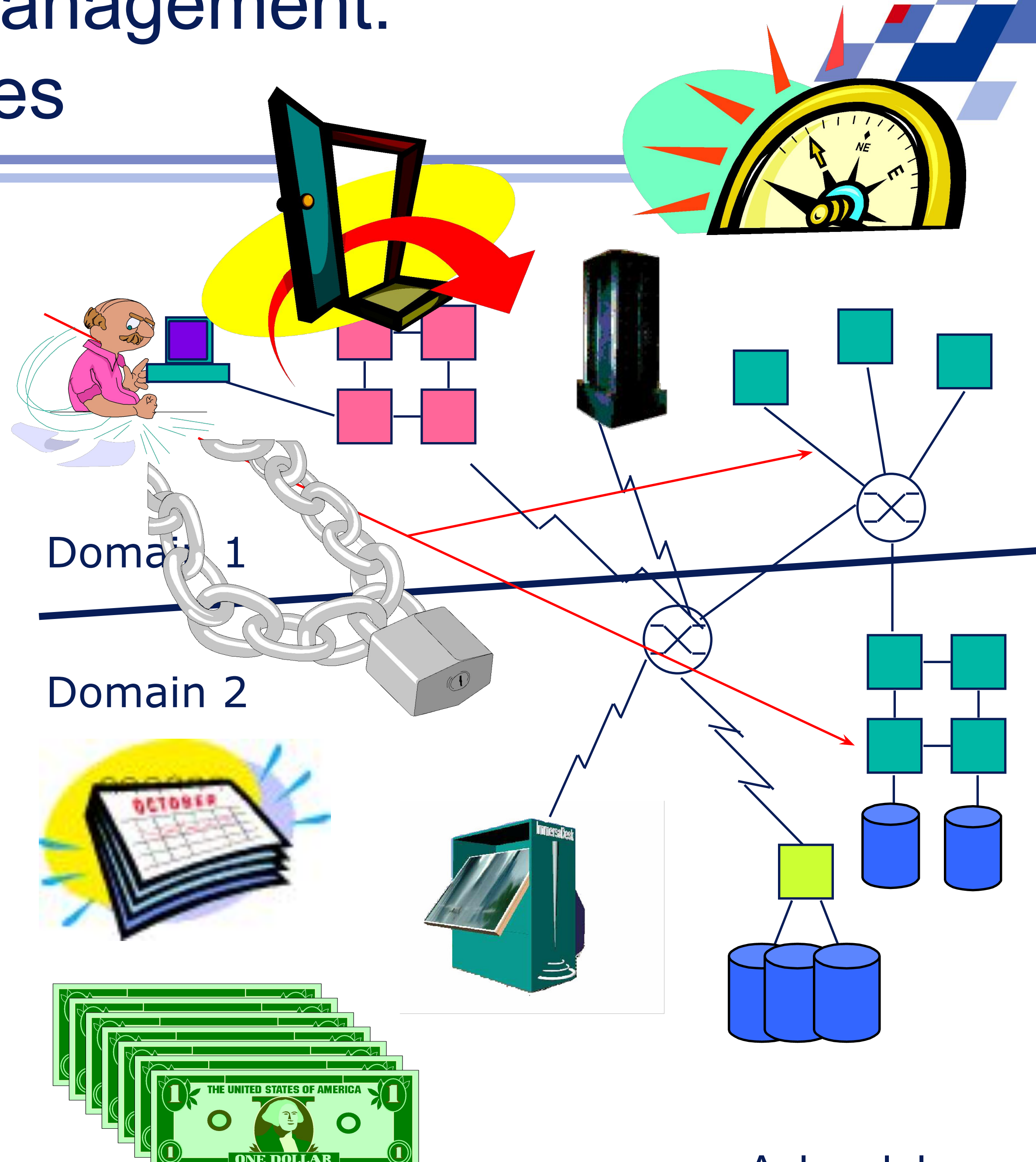
Scope of Grids



Source: Ian Foster

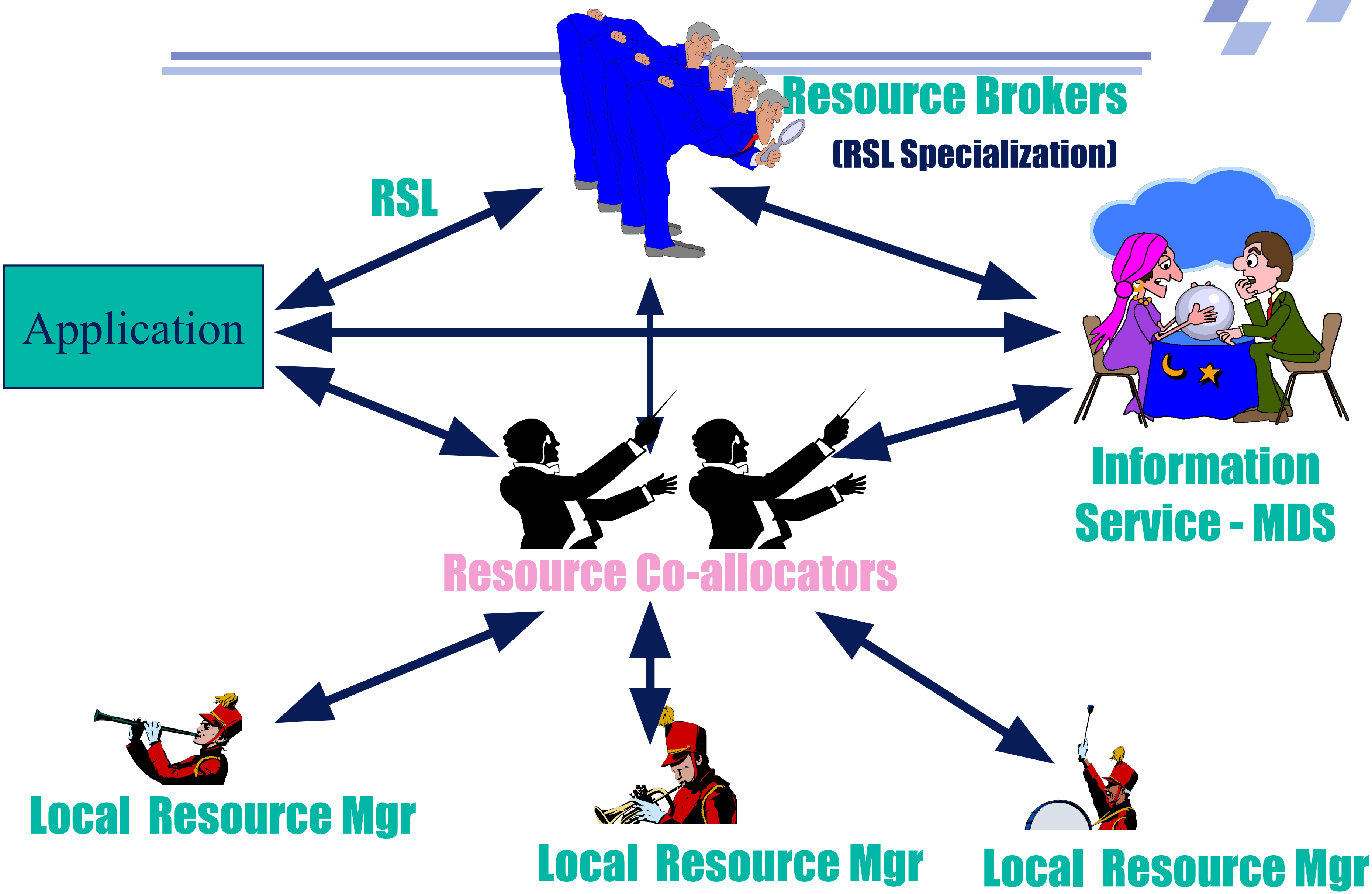
Grid Resource Management: Challenging Issues

- Authentication (once)
- Specify simulation (code, resources, etc.)
- Discover resources
- Negotiate authorization, acceptable use, Cost, etc.
- Acquire resources
- Schedule Jobs
- Initiate computation
- Steer computation
- Access remote data-sets
- Collaborate on results
- Account for usage



Ack.: globus..

Resource Management Architecture



Resource Management Layer

Grid Resource Management System consists of :

- *Local resource management system (Resource Layer)*
  - Basic resource management unit
  - Provide a standard interface for using remote resources
  - e.g. GRAM, etc.

- *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.
  - e.g. Meta-scheduler, Resource Broker, etc.

Remote Execution Steps

Choose Resource

Transfer Input Files

Set Environment

Start Process

Pass Arguments

Monitor Progress

Read/Write Intermediate Files

Transfer Output Files

Summary View
Job View
Event View

+Resource Discovery, Trading, Scheduling, Predictions, Rescheduling

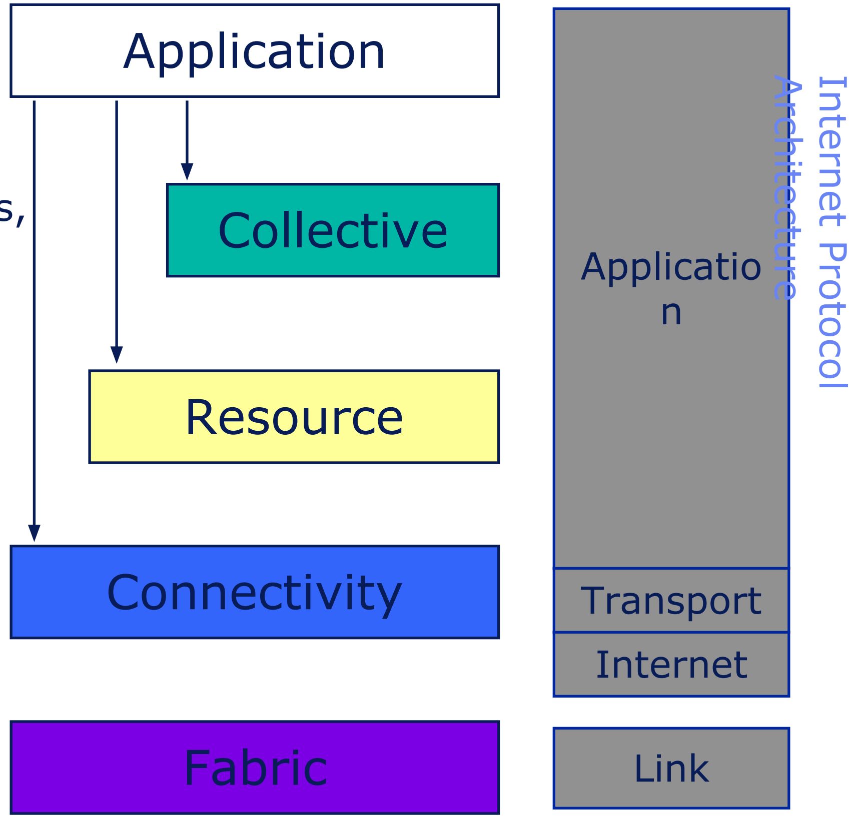
Grid Middleware

"Coordination of several resources": infrastructure services, application services

"Add resource": Negotiate access, control access and utilization

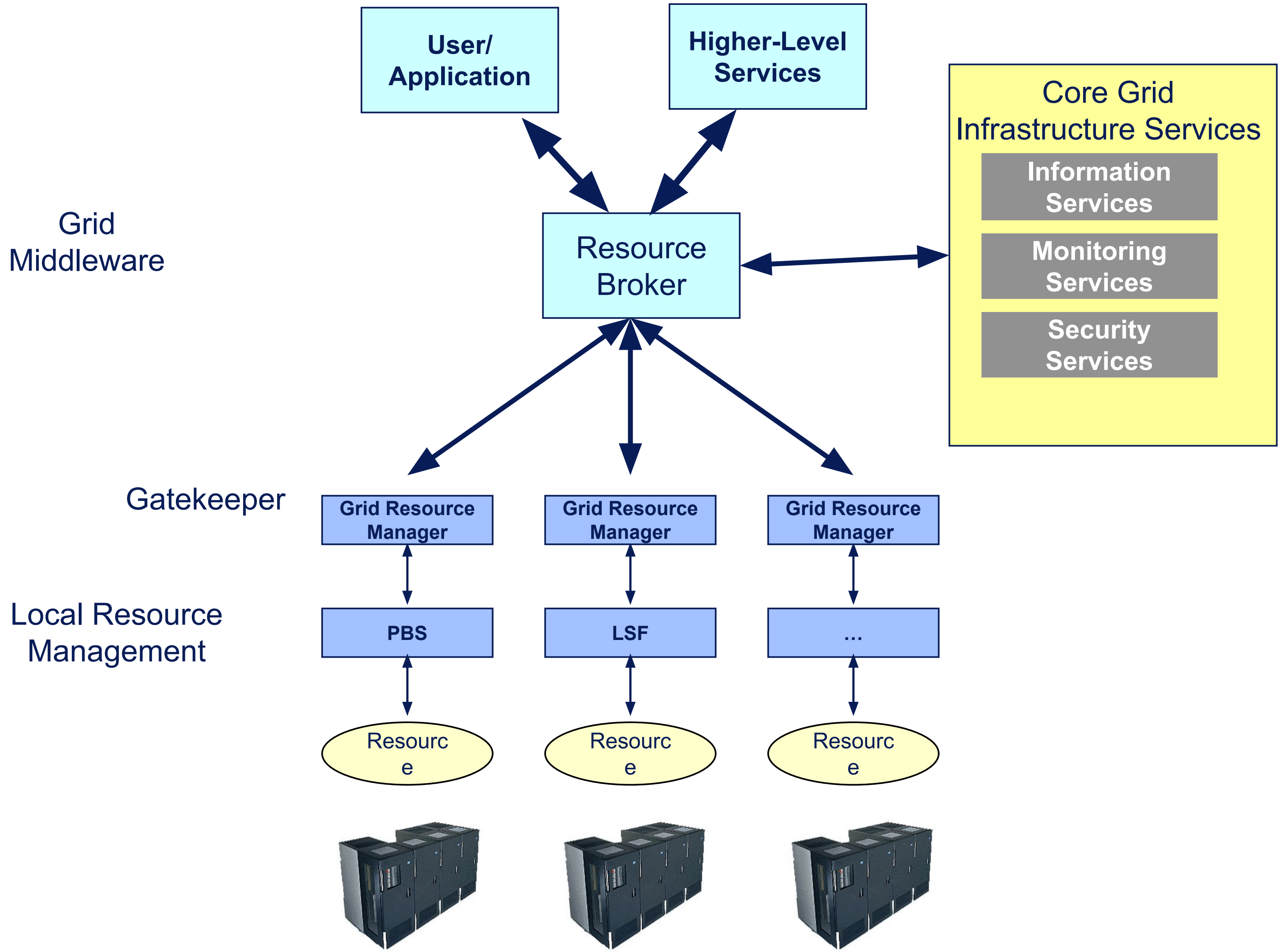
"Communication with internal resource functions and services"

"Control local execution"



Source: Ian Foster

Grid Middleware (2)



Globus Grid Middleware

- Globus Toolkit
  - common source for Grid middleware
  - GT2
  - GT3 – Web/GridService-based
  - GT4 – WSRF-based

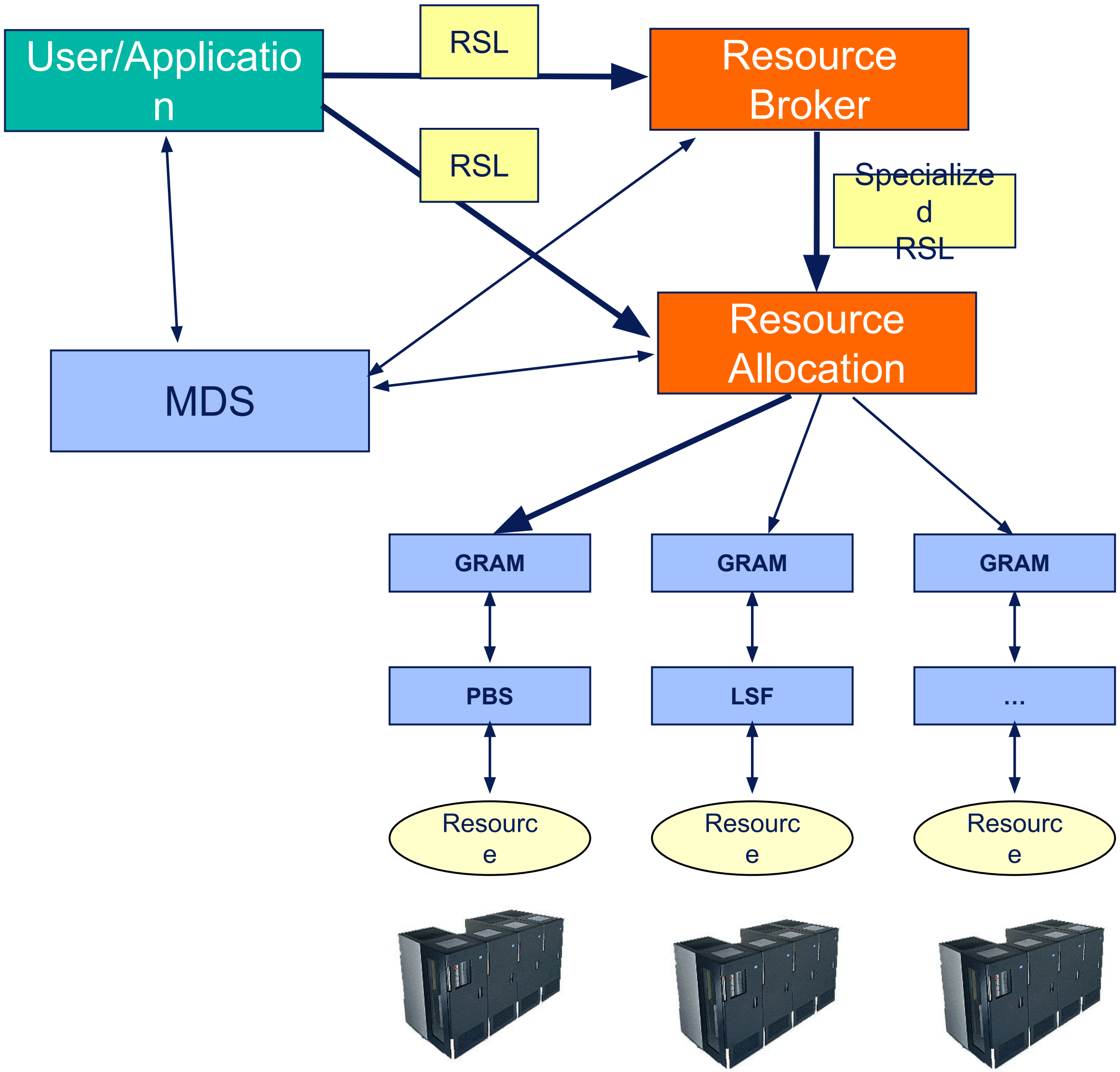
- GRAM is responsible for providing a service for a given job specification that can:
  - Create an environment for a job
  - Stage files to/from the environment
  - Submit a job to a local scheduler
  - Monitor a job
  - Send job state change notifications
  - Stream a job's stdout/err during execution

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
  - e.g. using the GASS service
- The job is submitted to the local scheduling system
- If necessary, file stage-out is performed after job finishes.

Globus GT2 Execution



RSL

- Grid jobs are described in the resource specification language (RSL)
- RSL Version 1 is used in GT2
- It has an LDAP filter-like syntax that supports boolean expressions:
- Example:

```
& (executable = a.out)
(directory  = /home/nobody )
(arguments  = arg1 "arg 2")
(count = 1)
```

Job Description with RSL2

- The version 2 of RSL is XML-based
- Two namespaces are used:
  - rsl: for basic types as int, string, path, url
  - gram: for the elements of a job

```
*GNS = "http://www.globus.org/namespaces"
<?xml version="1.0" encoding="UTF-8"?>
<rsl:rsl
     xmlns:rsl="GNS/2003/04/rsl"
     xmlns:gram="GNS/2003/04/rsl/gram"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="
           GNS/2003/04/rsl
           ./schema/base/gram/rsl.xsd
           GNS/2003/04/rsl/gram
           ./schema/base/gram/gram_rsl.xsd">
<gram:job>
      <gram:executable><rsl:path>
          <rsl:stringElement value="/bin/a.out"/>
      </rsl:path></gram:executable>
   </gram:job>
</rsl:rsl>
```

RSL2 Attributes

- <count> (type = rsl:integerType)
  - Number of processes to run (default is 1)
- <hostCount> (type = rsl:integerType)
  - On SMP multi-computers, number of nodes to distribute the “count” processes across
  - count/hostCount = number of processes per host
- <queue> (type = rsl:stringType)
  - Queue into which to submit job
- <maxWallTime> (type = rsl:longType)
  - Maximum wall clock runtime in minutes
- <maxCpuTime> (type = rsl:longType)
  - Maximum CPU runtime in minutes
- <maxTime> (type = rsl:longType)
  - Only applies if above are not used
  - Maximum wall clock or cpu runtime (schedulers’s choice) in minutes

Job Submission Tools

- GT 3 provides the Java class GramClient
- GT 2.x: command line programs for job submission
  - globus-job-run: interactive jobs
  - globus-job-submit: batch jobs
  - globusrun: takes RSL as input

Globus 2 Job Client Interface

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 -dumprsl -: 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

Example:

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

Problem: Job Submission Descriptions differ

The deliverables of the GGF Working Group JSDL:

- A specification for an abstract standard Job Submission Description Language (JSDL) that is independent of language bindings, including;
  - the JSDL feature set and attribute semantics,
  - the definition of the relationship between attributes,
  - and the range of attribute values.
- A normative XML Schema corresponding to the JSDL specification.
- A document of translation tables to and from the scheduling languages of a set of popular batch systems for both the job requirements and resource description attributes of those languages, which are relevant to the JSDL.

JSDL Attribute Categories

- The job attribute categories will include:
  - Job Identity Attributes
    - ID, owner, group, project, type, etc.
  - Job Resource Attributes
    - hardware, software, including applications, Web and Grid Services, etc.
  - Job Environment Attributes
    - environment variables, argument lists, etc.
  - Job Data Attributes
    - databases, files, data formats, and staging, replication, caching, and disk requirements, etc.
  - Job Scheduling Attributes
    - start and end times, duration, immediate dependencies etc.
  - Job Security Attributes
    - authentication, authorisation, data encryption, etc.

Grid Scheduling

How to select resources in the Grid?

Different Level of Scheduling

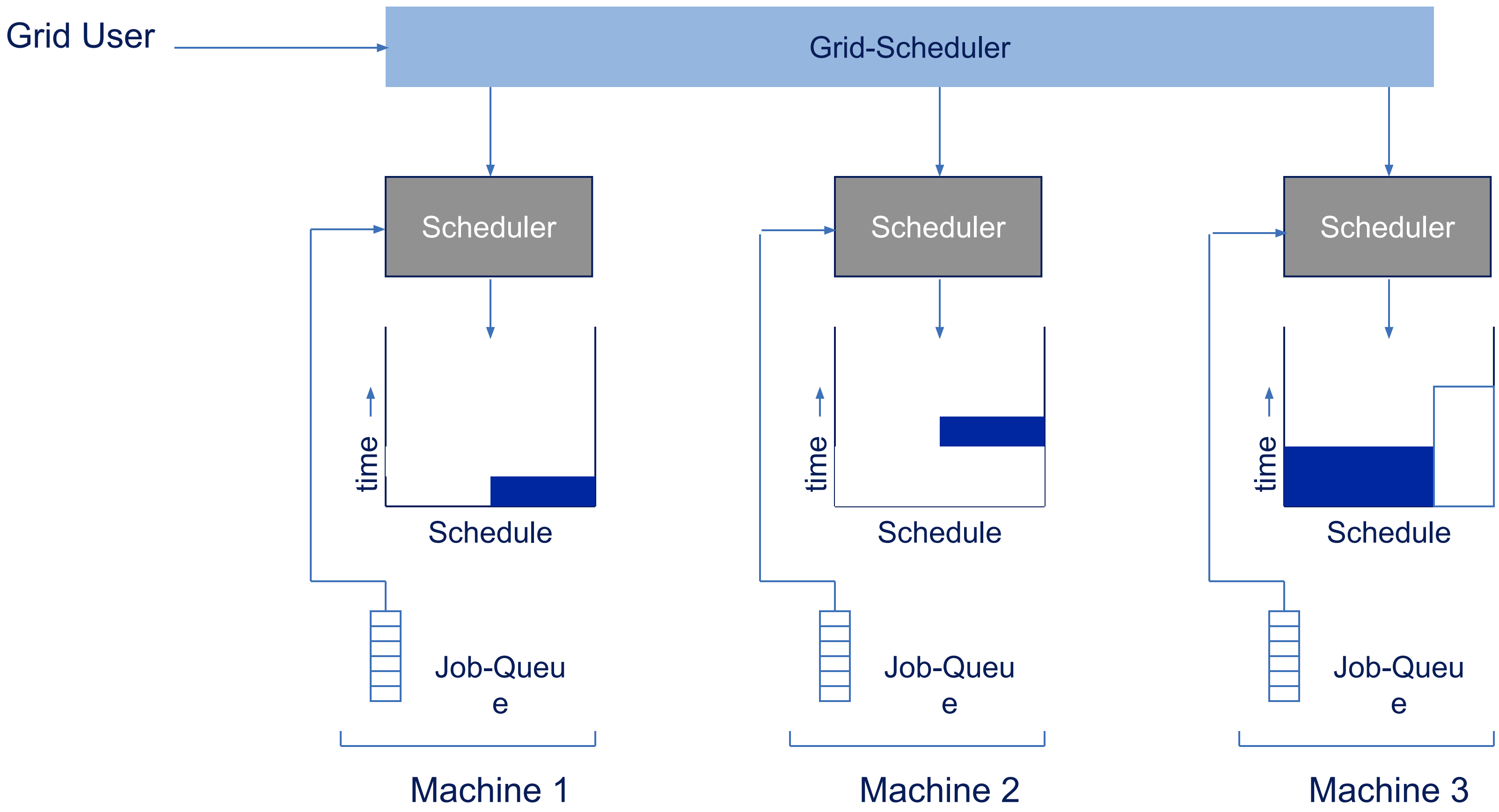
- **Resource-level scheduler**
  - low-level scheduler, local scheduler, local resource manager
  - scheduler close to the resource, controlling a supercomputer, cluster, or network of workstations, on the same local area network
  - Examples: Open PBS, PBS Pro, LSF, SGE
- **Enterprise-level scheduler**
  - Scheduling across multiple local schedulers belonging to the same organization
  - Examples: PBS Pro peer scheduling, LSF Multicluster
- **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
  - Example: gLite WMS, GridWay

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

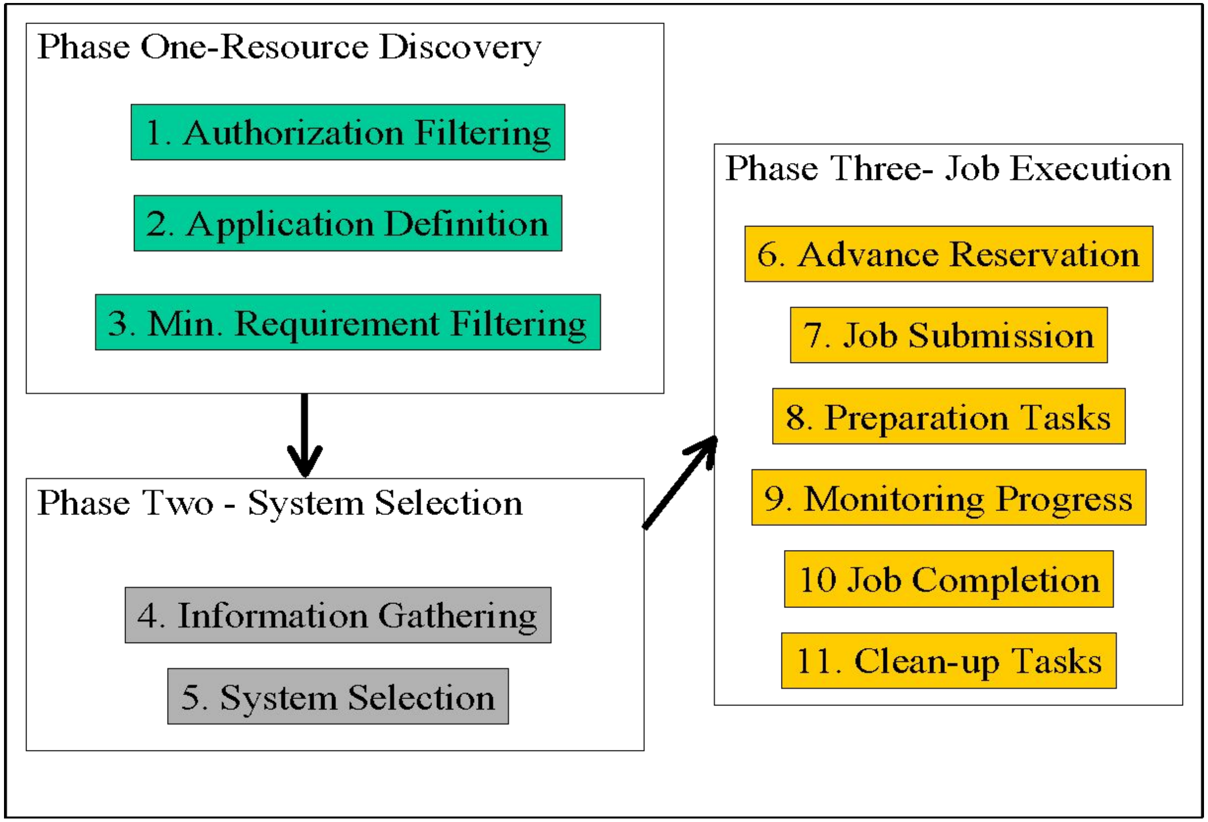
- Architecture:
  - Centralized: all lower level schedulers are under the control of a single Grid scheduler
    - not realistic in global Grids
  - Distributed: lower level schedulers are under the control of several grid scheduler components; a local scheduler may receive jobs from several components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

Grid Scheduling



Activities of a Grid Scheduler

- GGF Document:
  "10 Actions of Super Scheduling (GFD-I.4)"



Source: Jennifer Schopf

Grid Scheduling

- 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 manger(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

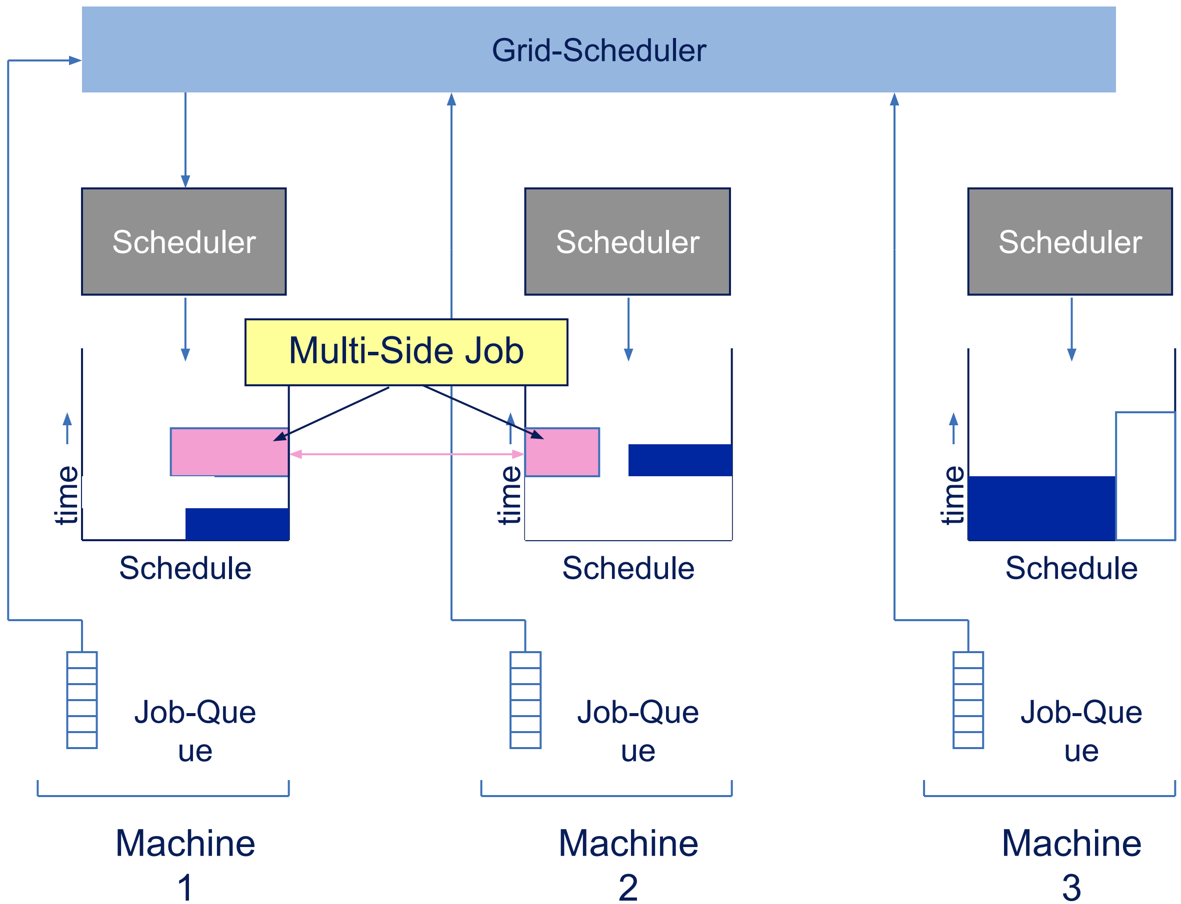
- Distribute jobs in order to balance load across resources
  - not suitable for large scale grids with different providers
- Data affinity: run job on the resource where data is located
- Use heuristics to estimate job execution time.
- Best-fit: select the set of resources with the smallest capabilities 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?

Co-allocation

- It is often requested that several resources are used for a single job.
  - that is, a scheduler has to assure that all resources are available when needed.
    - in parallel (e.g. visualization and processing)
    - with time dependencies (e.g. a workflow)
- The task is especially difficult if the resources belong to different administrative domains.
  - The actual allocation time must be known for co-allocation
  - or the different local resource management systems must synchronize each other (wait for availability of all resources)

Example Multi-Site Job Execution



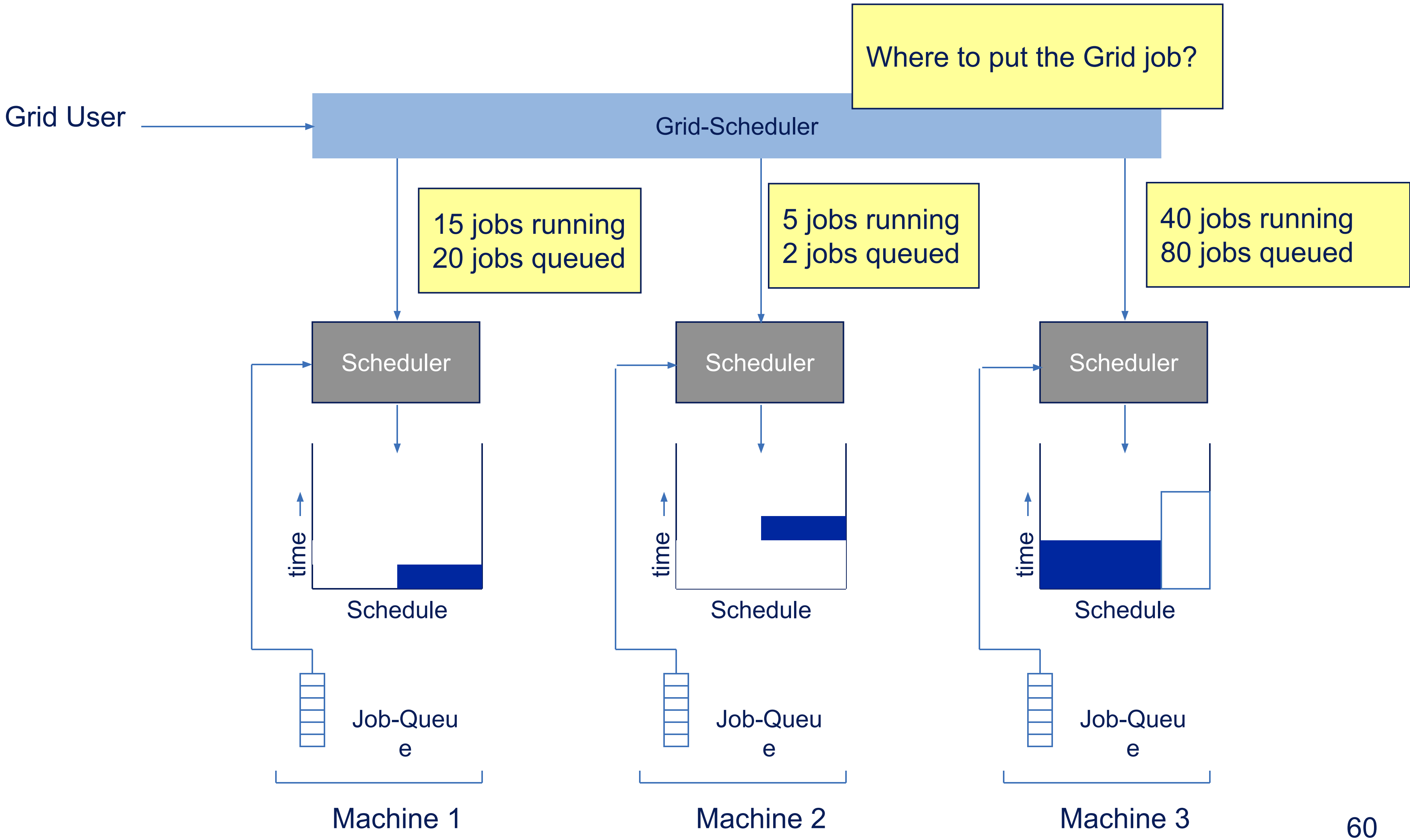
- A job uses several resources at different sites in parallel.
- Network communication is an issue.

Advanced Reservation

- Co-allocation and other applications require a priori information about the precise resource availability
- With the concept of advanced reservation, the resource provider guarantees a specified resource allocation.
  - includes a two- or three-phase commit for agreeing on the reservation

- Implementations:
  - GARA/DUROC/SNAP provide interfaces for Globus to create advanced reservation
  - implementations for network QoS available.
    - setup of a dedicated bandwidth between endpoints

Example of Grid Scheduling Decision Making



Available Information from the Local Schedulers

- Decision making is difficult for the Grid scheduler
  - limited information about local schedulers is available
  - available information may not be reliable

- Possible information:
  - queue length, running jobs
  - detailed information about the queued jobs
    - execution length, process requirements,…
  - tentative schedule about future job executions

- These information are often technically not provided by the local scheduler
- In addition, these information may be subject to privacy concerns!

Consequence

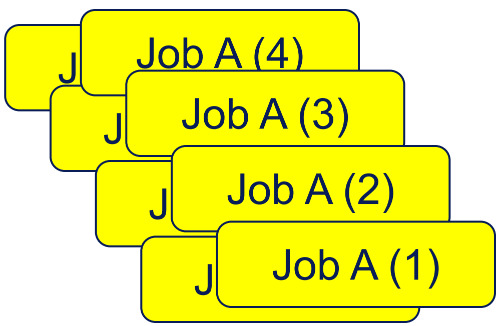
- Consider a workflow with 3 short steps (e.g. 1 minute each) that depend on each other
- Assume available machines with an average queue length of 1 hour.
- The Grid scheduler can only submit the subsequent step if the previous job step is finished.

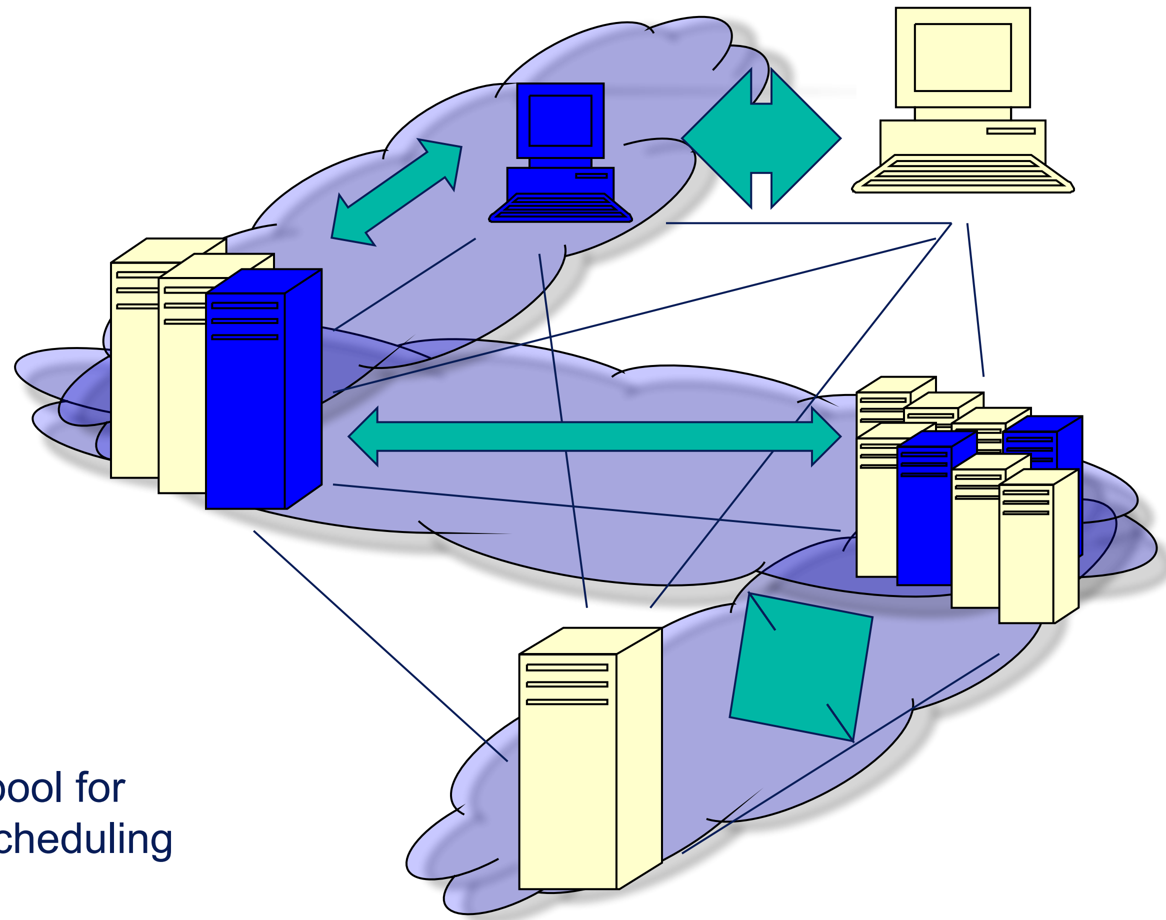
- Result:
  - The completion time of the workflow may be larger than 3 hours (compared to 3 minutes of execution time)
  - Current Grids are suitable for simple jobs, but still quite inefficient in handling more complex applications

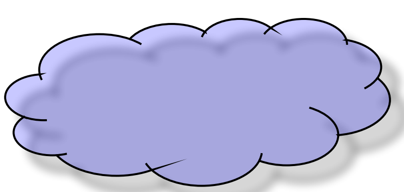
- Need for better coordination of higher- and lower-level scheduling!

User-level scheduling

- Using “placeholder” or “pilot” jobs that acquire resources and accept further application requests directly







- resource pool for User-Level Scheduling

Data and Network Scheduling

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

Example of a Scheduling Process

Scheduling Service:

1. receives job description
2. queries Information Service for static resource information
3. prioritizes and pre-selects resources
4. queries for dynamic information about resource availability
5. queries Data and Network Management Services
6. generates schedule for job
7. reserves allocation if possible
   otherwise selects another allocation
8. delegates job monitoring to Job Supervisor

Job Supervisor/Network and Data Management:
service, monitor and initiate allocation

Example:

40 resources of requested type are found.

12 resources are selected.

8 resources are available.

Network and data dependencies are detected.

Utility function is evaluated.

6th tried allocation is confirmed.

Data/network provided and
job is started

Re-Scheduling

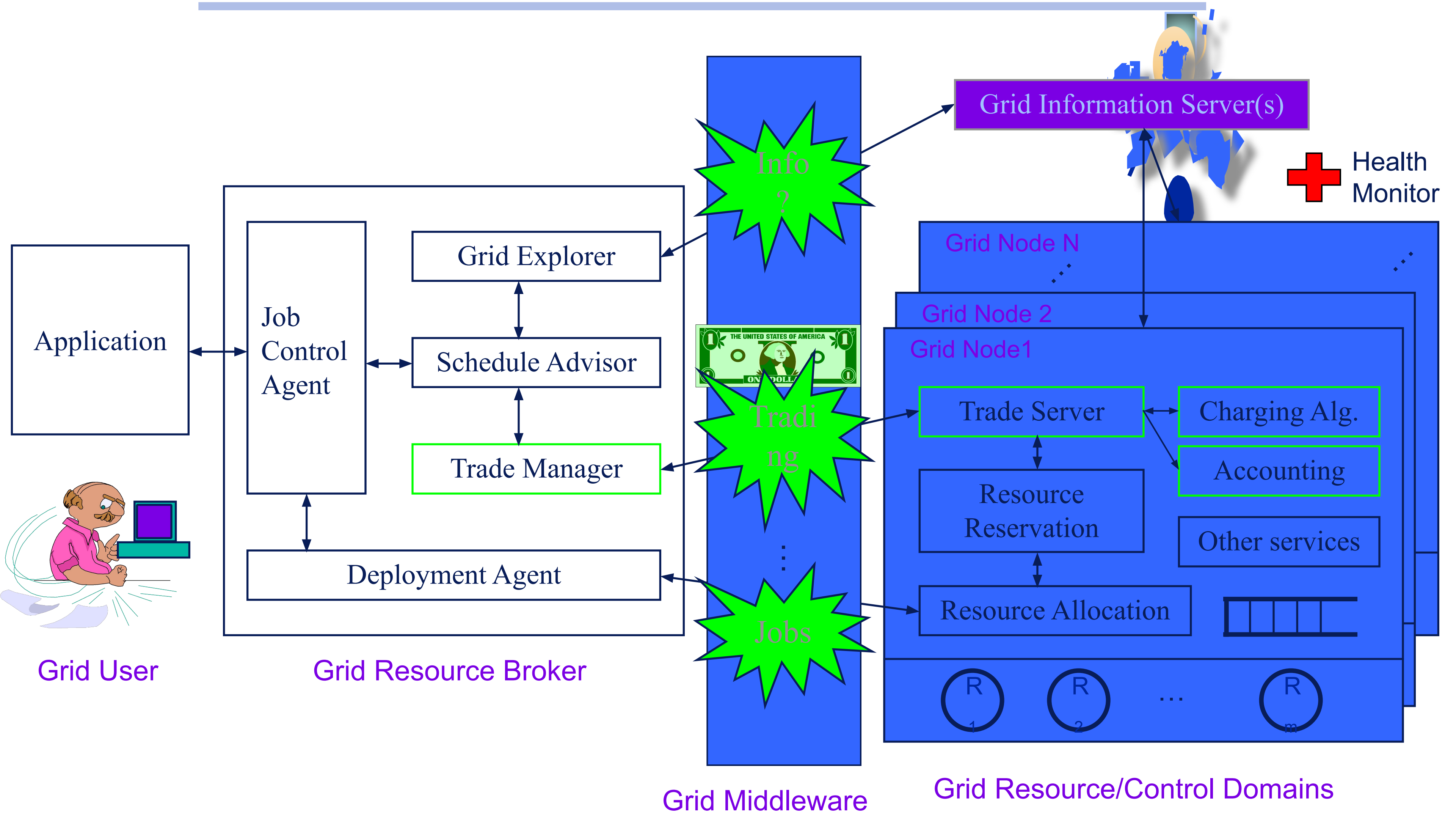
- Reconsidering a schedule with already made agreements may be a good idea from time to time
  - because resource situation may have changed, or
  - workload situation has changed
- Optimization of the schedule can only work with the bounds of made agreements and reservations
  - given guarantees must be observed
- The schedulers can try to maximize the utility values of the overall schedule
  - a Grid scheduler may negotiate with other resource providers in order to get better agreements; may cancel previous agreements
  - a local scheduler may optimize the local allocations to improve the schedule.

Computational Economy in Resource Management

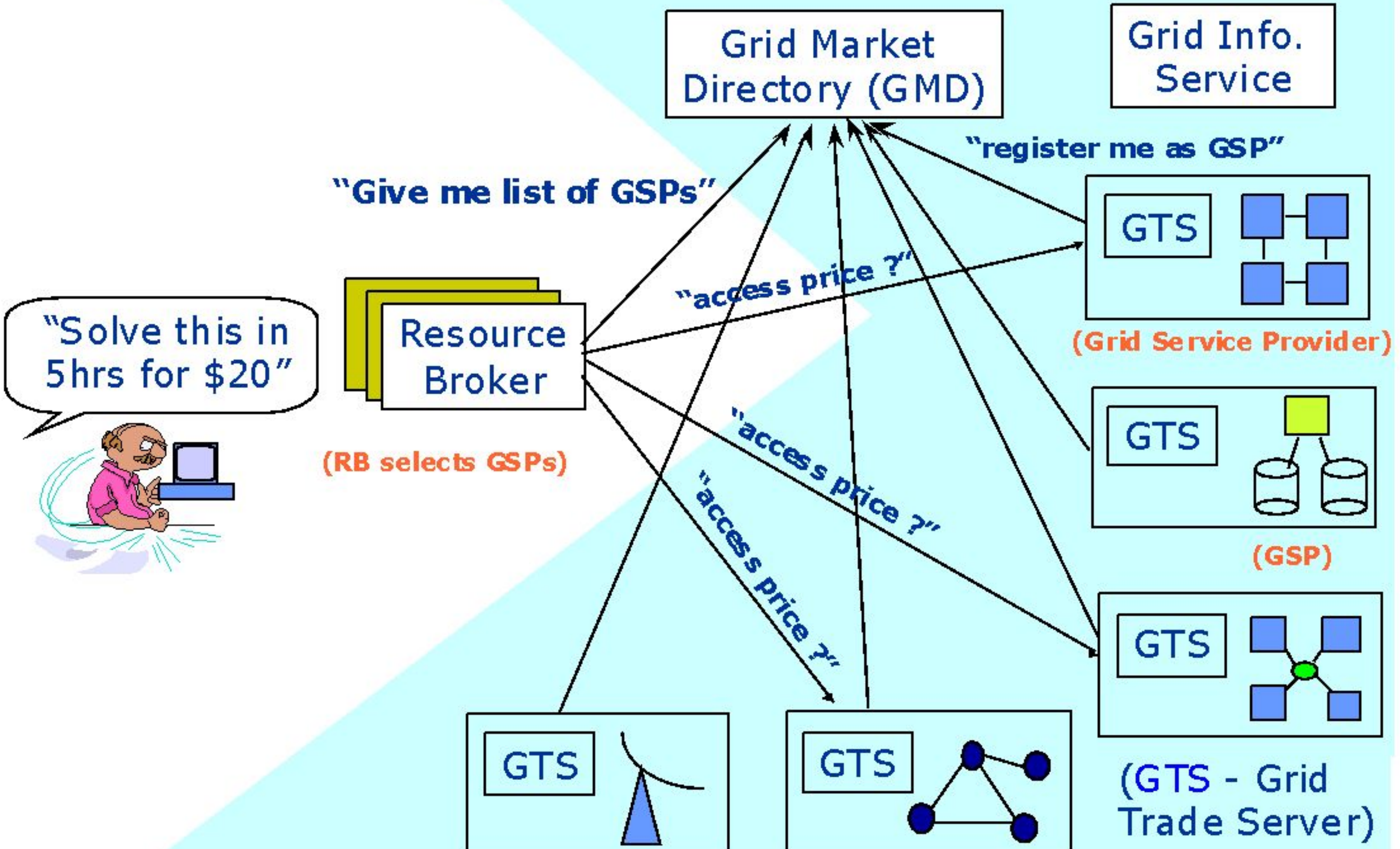
"Observe Grid characteristics and current resource management policies"

- Grid resources are not owned by user or single organisation.
- They have their own administrative policy
- Mismatch in resource demand and supply
  - overall resource demand may exceed supply.
- **Markets** are an effective institution in coordinating the activities of several entities.
- Traditional System-centric (performance matrix approaches does not suit in grid environment.
  - System-Centric --> User Centric
- Like in real life**, economic-based approach** is one of the best ways to regulate selection and scheduling on the grid as it captures user-intent.

Computational Market Model for Grid Resource Management



A Commodity Market Model



Conclusion

- Resource management and scheduling is a key service in an Next Generation Grid.
  - In a large Grid the user cannot handle this task.
  - Nor is the orchestration of resources a provider task.
- System integration is complex but vital.
  - The local systems must be enabled to interact with the Grid.
  - Providing sufficient information, expose services for negotiation
- Basic research is still required in this area.
  - No ready-to-implement solution is available.
  - New concepts are necessary.
- Current efforts provide the basic Grid infrastructure. Higher-level services as Grid scheduling are still lacking.
  - Future RMS systems will provide extensible negotiation interfaces
  - Grid scheduling will include coordination of different resources

References

- Book: “Grid Resource Management: State of the Art and Future Trends”,
  co-editors Jarek Nabrzyski, Jennifer M. Schopf, and Jan Weglarz, Kluwer Publishing, 2004

- PBS, PBS pro: www.openpbs.orgPBS, PBS pro: www.openpbs.org and www.pbspro.com
- LSF, CSF: www.platform.com
- Globus: www.globus.org
- Global Grid Forum: www.ggf.org, see SRM area