Implementations of the Grid Architecture

John Watt

http://csperkins.org/teaching/2004-2005/gc5/

UNIVERSITY of GLASGOW

Overview

- Message Passing Interface
- The Globus Toolkit
  - The Globus Alliance
  - Globus Toolkit 3
  - GT3 Architecture
  - Resource Management
- Other Grid Technologies
  - Microsoft .NET
  - Apple X-Grid

Status

- OGSA does not exist!
  - Specification is not fully defined
- However...
  - Some components are available and can be used to build Grid Services
- But before Globus…
  - A short recap of Parallel Computing, and how we can build a Grid-ready application

Parallel Computing

- Researchers require computing resources to solve increasingly complex problems
- Demand for higher and higher processing speed, and more and more memory from normal desktop PCs.
  - Expensive!
- Distributed parallel applications provide a cheap, if complex solution
  - Requires new parallel applications, or the parallelisation of existing sequential programs.

Parallel Programs

- Definition:
  - A parallel program runs on multiple CPUs simultaneously.

- If we can rewrite our program so that it runs on many processors at the same time, our computation time can be substantially reduced.

  - How can we do this….?

Hardware Configurations

- Massively parallel machines
  - Single Instruction Multiple Data
    - Very large array of CPUs with one instruction unit which issues instructions to each CPU with its own data.
    - No longer manufactured (PC prices much lower)
- Shared Memory Processors or Symmetric Multi-Processor (SMP) machines
  - Individual computers with 2 or more CPUs
    - Can achieve parallelism by running single program across the processors
    - Need special techniques when more than one CPU attempts to access the same data in memory (i.e. all interaction between processes is through SHARED MEMORY)

Hardware Configurations

- Distributed clusters of machines
  - Each node in the cluster is an autonomous computer with its own operating system, memory and storage.
  - Two programming models:
  - Single Program Multiple Data (SPMD)
    - Same program runs everywhere on different sets of data
    - Loosely synchronous
  - Multiple Instruction Multiple Data (MIMD)
    - Different programs executed on different nodes on different sets of data
    - Asynchronous
  - Interactions performed by “Message Passing”

Message Passing

- Definition:
  - Message Passing is the process of sending data from a program running on one of the nodes, to a program running on one of the other nodes.

- All interaction between processes is achieved through an explicit exchange of messages.
  - Recap: processes can interact through either:
    - Message passing, or
    - Shared memory
      - What can we use for the Grid??

Shared Memory

- Standard:
  - OpenMP
    - First standard for shared memory parallelism
      - Previous standard (X3H5) never completed!
    - Specification for a set of compiler directives, library routines, and environment variables that specify shared memory parallelism.
    - Geared towards tightly coded applications running on systems with globally addressable and cache coherent distributed memories.
    - Designed for FORTRAN and C/C++
  - Not a standard that can be used to great effect on the Grid.

Message Passing

- Standard:
  - Message Passing Interface (MPI)
    - A specification of a message passing library
    - First message passing interface standard (MPI Forum, 1992)
      - Sixty people from 40 different organisations
      - Two years of proposals meetings and reviews
    - Interface specifications for C and FORTRAN with Java binding being worked on
    - Allows for efficient implementation, portable source code and support for heterogeneous parallel architectures.

Messages

- Messages between processes are simply packets of data with the following attributes:
  - Name of the sending process
  - Source location
  - Data type
  - Data size
  - Name of the receiving process
  - Destination location
  - Receive buffer size

Point-to-point communication

- Simplest form of message passing
  - One process sends a message to another
- Synchronous sends
  - Provides information about the message completion
    - You know they got it, but you may have to wait (e.g. beep from fax)
- Asynchronous sends
  - You only know that the message was sent
    - You don't have to wait, but you don't know if they got it. (posting a letter)

“I am process X” example

```
#include <mpi.h>
#include <stdio.h>
main (int argc, char **argv)
{
   int size,rank;
   MPI_Init(&argc,&argv);
   MPI_Comm_size(MPI_COMM_WORLD,&size);
   MPI_Comm_rank(MPI_COMM_WORLD,&rank);

   printf(“Hello, I am process %d of %d.\n”,rank,size);

   MPI_Finalize();
   exit(0);
}
```

MPICH-G2 – Globus compatible MPI library

- Need more than just this though!

What about OGSA?

- Message Passing
  - Good for building applications that can run simultaneously across the grid.

- But we need a Grid Infrastructure for it to run on!
- Have heard about the Open Grid Services Architecture in the previous lecture…
  - Does it exist in real life?
  - No, but yes. (?!)

The Globus Project

- Established 1995
  - U.S. Argonne National Laboratory
  - University of Southern California/Information Sciences Institute (USC/ISI)
  - University of Chicago

- Consortium dedicated to collaborative design, development, testing and support of the Globus Toolkit.

The Globus Alliance

- The Globus Project became the Globus Alliance in 2003.
  - New members form international consortium:
    - Swedish Centre for Parallel Computers (PDC)
    - University of Edinburgh Parallel Computing Centre (EPCC)
  - Includes Academic Affiliates program with participation from Asia-Pacific, Europe and US
  - US federal sponsorship:
    - NASA, Department of Energy, National Science Foundation, Defense Advanced Research Projects Agency
  - Industry sponsorship:
    - IBM, Microsoft Research

The Globus Toolkit

- An open architecture, open source set of software services and libraries that support computational grids.
- Components can be used independently, or together to develop useful grid applications.

"the de facto standard for grid computing"

New York Times

Globus Toolkit timeline

- GT1 – 1998
  - GRAM and MDS
- GT2 - 2001
  - GridFTP, Packaging (GPT)
- GT3 – 2002 (deployment June 2003)
  - Implementation of OGSA
- GT4 – soon!
  - Implementation of WSRF specification
    - Available in development release

Globus Toolkit 2 (GT2)

"100 most significant technical products of 2002"

R&D Magazine

- Current stable release GT2.4.3 will be used for the UK e-Science National Grid Service (online 2005)

- GT2 is still available
  - As a separate release (Most recent 2.4.4)
  - Encapsulated as the "Pre-WS/OGSA" components of GT3
    - The future is web services

Globus Toolkit 3

- Globus Toolkit 3 is an useable implementation of the Open Grid Services Infrastructure (OGSI)
  - Remember OGSI is a formal technical specification of the Grid Services defined in the Open Grid Services Architecture (OGSA)
- So, GT3 is an implementation of the OGSA framework

GT3 v GT2

- GT2 and GT3 both provide a set of Grid services for security, resource management, information access, and data management
- GT3 provides the same services as GT2, as well as extensions to the Globus protocol suite and grid service development tools.
- GT2 was designed before OGSA/OGSI, whereas GT3 is OGSI-compliant
- The primary benefit of using GT3 over GT2 is that GT3 implements standards that are being adopted by the e-Science and e-Business communities

The Three Pillars

Resource Management

Information Services

Data Management

----------Grid Security Infrastructure (GSI)---------

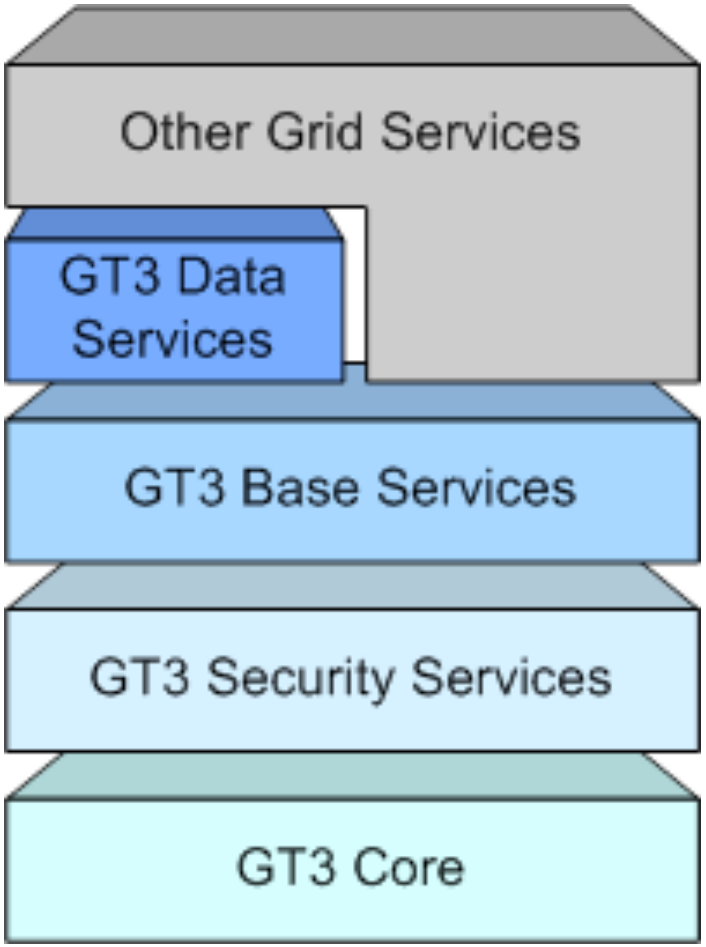
The Globus Toolkit

Pillar Implementation

- Resource Management (this lecture)
  - Globus Resource Allocation Manager (GRAM)
  - Managed Job Service in GT3
- Information Services (in Lecture 5)
  - Metacomputing Directory Service (MDS)
  - Index Service in GT3
- Data Management (in Lecture 12)
  - GridFTP
  - Reliable File Transfer (RFT) in GT3

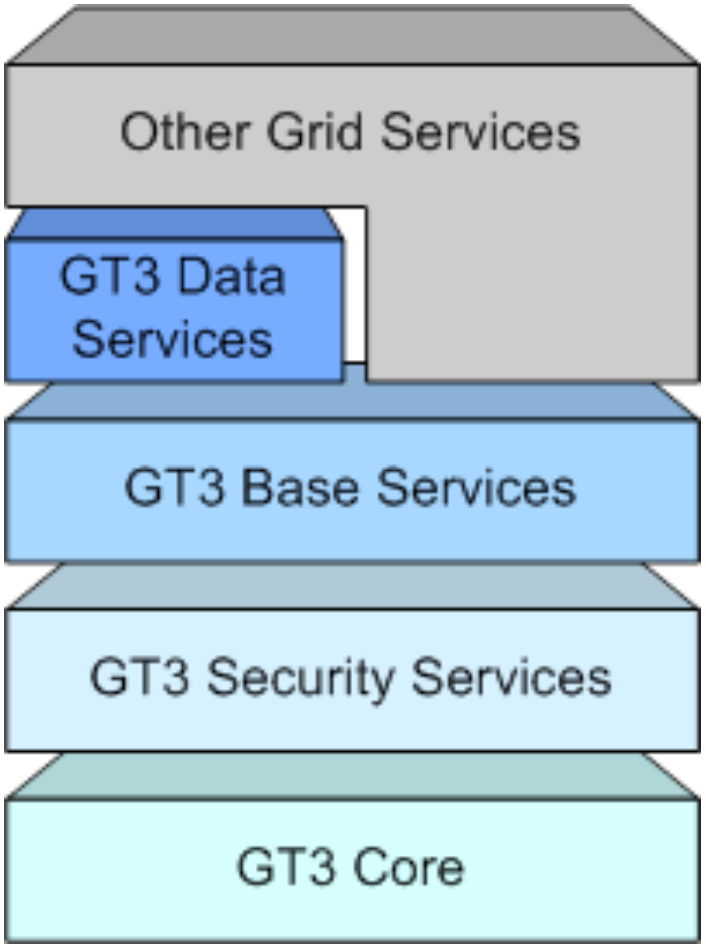
- All using the Grid Security Infrastructure (GSI) at the connection layer (Lecture 9).

GT3 Architecture



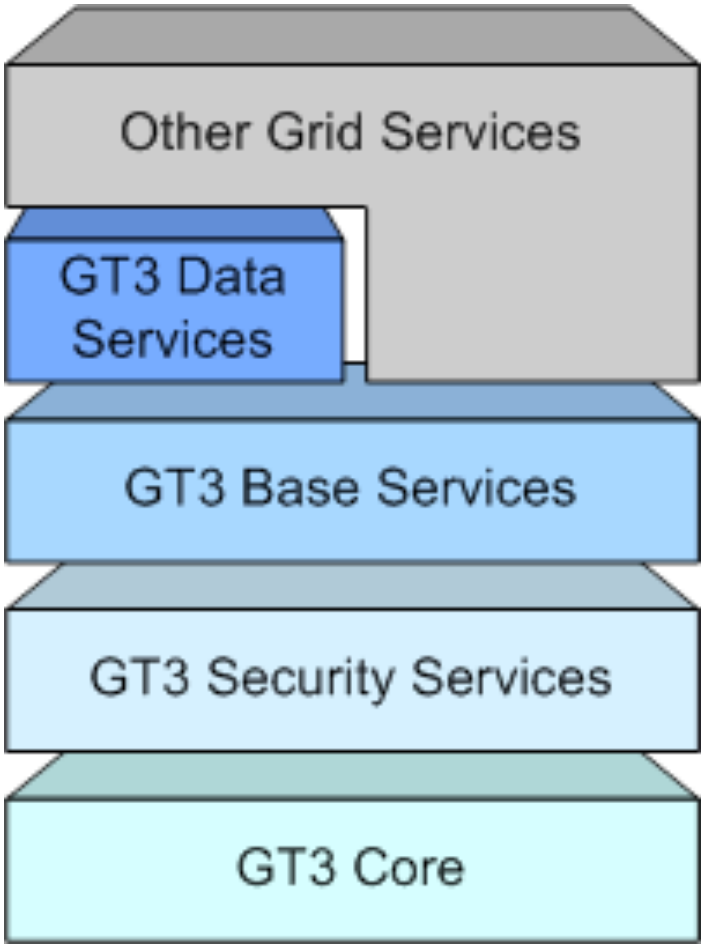
- GT3 Core
  - Grid Service Implementation of OGSI v.1.0
    - (to follow in Lecture 6)
  - Common APIs
    - Notification (Source, Sink, Subscription)
    - Registration, Factory, Handle Resolver
    - State management
  - Container Framework (portability across platforms)
  - Development and Runtime Environment
    - For building new Grid Services

GT3 Architecture



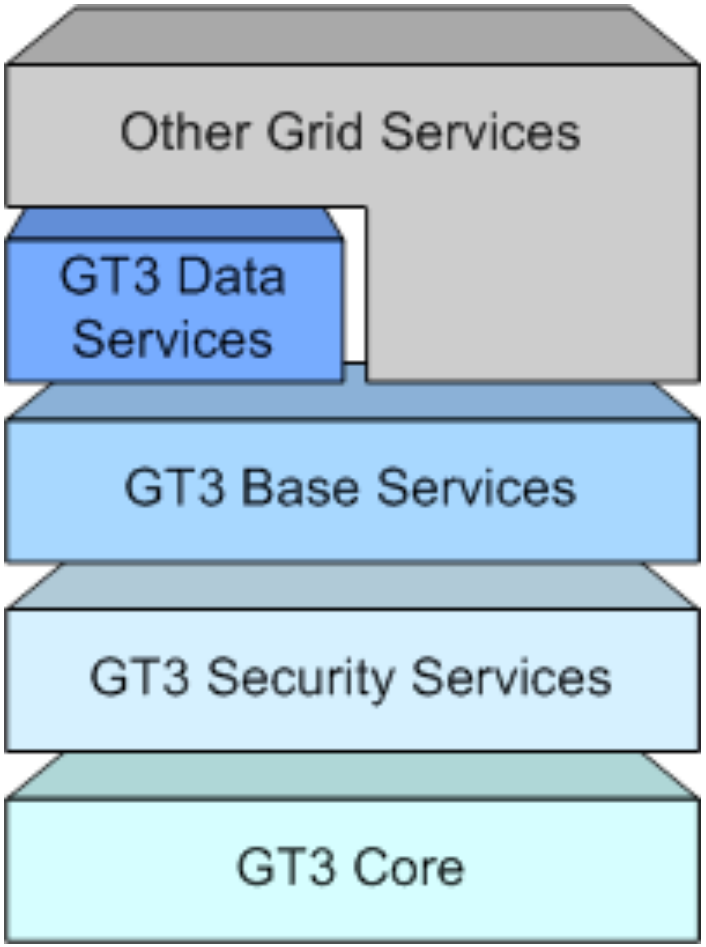
- GT3 Security Services
  - New Transport Layer/SSL protocol called "httpg" to indicate it's a GSI-enabled http-based protocol
  - SOAP layer security
    - XML Encryption
    - XML Signature
    - WS-Security
  - Use of X509 certificates for authentication
  - Use of X509 Proxy certificates for single sign on
  - Improved security model
    - Reduces amount of privileged code needed by a service
    - Easier to set up Gt3 behind a firewall
- Will be covered in detail in Lecture 9….

GT3 Architecture



- GT3 Base Services
  - The “pillars” we have talked about
  - GRAM (Managed Job service)
    - End of todays lecture…
    - Want to check progress and have control over jobs
  - Index Service (see Lecture 5 on Monday)
    - Finding Grid Services out there which will work best for YOU
  - RFT (Reliable File Transfer)
    - Will be introduced in Lecture 12
    - Allows large file transfers to occur between the client and the Grid Servcie

GT3 Architecture



- GT3 Data Services
  - Contains several non-OGSI (yet) compliant services
  - GridFTP (used by Reliable File Transfer service)
  - Replica Location Service (RLS)
    - Distibuted registry service that records the locations of data copies and allows discovery of replicas
    - Designed and implemented in collaboration with Globus and DataGrid projects
    - Handy for applications that deal with large sets of data.
    - We usually don't want to download the whole thing, just a subset.
    - Replica Management keeps track of these subsets for us
- Other Grid Services
  - Where non-GT3 services run….

GRAM Requirements

- Given the specifications of a job, we want to provide a service which can
  - Create an environment for the job to run in
  - Stage any files to/from the job environment
  - Submit the job to a local scheduler
  - Monitor the job
  - Send notifications about the state of the job
  - Stream the job's stdout/err during execution

Pre-WS GRAM Implementation

- Resource Specification Language (RSL)
  - Used to communicate job requests
- Non-OGSI compliant services
  - Gatekeeper
  - Jobmanager
    - Remote jobs run under local users accounts
    - Client to service credential delegation done through a third party (the gatekeeper)

Resource Management

- Three main components to the Pre-WS Globus resource management system
  - Resource Specification Language (RSL)
    - Method of exchanging info about resource requirements
  - Globus Resource Allocation Manager (GRAM)
    - Standard interface to all the local resource management tools
  - Dynamically-Updated Request Online Coallocator (DUROC)
    - Coordinates single job requests which may span multiple GRAMs

GT3 GRAM Implementation

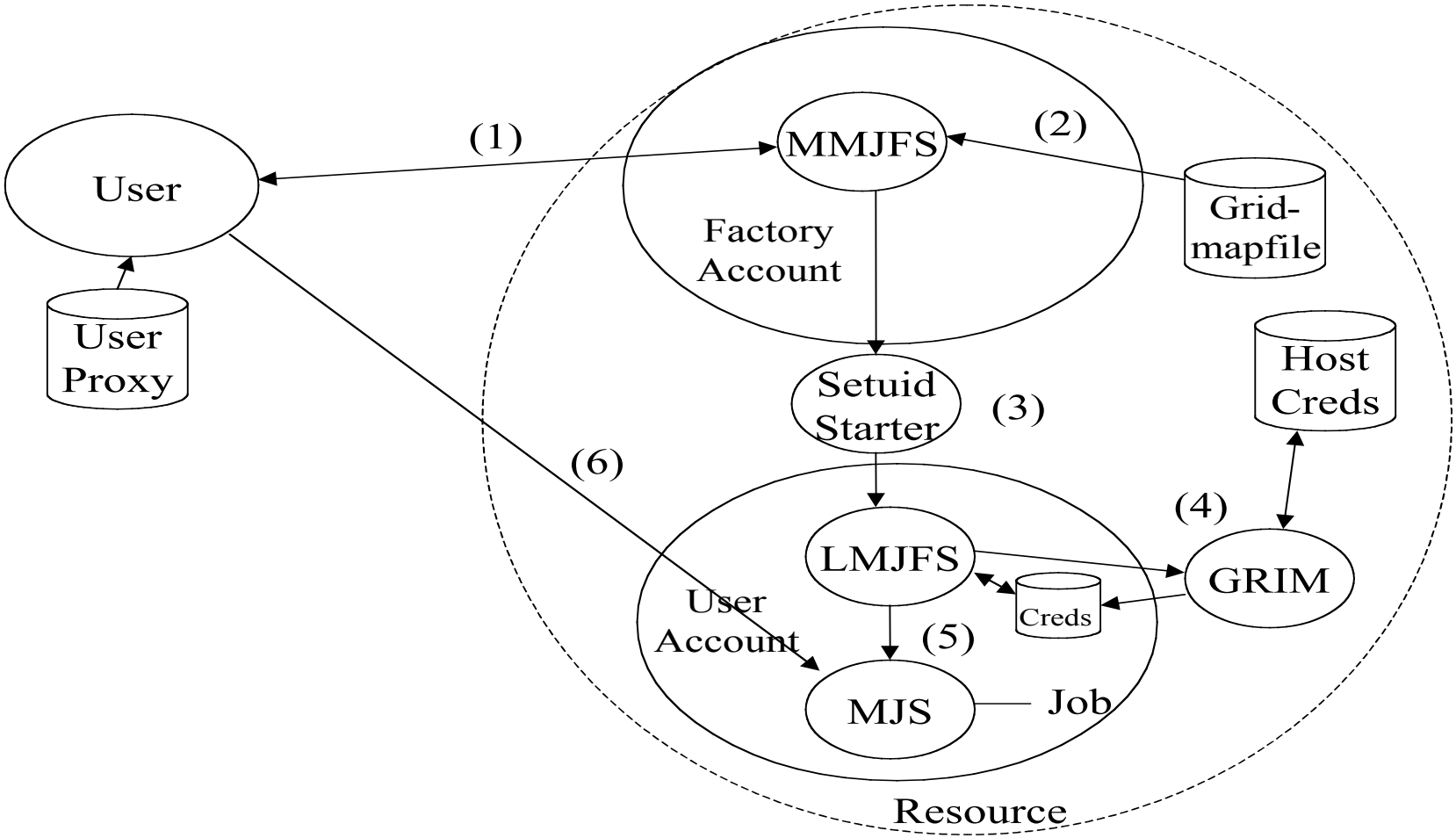
- Resource Specification Language
  - Communicates requirements (RSL-2 schema)
- Resource management services
  - (Master) Managed Job Factory Service (MMJFS)
  - Managed Job Service (MJS)
    - And..
  - File Stream Factory Service (FSFS)
  - File Stream Service (FSS)

    - Remote jobs run under local users accounts
    - Client to service credential delegation done user to user, not throug a third party)

Job Submission

- In GT3, job submission is based on the Grid Service Factory model
  - Create service
    - Service instance created, request validated
    - User's job request is ready to execute
  - Start Operation
    - User's job request starts
    - The service instance monitors the job request
    - Updates the request Service Data Element(s)
  - Job Control
    - Ensures client received a handle to the job before the resource is consumed

GT3 Job Submission example



Other Grid Technologies

- There exist many other Grid Technologies which are more geared to highly coupled systems
  - Xgrid (Apple)
    - Turns a group of Macs into a “supercomputer”
  - .NET (Microsoft)
    - Infrastructure for Windows based grids with single sign on capablility
  - Condor
    - CPU cycle harvesting across multi platform clusters
- Will be getting hands on experience with Condor in the Programming Exercise