Basic Scheduling in Grid environment &Grid Scheduling Ontology

By: Shreyansh Vakil
CSE714 – Fall 2006 - **Dr. Russ Miller**.

Department of Computer Science and Engineering, SUNY Buffalo

What is Grid Computing??

- Middleware which allows people and organisations to share computing resources in a coordinated manner
- Data and computation can be distributed between machines in other institutions around the country/world
- Remote access to resources that are not available locally

Grid computing aims to be *seamless* and *secure*

- Users interact with “the grid”, instead of individual machines
- Different platforms integrated using standard protocols
- Support for a wide range of applications

Common ways in which a grid is used

- Execute a program on a remote host
- Read and write data stored on another machine
- Access services provided by a particular server

Types of grids

“Heavyweight” grids (Or Large Grids Or Global Grid)

- Consist of supercomputers and other high-end machines
- Generally used for computation sharing
- High level of security and autonomy
- Can be complex to set up and maintain

“Lightweight” grids (Or Medium Grids Or Enterprise Grid)

- Commodity desktop machines in homes and offices
- Generally used for computation sharing
- Usually based on master-worker model
- Software are much simpler, can be installed by end users

Service based grids (Or Small Grids Or Cluster Grid)

- Can consist of any type of machine
- Provides specific functionality rather than generic compute cycles
- Platform independent; implementation details hidden behind interface

What is Scheduling??

Need to decide when and where computation and other things are to occur

Traditional CPU scheduling

- Familiar case of task scheduling on a single CPU - time-slicing allocates CPU time to processes
- SMP systems - several CPUs - time slices are allocated between processors
- Parallel computers - many processors - tasks usually have exclusive access to a certain number of processors
- Clusters of workstations – similar case – tasks must be assigned to specific processors

Lots of past research already done for parallel computers and clusters

- Grid computing has some similarities to these
- However, many additional factors have to be considered

Grid Scheduling

Many extra complexities

Scheduling between multiple computers

- Different CPU speeds, architectures
- Network connectivity can vary widely between machines
- Many different users and concurrent tasks

Data location is important

- Data should be close to computation for efficient access
- Affected by network bandwidth and available storage resources

Centralised vs. distributed scheduling

- Centralised scheduling offers more control, but limits autonomy of individual resources
- Centralised algorithms only scale to a few hundred machines
- Distributed scheduling gives more control to machine owners, and is much more scalable

Grid Scheduling

Current scheduling strategies for grids are limited

- Some only support specific application models e.g. task farming
- Generic mechanisms do not usually support parallel programs
- Scheduling is generally based on independent jobs, or application-specific parallel scheduling
- Centralised schedulers - limited scalability

Three main types of scheduling

- Job submission
- Services
- Data placement (replica management)

These types of scheduling are normally independent of each other.

Job submission

- Similar to batch processing model used on mainframes
- Client supplies details of job to run, including program name, command-line arguments, environment variables
- In many cases, client also uploads executable to run
- Server runs job immediately or in the future, and notifies user on completion
- Useful for gaining access to faster computers on the grid to run your own programs
- Platform-dependent; client needs knowledge of server configuration, and if binary executable supplied, server must have specific OS/architecture

Common examples

- Globus GRAM
- Condor
- LSF
- PBS

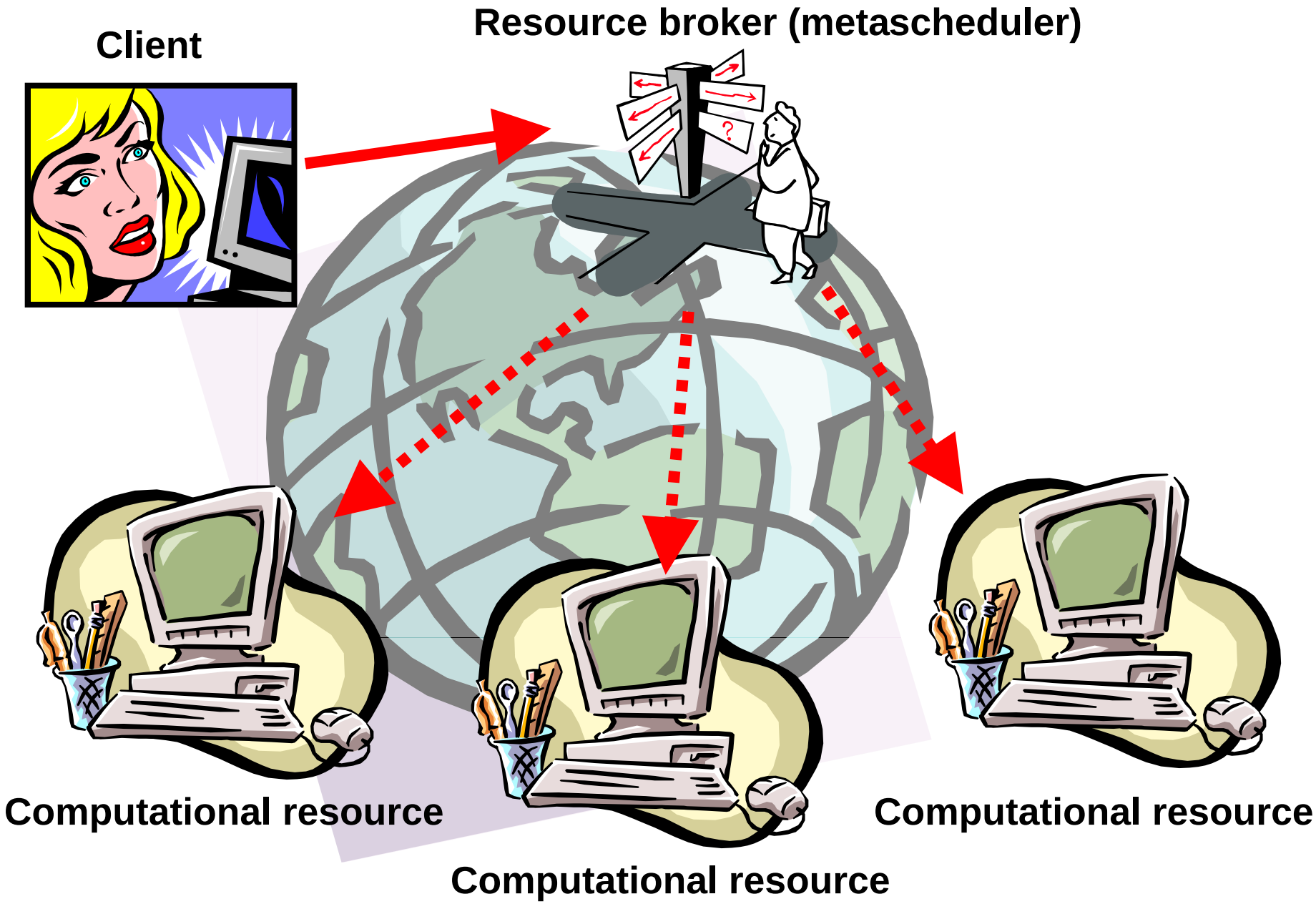
Job submission - Scheduling

Metascheduling (done at grid level)

- On which resource should the job run?
- Choice based on job requirements, access permissions, machine load and other parameters
- Parallel jobs can be scheduled to run across multiple resources
- Each resource may physically contain multiple processors – e.g. parallel computer or cluster
- Example – parameter sweep application. Each resource handles a particular range.

Local scheduling (done at individual resource level)

- Once job is assigned to a resource, when should it run?
- Usually determined by local job queuing system
- Job is run if machine is currently unused, or may be delayed until the other jobs have completed
- Alternatively, job may start straight away and run concurrently with any other existing jobs on that machine


Client
Resource broker (metascheduler)
Computational resource
Computational resource
Computational resource

Services

- Well-defined interface with a set of operations
- Accessed via standard protocol – no knowledge of platform necessary
- Different implementations of a service, all conforming to the same interface, can be accessed transparently by clients
- Set of services provided by a machine is generally fixed – clients can't supply code to execute as they can in the job submission model
- Additional services can only be installed/configured by machine owner

Uses

- Client-server app – access a single service
- Workflow application – access many services and coordinate flow of data between them

Common technologies

- Java RMI
- Web Services
- Sun RPC

Services - Scheduling

- If a service is provided by more than one machine, client can chose which to connect to
- Some machines may be more desirable than others, based on expected time to perform operations or transfer data
- Clients either obtain list of servers from a central registry, or have messages redirected by some intermediate entity

Example: Website mirroring

- Multiple web servers host copies of the same site
- Load balancer intercepts requests from clients and redirects to servers according to some scheduling algorithm
- Alternatively, multiple DNS entries for the same site – client selects a machine

Data scheduling and replication

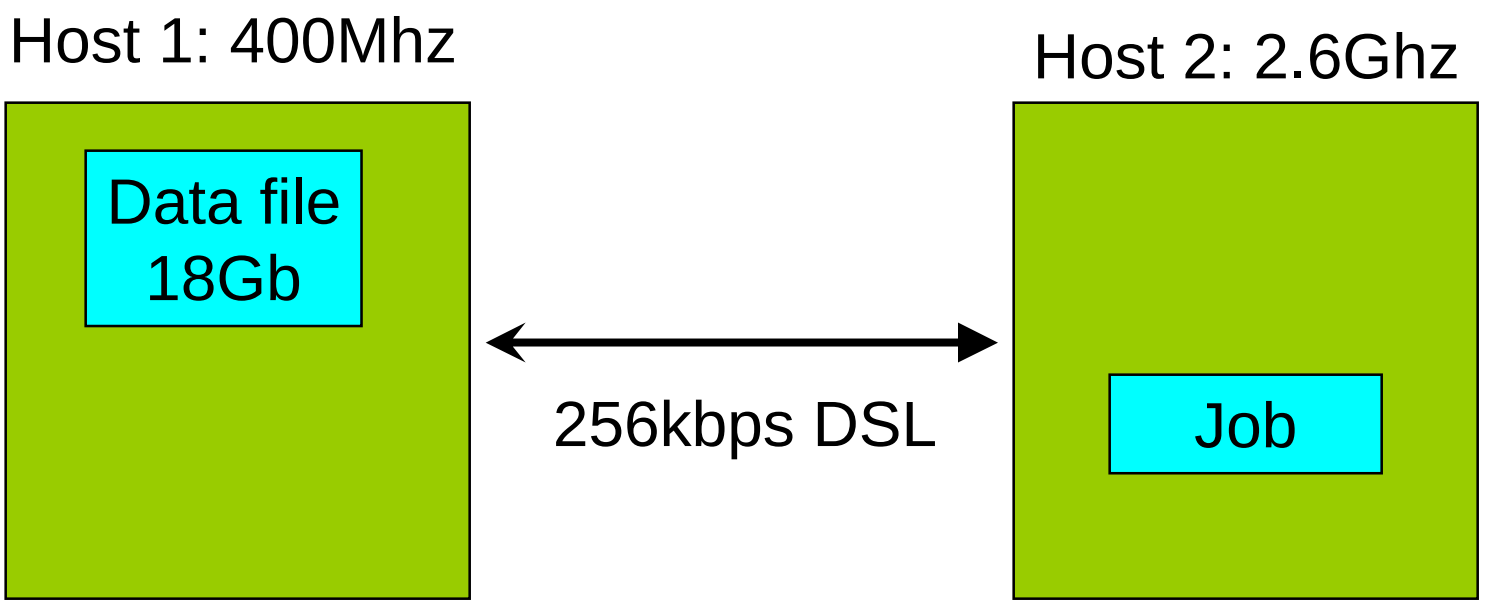
- The faster a program can access its data, the better
- Can move data to the program, or move program to the data
- Multiple copies (replicas) of the data can exist on different machines
- Programs access the “closest” replica (the one they have the fastest access to)
- Data from remote hosts may be cached locally

Knowledge about data access patterns can help with replication

- e.g. if many jobs access the same file, can pre-stage that file to remote machine(s) that will run the jobs

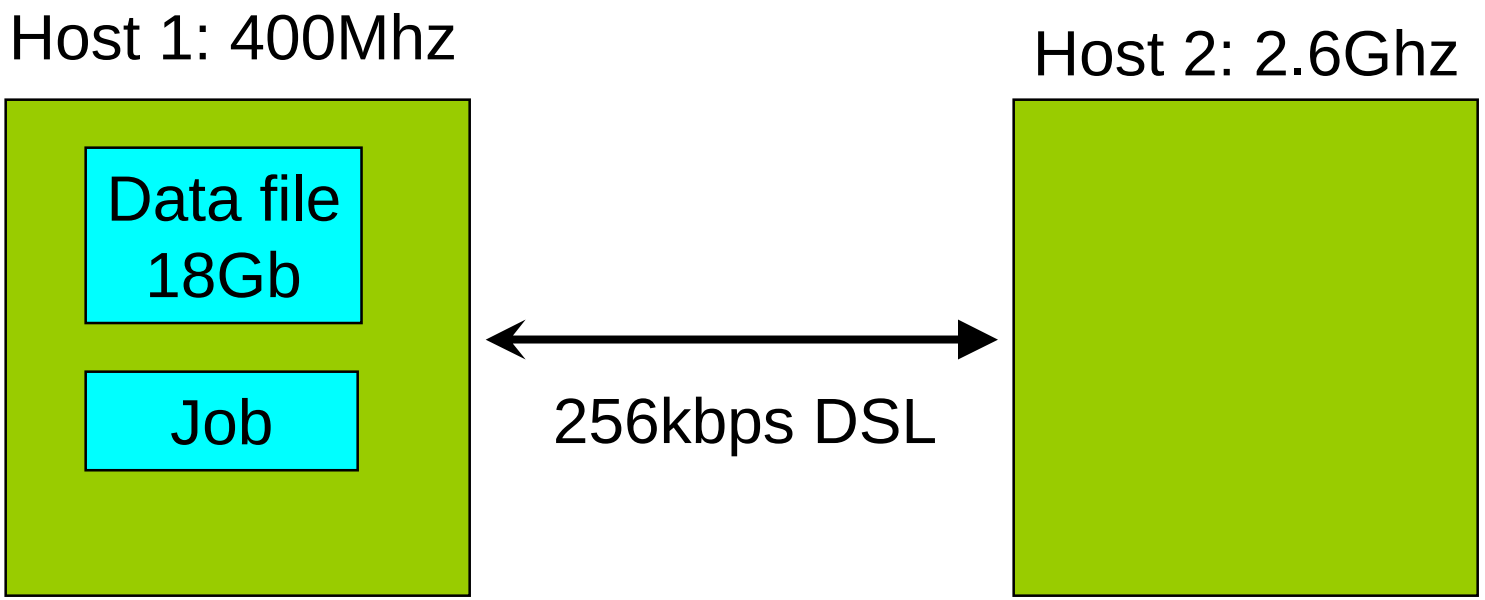
Data vs. Computation scheduling

- Scheduling decisions that choose a machine just on CPU speed/load could result in data access that is very slow.
- Is placing the job on Host 1 or Host 2 better?
- This depends on how much data it reads…



Data vs. Computation scheduling

- If the job access the entire file and does only a small amount of computation, it is better to run it on Host 1
- But if it only reads a few kb from different parts of the file, running the job on Host 2 would be quicker
- File could be pre-staged to Host 2 if it is reused by multiple jobs



Bringing Knowledge to Middleware---Grid Scheduling Ontology

Motivation

- In today's scenario, grid paradigm involves sharing of variety of resources across multiple administrative domains.
- Environment being dynamic, there is a need to abstract potential drawbacks away from resource users and resource providers
- Need of a sophisticated scheduling and resource management framework
- Scheduling HPC systems is already a challenge, but coordinated scheduling of multiple resources to automatically process a complex work-flow is next to impossible, if capabilities of resources are not known prior

- So, the need of Scheduling domain ontology

- Here, this method provides a common semantic understanding to be shared between the components involved in the scheduling process

- By integrating such an ontology, there is an increase in the automation level and administration of grids easier.

Other Advantages of this include:

- Seamlessness
- Resource autonomy and platform independence

What is Ontology??

- **Ontology** is a data model that represents a domain and is used to reason about the objects in that domain and the relations between them.

- Used in **Artificial intelligence** the semantic web, **software engineering and information architecture** as a form of knowledge representation about the world or some part of it.

Ontologies generally describe:

- **Individuals**: the basic or "ground level" objects
- **Classes**: sets, collections, or types of objects
- **Attributes:** properties, features, characteristics, or parameters that objects can have and share
- **Relations**: ways that objects can be related to one another

Environment & Requirements

To perform a scheduling task having several independent resources, knowledge and understanding of the environment is required on several levels like:

1. Single resources are described in a site-dependent way

2. Local scheduling systems provide different formats and interfaces to describe and manage resources

3. Meta-scheduler uses specific methods to map a user request to the requests addresses to different local scheduling systems

Human Involvement in scheduling could be either:

- Resource Requester (RR) Or
- Resource Provider (RP)

An entity could be a RR or RP at the same time

E.g. meta-scheduler (acts as a RR to underlying scheduler and take the RP role with respect to superordinated schedulers)

Requirement is to automise the mapping from RR space to RP and vice-versa and so reduce the manual intervention of users and providers in scheduling and RM process.

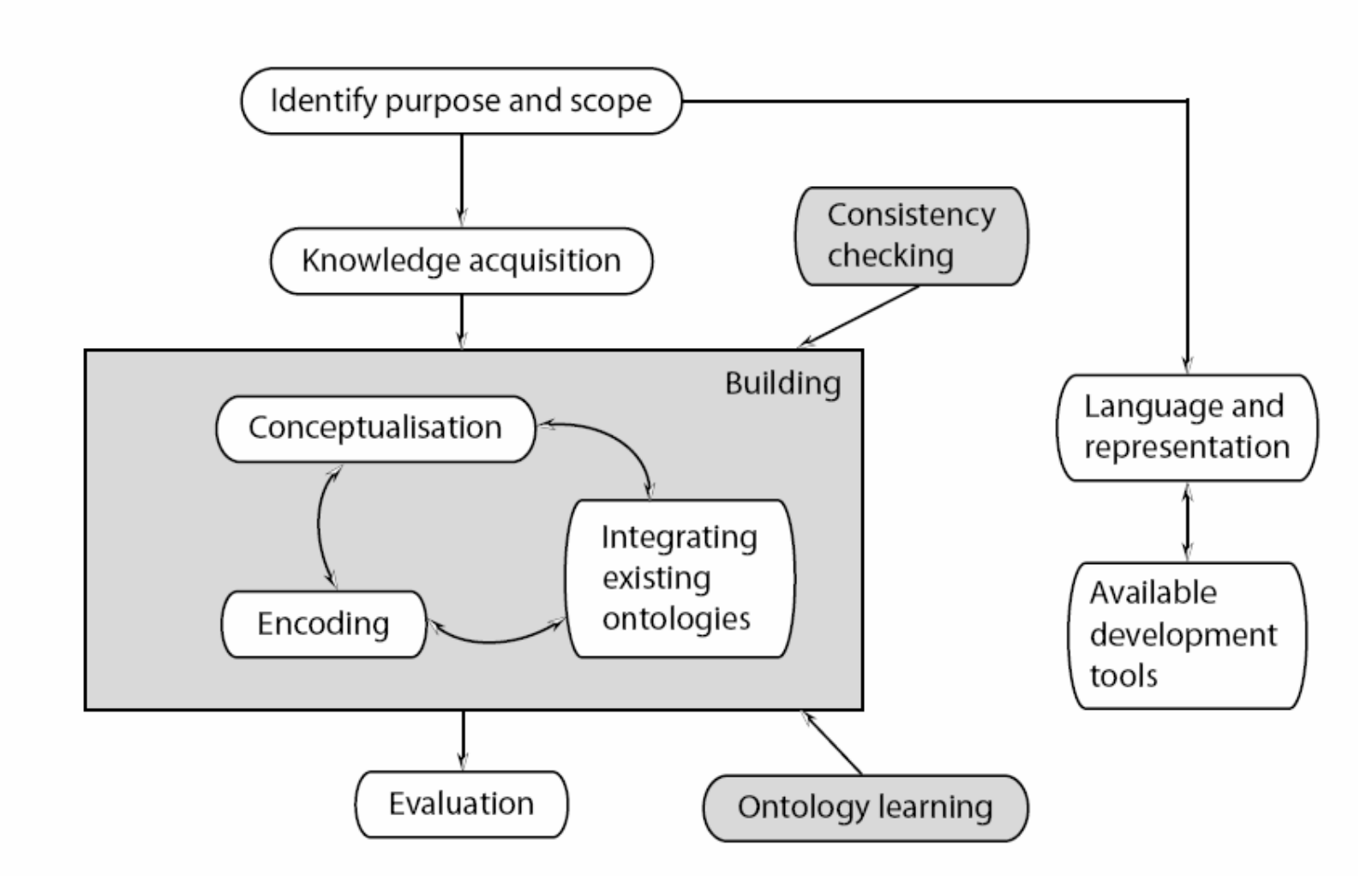
- Resource Ontology--Categorises and draws the relationships between the various ways the resources and services are described

- Scheduling domain ontology seems to be a promising way to introduce knowledge exploitable by machines into the scheduling process in grids

- Sharing such an ontology among resource brokers, resource management systems and schedulers would make the discovery of resources and their mapping of requests to resources less arbitrarily

- Negotiation between sites more distinct and may be carried out automatically by a meta-scheduler

Ontology Building Life-cycle



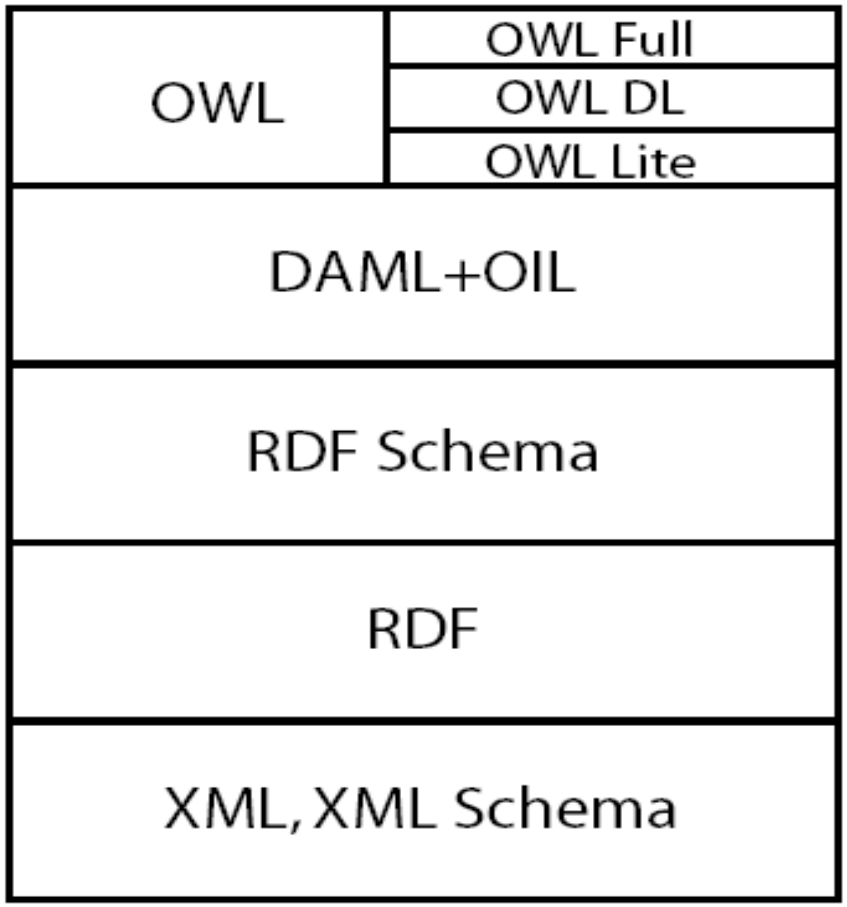
Knowledge Acquisition

- Grid Scheduling Dictionary (GSD) –an international document produced by the Grid Forum
- Purpose – Create a dictionary that has common terms and their definitions and used by various schedulers-local and global grid level.
- Purpose and scope of GSD makes it an ideal input to knowledge acquisition
- GSD has terms, their definite descriptions and the relations between them helps reducing the effort to transfer the domain-specific piece of knowledge from human readable to machine – processable form
- Also, command line parameters from interfaces and API's of available scheduling systems considered as input and potential new terms for adding to the dictionary

Semantic Markup Languages and Tools

- Machine processable ontologies created using semantic markup languages
- Unlike general markup language, the semantic markup language includes additional information attempting to encode the meaning of the content described using the markup language
- Next slide ---- Semantic Markup languages stack

Semantic Markup Languages Stack



Semantic Markup Languages Stack

- XML,XML schema – simplest of all markup languages
- RDF (Resource Description Framework) – document made of statements consisting of subject, predicate, object called triples and attribute value pairs
- RDF schema on top of RDF providing a formal definition of RDF adding classes and properties
- RDF resources – web pages, computer codes, data, hardware, algorithms, research group etc.
- DAML+OIL combination of DARPA Agent Markup Language (DAML) ontology language (ONT) and OIL which is Ontology inference layer for RDFS

Semantic Markup Languages Stack

- Our language of choice – OWL DL (Web Ontology Language – Description Logics)
- Based on Description Logics and therefore allows automated reasoning over an ontology.
- OWL Full – too rich language to allow this
- OWL Lite - quite simple to fulfill the requirements of Grid Scheduling Ontology
- Many tools dveloped supporting the creation and maintaining of complex DAML+OIL or OWL ontologies
- Eg: OntoMat, PC Pack, Protégé (preferable), etc…

Comparison between different approaches in Grid computing

- Resources and resource requests today are described using descriptive and constraint languages

Exact syntactic matching is done in Condor

Syntactic translation done in Globus

Database lookup and mapping done in UNICORE

- Advantage of these methods:

1. little overhead in building the systems

- Disadvantage:

1. The development of Resource Provider space and Resource Requester space descriptions must be synchronised.
2. The low flexibility to react on changes in RP or RR.

Comparison between different approaches in Grid computing

- In Using ontologies, definitely overhead in creating and maintaining ontologies

- But once published ,it might be used without additional effort in other environments too

- Advantages

1. The technology developed for the Semantic Web can be exploited.
2. The development of RP and RR space ontologies may happen independently.
3. The high flexibility to react on changes in RP or RR.

- Grid Scheduling Ontology aims to fill this gap and will allow negotiation between sites with different resources available under different scheduling policies to become more distinct and carried out automatically by a meta-scheduler.

Realisation

- Driving force for the development and usage of ontologies, languages and tools --- the idea of a Semantic Web.
- Originating from a Web (Service) environment ontologies are easy to integrate into the future versions of service oriented and WSRF-based Grid systems like upcoming implementations of UNICORE or the Globus Toolkit.
- When the meta-scheduler receives a request to schedule a job comprising multiple resources ,the meta-scheduler will start querying the individual local schedulers about their capabilities.
- Eg: If inference engine returns for example "NQS and OpenPBS are schedulers not capable of doing advance reservation" ,then the meta-scheduler decides that a remote scheduling system (controlling a resource needed for a job) that is exposed as "NQS" is not suitable to schedule a component of an application that has to run in parallel with other components using other resources.

Realisation

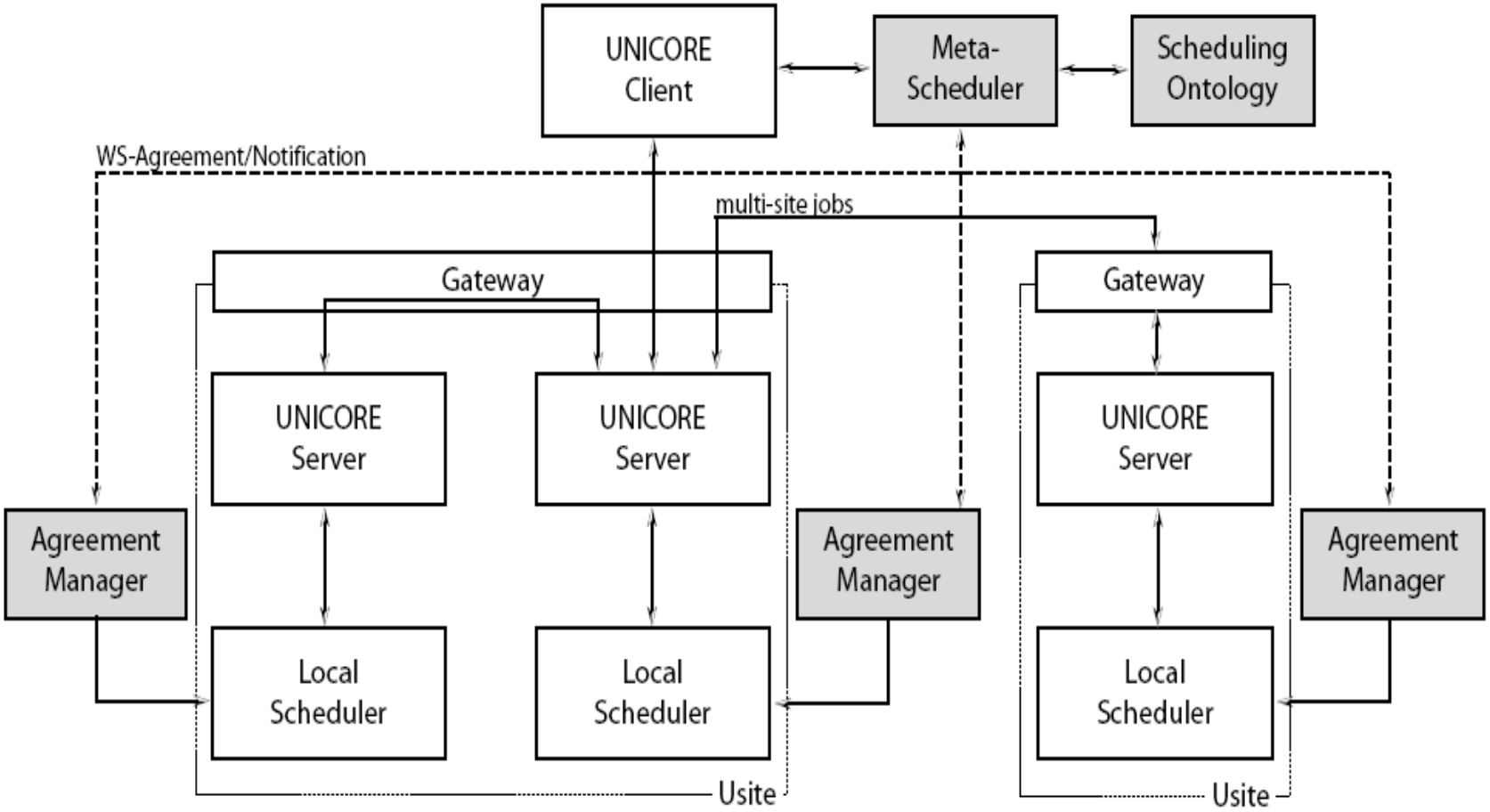
- The RACER inference engine will be used to find out for each scheduler to which class of schedulers it belongs and whether the scheduler has the necessary capabilities.

- Based on this knowledge the meta-scheduler starts negotiations with the appropriate local schedulers.

- The set of appropriate schedulers may be empty, of course, because none of the resources available has a local scheduler with the necessary capabilities,

e.g. advance reservation or interactive use of the resource.

Example Conceptualisation of the Ontology



Conclusion

- Introduced domain-knowledge into scheduling middleware making scheduling-specific parts of such knowledge exploitable encoded into a scheduling domain ontology.
- It provides a common semantic understanding to be shared between the components involved in the scheduling process
- It increases the automation level, and makes usage and administration of Grids easier.
- We identified purpose and scope of the ontology.
- We selected OWL DL as the appropriate language.
- We evaluated and selected the development tools to be used in the process.
- We finalised the initial knowledge acquisition.
- We saw with the conceptualisation of the ontology.

Future Work

- Lots need to be done
- Examine existing resource ontologies with respect to their re-usability.
- Need to evaluate Grid Scheduling Ontology's relationship to other ontologies which serve similar needs in a service-context
- Necessary to design and create the adapters of schedulers that cannot be modified and integrate the respective components in existing scheduling and RM systems

References

- [1] R. Menday and Ph.Wieder. GRIP: The Evolution of UNICORE towards a Service-Oriented Grid. In *Proc. of the 3rd Cracow Grid Workshop (CGW'03)*, Oct. 27-29, 2003.

- [2] C. Catlett, W. Johnston and I. Foster. *Global Grid Forum Structure*. Grid Forum Document GFD.2, Global Grid

  Forum, 2002.

- [3] VIOLA – Vertically Integrated Optical Testbed for Large Application in DFN. Project web site, 2005. Online: http://www.viola-testbed.de/.

References (contd..)

- [4] J. Schopf. Ten Actions when Grid Scheduling. In *Grid Resource Management* (J. Nabrzyski, J. Schopf and J. Weglarz, eds.), pages 15-23, Kluwer Academic Publishers, 2004.

- [5] J. Brooke, K. Garwood and C. Goble. Interoperability of Grid Resource Descriptions: A Semantic Approach. In *Proc. of the GGF 9 Semantic Grid Workshop*, 2003.

- [6] C. Goble and N. Shadbolt. *Ontologies and the Grid*. Tutorial held at GGF 4, 2002. Online: http://www.semanticgrid.org/presentations/ontologies-tutorial/.

THANK YOU!!!