Distributed Operating Systems

Computing Grids

dr inż. Adam Kozakiewicz

akozakie@ia.pw.edu.pl

Institute of Control and Information Engineering

Warsaw University of Technology

Computing Grids

- √ Problems with the definition of grids
- √ Properties of grids
- √ Virtual organization
- √ Files in a grid
- √ Standards in the world of grid development
  - ⋆ OGSA/OGSI
  - ⋆ GridRPC
  - ⋆ JSDL
- √ Examples of grid middleware
  - ⋆ NEOS
  - ⋆ *@home, BOINC
  - ⋆ Condor
  - ⋆ Unicore
  - ⋆ Globus Toolkit, LCG, gLite...
  - ⋆ Ninf, NetSolve

Problems with the Definition of Grids

Grid

- √ is a hardware and software infrastructure providing reliable, consistent, cheap and ubiquitous access to high performance computing resources.
- √ is a mechanism for controlled and coordinated sharing of resources in dynamic, scalable virtual organizations.
- √ is a collection of loosely coupled, geographically distributed, heterogenous computing resources.

Do not confuse the general definition with the (utopian?) idea of The Grid: a generalization of the idea of Internet, a single grid including most of the world's computing resources, reducing the problem of solving arbitrarily large problems to creation of a sufficiently large virtual organization.

Properties of Grids (1)

- √ Single log-on
- √ Automatic process scheduling and data transfers
- √ Heterogeneous but virtually homogenous environment
- √ Use of open standards in protocols and interfaces
- √ Straighforward adding and removing resources to/from the grid
- √ Scalability
- √ Fault resistance

Properties of Grids (2)

- √ Simple, easy to use interface
- √ Process migration
- √ Autonomy of computing centers
- √ High level of security
- √ Compatibility and stability
- √ Minimal constraints for the user

Virtual Organizations

How to solve the problem of granting access to thousands of resources to thousands of users?

Virtual organizaton
a set of people and organizations sharing grid resources for a common goal.

- √ access to resources is granted to virtual organizations instead of individual users,
- √ a user may be a member of more than one VO, in every project he must identify himself as a member of one of them.

Files in a Grid

Files in a grid:

- √ Local
  - ⋆ Temporary files, files from the user’s disk, etc.
  - ⋆ Submitted as a part of a computational task (job), recieved as the result, transmitted between subtasks
  - ⋆ Removed after use (except the user’s local files, of course)
- √ Mass storage
  - ⋆ Data stored directly in the grid
  - ⋆ Normally used to store large amounts of data – huge files
  - ⋆ Usually cannot be modified, allowed operations are only creation, reading and removal, making versioning much easier
  - ⋆ Federation and replication are possible

Replication and Federation

Federation

joining of several files in one logical file. Very useful if the amount of data to be stored is beyond the capabilities of even a large data storage center.

Replication

storing the same data in several locations in the grid with the guarantee that the copies are absolutely identical. Provides load balancing and performance increase.

Standards in the Grid Community

- √ 1999 – first meetings of interested parties, initially in the USA, later also in Europe and Japan,
- √ 2000 – beginning of an organization called *Global Grid Forum*, grouping representatives of scientific/academic institutions (mostly), commercial companies and other institutions interested in grid technologies, work on first standards starts,
- √ 2006 – GGF merges with EGA (*Enterprise Grid Alliance*), creating *Open Grid Forum* (OGF).

OGSA/OGSI

OGSA – *Open Grid Services Architecture*, is a standard describing the architecture of a grid system.

OGSI – *Open Grid Services Implementation* was supposed to be a specification for implementation of OGSA, but slow performance forced the OGF to scrap the idea of Grid Services and use WSRF instead.

Main assumption of OGSA – use of Web Services, implying:

- √ communication via HTTP
- √ remote function calls using SOAP
- √ interface specification using WSDL
- √ broad use of XML

OGSA – Outline of the Specification (1)

Infrastructure Services

Specification of methods for accessing resources, defines the set of technologies (SOAP, WSDL, etc.), naming system (addressing), basic security mechanisms, state preservation techniques, etc.

Execution Management Services

Mechanisms for job submission and monitoring. Discusses the problems of job-resource matchmaking, job preparation execution and monitoring. The most important part for an end user is the selection of JSDL as the standard job description language.

OGSA – Outline of the Specification (2)

Data Services

Mechanisms for data management in a grid. Discusses remote access, replication and federation of data, providing data for computing jobs, automatic data conversions, storing and accessing metadata.

Resource Management Services

Mechanisms for resource management, both on the basic hardware and software level and on the grid services level.

Security Services

Methods for securing the grid, including a detailed specification of a security model based on virtual organizations. Discusses problems of rights delegation, mapping of grid users to local accounts, etc.

OGSA – Outline of the Specification (3)

Self-Management Services

Methods providing adaptability of the grid: detection of malfunctions, automatic reorganization of the grid (bypassing the broken part), optimization of performance through monitoring of the performance of individual resources, etc.

Information Services

Collection and publication of all kinds of information about the grid – finding resources with given properties, monitoring their status, and so on.

OGSA – Layers

Layers of the Open Grid Services Architecture:

1. Fabric
2. Connectivity
3. Resource
4. Collective
5. Application

The application layer has direct access to connectivity, resource and collective layers.

GridRPC, JSDL

JSDL

Job description language based on XML.

- √ Job contents
- √ Requirements for the system executing this job
- √ Location of data
- √ Scheduler preferences
- √ No support for complex jobs (subjob graphs, parametric jobs)

GridRPC

API enabling distributed programming in a grid environment using the RPC paradigm. Standard supported by both mature solutions in this group (NetSolve and Ninf), but ambiguous, allowing slight differences in interpretation, resulting in incompatibility of both solutions and reduced portability of code (luckily the difference is really small and easy to fix).

Grid Middleware

Grid middleware examples:

- √ NEOS
- √ …@home solutions, including BOINC
- √ Condor
- √ Unicore
- √ Globus Toolkit and related middleware packages
- √ GridRPC solutions
  - ⋆ NetSolve
  - ⋆ Ninf

NEOS

NEOS – Network Enabled Optimization System – a simple system providing remote access to optimization problem solvers, with many properties of a grid. Can be adapted for other applications.

MetaNEOS, project of a *next-gen NEOS* is a lot more grid-like, but the project seems dormant, with very little development activity.

- √ Client
- √ Server
- √ Solver

*@home, BOINC

- √ Grids for very well decomposable tasks.
- √ SETI@home (Berkeley)
  - ⋆ the first popular program of this type
  - ⋆ 55 TFLOPS when the world's most powerful supercomputer was the Earth Simulator in Japan, with 40 TFLOPS
- √ BOINC
  - ⋆ new version of the SETI@home software
  - ⋆ much more universal – a complete middleware system, enabling users to create their own projects
  - ⋆ January 2008: 815 TFLOPS (compared to 478 TFLOPS of the fastest HPC system on TOP 500 in November 2007), including SETI@home 373 TFLOPS
- √ Folding@home (Stanford)

*@home, BOINC

- √ Grids for very well decomposable tasks.
- √ SETI@home (Berkeley)
  - ⋆ the first popular program of this type
  - ⋆ 55 TFLOPS when the world's most powerful supercomputer was the Earth Simulator in Japan, with 40 TFLOPS
- √ BOINC
  - ⋆ new version of the SETI@home software
  - ⋆ much more universal – a complete middleware system, enabling users to create their own projects
  - ⋆ January 2008: 815 TFLOPS (compared to 478 TFLOPS of the fastest HPC system on TOP 500 in November 2007), including SETI@home 373 TFLOPS
- √ Folding@home (Stanford)
  - ⋆ client software for ATI graphics cards (0.2% of clients, but 3.5% TFLOPS!)

*@home, BOINC

- √ Grids for very well decomposable tasks.
- √ SETI@home (Berkeley)
  - ⋆ the first popular program of this type
  - ⋆ 55 TFLOPS when the world's most powerful supercomputer was the Earth Simulator in Japan, with 40 TFLOPS
- √ BOINC
  - ⋆ new version of the SETI@home software
  - ⋆ much more universal – a complete middleware system, enabling users to create their own projects
  - ⋆ January 2008: 815 TFLOPS (compared to 478 TFLOPS of the fastest HPC system on TOP 500 in November 2007), including SETI@home 373 TFLOPS
- √ Folding@home (Stanford)
  - ⋆ client software for ATI graphics cards (0.2% of clients, but 3.5% TFLOPS!)
  - ⋆ client software for Playstation 3 (12.6% of clients, but 74% of TFLOPS!)

*@home, BOINC

- √ Grids for very well decomposable tasks.
- √ SETI@home (Berkeley)
  - ⋆ the first popular program of this type
  - ⋆ 55 TFLOPS when the world's most powerful supercomputer was the Earth Simulator in Japan, with 40 TFLOPS
- √ BOINC
  - ⋆ new version of the SETI@home software
  - ⋆ much more universal – a complete middleware system, enabling users to create their own projects
  - ⋆ January 2008: 815 TFLOPS (compared to 478 TFLOPS of the fastest HPC system on TOP 500 in November 2007), including SETI@home 373 TFLOPS
- √ Folding@home (Stanford)
  - ⋆ client software for ATI graphics cards (0.2% of clients, but 3.5% TFLOPS!)
  - ⋆ client software for Playstation 3 (12.6% of clients, but 74% of TFLOPS!)
  - ⋆ mid-January 2008: 1078 TFLOPS total!

Condor

- √ HTC system (*High Throughput Computing*), i.e. designed to process very many very long jobs, unlike HPC systems (*High Performance Computing*), designed to provide very large computing power to a few currently running jobs.
- √ Difficult to classify:
  - ⋆ Cluster-like batch job queuing system (like PBS)
  - ⋆ Volunteer grid using spare processing power (like BOINC)
  - ⋆ Large-scale grid (through *flocking*)
  - ⋆ Integration with other grid systems (*glide-in*, also used as a scheduler for Globus-based grids)
- √ A very good, universal scheduler, based on ClassAd:
  - ⋆ Resources report their properties and capabilities
  - ⋆ Submitted jobs include a list of requirements (architecture, operating system, software, etc.)
  - ⋆ Scheduler finds matches, assigns jobs only to capable machines

Properties of a Condor System (1)

- √ State snapshots and checkpoint-based migration (requires *linking* with Condor libraries)
- √ Remote system calls
  - ⋆ no need to have an account on the remote machine or explicitly make files remotely accessible
  - ⋆ a program started by Condor acts as if it was running on the machine, on which it was submitted, regardless of the actual machine running it.
  - ⋆ only available in the Standard Universe
- √ Programs require no changes to the source code, linking with Condor libraries is sufficient; even if this is also not possible, a program may still be run in Condor, albeit with reduced functionality
- √ Joining of machine pools, a.k.a. *flocking* – separate machine pools may transfer jobs to each other

Properties of a Condor System (2)

- √ Job queuing – a directed acyclic graph can be used to define relationship between subtasks
- √ Grid computation:
  - ⋆ can be used as a queue manager and scheduler for grids based on Globus Toolkit
  - ⋆ *glide-in* – remote job submission by Condor to a grid based on Globus Toolkit
- √ Local administrator's priority – the owner of the machine has absolute, automatically guaranteed priority over Condor users

Condor – Universes

Universe – a category of Condor jobs. The universe assigned to a job defines the subset of Condor functionality available to the job.

- √ Standard Universe – programs linked with Condor libraries, checkpointing and remote system calls are available
- √ Vanilla Universe – programs that cannot be linked with Condor libraries, including script jobs
- √ PVM Universe – programs using the PVM library
- √ MPI Universe – programs using the MPI library
- √ Java Universe – programs implemented in Java
- √ Globus Universe – Condor as interface to Globus Toolkit, jobs are translated and forwarded to the GT-based grid
- √ Scheduler Universe – internal environment, used by the Condor scheduler

UNICORE

- √ **Un**iform **I**nterface to **Co**mputing **Re**sources
- √ Developed in Germany in order to reduce the number of supercomputing centers
- √ Assumptions – the system will be used by experts in fields **other** than computers:
  - ⋆ graphical interface for job submission
  - ⋆ *plug-ins* for the interface providing access to applications installed on the supercomputers in a point&click way
  - ⋆ very portable client – therefore implemented in Java
- √ The system is OGSA-compliant (but not a full implementation of the standard)

Initial Requirements of the UNICORE Project

Requirements specified at the start of the project (1999):

- √ UNICORE must be compatible with all major HPC operating systems.
- √ UNICORE may not modify or require modification of the configuration of those systems.
- √ HPC systems must remain autonomous in granting access rights to users.
- √ System must guarantee secure transmission of data via the Internet.
- √ UNICORE must cooperate with the security and authentication mechanisms of the HPC systems.
- √ UNICORE must support existing networking technologies.
- √ UNICORE must be compatible with both UNIX and Windows systems.

UNICORE Functionality (1)

- √ **Job creation and submission** using a GUI, choosing the node on which the job should be executed. A job may consist of several subjobs, including jobs executed on other nodes – UNICORE ensures correct order of execution and transfer of intermediate results.
- √ **Job management** – monitoring the state of every job in a job group and removing jobs.
- √ **Data management** – for each task, UNICORE creates a temporary dataspace (called **Uspace**), and the user defines data sets:
  - ⋆ imported from other UNICORE nodes or from the client workstation,
  - ⋆ exported (final results),
  - ⋆ transferred to other Uspaces (between subjobs).

  data management operations are available as separate job, which can be included in the job group.

UNICORE Functionality (2)

- √ **Applications** – the system is focused on providing access to existing, preinstalled applications, which explains the need for plugins.
- √ **Data flow management** – acyclic directed graph of jobs, with conditional execution and repeated execution (a given number of times, or until a condition is satisfied.
- √ **Meta-computing** – UNICORE enables simultaneous use of several nodes by one application (e.g. MPI) using systems such as PBS Pro.
- √ **Single log-on** using X.509v3 certificates.
- √ **Support for legacy use patterns** – traditional batch processing still possible using scripts as UNICORE jobs.
- √ **Resource management** – fully decentralized, the user is informed about currently available resources and the client verifies resource requirements of the jobs.

UNICORE Layers (1)

The system is organized in layers:

- √ **Client** – user interface, the layer communicating directly with the user.
- √ **Usite** – *UNICORE Grid Site*, main administrative unit of the grid, created in every computing center, groups resources available within that center.
- √ **Vsite** – *Virtual Site*, a group of resources within a Usite.

UNICORE Layers (2)

Software layers:

- √ **Client**, user interface, submits jobs using SSL.
- √ **Gateway**, the entry point for a Usite, provides basic authentication of a user, lists existing Vsites and acts as a gateway in the following communication.
- √ **NJS**, *Network Job Supervisor*, provides full authentication and authorization, translates jobs for the target systems, submits them for execution, manages the execution (scheduler); also acts as a client when submitting subjobs specified to run on other Usites…pretty much does everything.
- √ **TSI**, *Target System Interface*, inserts jobs as tasks in the local batch processing system and handles file transfers (the only part of UNICORE not written in Java, reference implementation in Perl).

Globus *et consortes*

Globus Toolkit

probably the most important middleware solution.

Versions:

1. of purely historical value
2. breakthrough – popular middleware which formed the basis for several very large scale projects (e.g. LCG), based on an original, successful architecture
3. official reference implementation of the original OGSA/OGSI – a flop, never became really popular due to abysmal performance
4. reference implementation of the current OGSA/WSRF

EDG, LCG, gLite

Functionality extensions for the Globus Toolkit designed for the LCG/EGEE project, provide things like replica handling, extended scheduling, etc. Current stable release is gLite 3.

Main Building Blocks of the Globus Toolkit

Security Module – an advanced security infrastructure, providing encrypted transmission, authentication and authorization with single log-on, based on PKI (X509v3 certificates).

Data Management Module – file transfers and a basic toolkit for handling replicas.

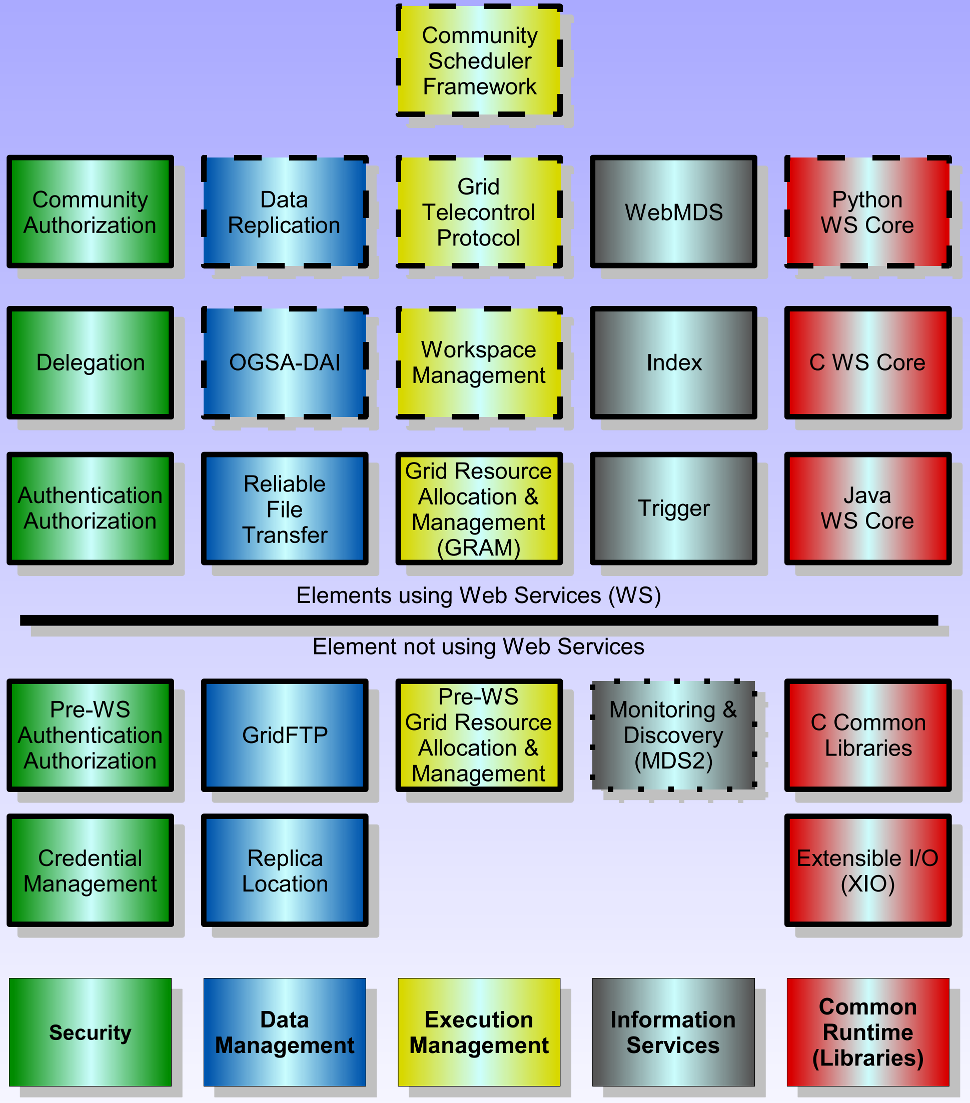
Execution Management Module – submitting, queuing and execution of jobs.

Information Services Module – provides information about resources available on each node and their current state, also includes modules for finding nodes given a set of constraints.

Base Libraries – libraries used during implementation of programs directly interfacing with the Globus Toolkit, including not only grid libraries, but also portable implementation of many standard libraries.

Globus 2 – popular protocols with some extensions (LDAP, FTP/GridFTP)
Globus 4 – Web Services (SOAP, HTTP) + backwards compatibility

Elements ot the Globus Toolkit



Community Scheduler Framework
Community Authorization
Data Replication
Grid Telecontrol Protocol
WebMDS
Python WS Core
Delegation
OGSA-DAI
Workspace Management
Index
C WS Core
Authentication Authorization
Reliable File Transfer
Grid Resource Allocation & Management (GRAM)
Trigger
Java WS Core
Elements using Web Services (WS)
Element not using Web Services
Pre-WS Authentication Authorization
GridFTP
Pre-WS Grid Resource Allocation & Management
Monitoring & Discovery (MDS2)
C Common Libraries
Credential Management
Replica Location
Extensible I/O (XIO)
Security
Data Management
Execution Management
Information Services
Common Runtime (Libraries)

Globus Toolkit – Additional Information

- √ GT may be used not only for building large, complex grids, but also as a communication infrastructure for a simple grid (e.g. MPICH-G2); unfortunately it's very **SLOW**.
- √ Job description languages:
  - ⋆ *RSL, Resource Specification Language* – Globus 2
  - ⋆ *XML Job Description* – Globus 4
  - ⋆ *JDL, Job Description Language* – EDG
  - ⋆ *JSDL*, still in the works
- √ Data model in a gLite grid – see lecture about naming

Ninf and NetSolve

- √ GridRPC implementations
- √ Ninf – a simple overlay for Globus Toolkit
- √ NetSolve – older system, much more functionality and complexity, can use Globus or work independently, three (!) different IDLs:
  - ⋆ interface configuration file, very difficult to write, but gives very fast programs
  - ⋆ NetSolve IDL – a later addition, easier to use, but not as good
  - ⋆ GridRPC – newest option
- √ Both solutions are not fully mutually compatible.

Ninf i NetSolve – Differences

Ninf	**NetSolve**
Globus Toolkit overlay	standalone solution, can use Globus Toolkit as an option
RPC only	RPC and a big selection of libraries
a simple, minimalistic solution – Globus does most of the work	a complicated, heavy system with many capabilities
a simple, skeleton server, Globus acts as the agent	complicated server and agent software
works with C and Java	works with Matlab, Octave, Mathematica, C and Fortran 77
multidimentional arrays can be sent directly	multidimentional arrays must be packed into one-dimentional ones before sending
not MT-Safe (explicit synchronization necessary in multithreaded programs)	MT-Safe

Most Important European Grid Projects

- √ Grid for LHC (*Large Hadron Collider*):
  - ⋆ LCG, *LHC Computing Grid*
  - ⋆ EGEE, *Enabling Grids for e-Europe* and EGEE-II – successors
  - ⋆ GEANT, DataGrid (EDG), ...– helpers
  - ⋆ huge amounts of data: 40TB per day, 15PB per year
- √ EuroGRID – UNICORE based grids (BioGRID, MeteoGRID, ...)
- √ GRIP – tool development – integration of UNICORE and Globus Toolkit.
- √ Main grid centers in Poland (former members of CrossGRID):
  - ⋆ ICM, Warszawa – Interdyscyplinarne Centrum Modelowania Matematycznego i Komputerowego Uniwersytetu Warszawskiego,
  - ⋆ PSNC, Poznań – Poznańskie Centrum Superkomputerowo Sieciowe (acronym: Poznań Supercomputing and Networking Center),
  - ⋆ Cyfronet, Kraków – Akademickie Centrum Komputerowe Cyfronet AGH.