



Introduction to Grid Programming with the Globus Toolkit Version 3

Michael Brown
IBM Linux Integration Center
Austin, Texas



Session Introduction

- Who am I?
 - mwbrown@us.ibm.com
 - Team Leader for Americas Projects
 - IBM Linux Integration Center, Austin
 - Author on 2 IBM Redbooks about Globus 3
- What will we talk about ?
 - Recent history of grid computing
 - Intro to the galaxy of Globus
 - Run a Globus V3 example (fingers crossed)



Introduction to Grid Computing

- Don't panic: it's just the new word for distributed computing
- We want to exploit resources on a wide, loose network
 - Not a cluster, but heterogeneous systems
 - CPU, storage, software, devices, all w/quality-of-service
- We started small in the early days with file copying, printing
- Later we moved to program-program communication
- We have to solve the same problems every time
 - Comms protocol, data format, security, management
- Why can't we just focus on the high-level tasks? Please?
 - Leverage existing standards
 - Follow accepted patterns



Classes of Grids

- Computation
 - The obvious one
 - Systems dedicated to tasks, or scavenging (including desktop)
- Data Virtualization
 - Intelligent, cross-network virtual filesystem
- Business Intelligence
 - Single coherent view of enterprise data
- Analytics
 - Computation + business intelligence
- High Availability
 - Provide better uptime due to no single point of failure



The Globus Project

- Write code to provide a grid infrastructure
 - <http://www.globus.org>
- Major contributors
 - Argonne National Labs
 - University of Chicago
 - University of Southern California
 - Northern Illinois University
- Sponsors
 - DARPA
 - U.S. Department of Energy
 - National Science Foundation
 - NASA
- Supporters



What Is Globus Toolkit 2.x?

- Known as GT2
- C-based grid toolkit – de facto grid standard
 - Used heavily in academic and research grids
 - Still supported, some features used in GT3
- Layers
 - Applications
 - High-Level Services and Tools
 - Core Service
 - Local Services

GT2 Architecture

Applications

High Level Services & Tools

DRM Cactus MPICH-G2 globusrun PUNCH Nimrod/G Grid Status Condor-G

Core Services

GASS GridFTP MDS GSI Replica Catalog I/O GRAM

Local Services

Condor LSF MPI PBS NQE Common Linux TCP AIX UDP Solaris





GT2 Local Services

- Condor
 - Job and resource manager for compute-intensive jobs
- MPI – Message Passing Interface
 - Portability across platforms
- LSF – Load Sharing Facility
 - Management of batch workload from Platform, Inc.
- PBS – Portable Batch System
 - Scheduling / resource management from Veridian Group *via* NASA
- NQE – Network Queueing Environment
 - Resource manager on Cray systems
- Common
 - Thread library, libc wrappers, callbacks, object management



GT2 Core Services

- GASS – Globus Access to Secondary Storage
 - File and executable staging and I/O redirection
- GridFTP - Grid File Transfer Protocol
 - Reliable, high performance FTP with 3rd party transfer capability
- MDS - Metacomputing Directory Service
 - Maintains information about available resources
- GSI - Grid Security Interface
 - Authentication, authorization *via* proxies, delegation, PKI, SSL (not Kerberos!)
- Replica Catalog
 - Manages partial copies of full data set across grid
- GRAM - Grid Resource Allocation Management
 - Allocation, reservation, monitoring and control of programs on remote systems
- I/O
 - Wrapper TCP, UDP, IP multicast and file I/O



GT2 High Level Services

- DRM – Distributed Resource Management
 - Resource manager on ASCI supercomputer
- Cactus
 - Grid-aware numerical solver framework
- MPICH-G2
 - Grid-enabled MPI
- Globusrun
 - More complicated version of globus-job-run
 - Complex Resource Specification Language expressions



GT2 High Level Services Cont.

- PUNCH
 - Web browser resource manager from Purdue
- Nimrod/G
 - Model computational jobs from Monash
- Grid Status
 - Repository of state of jobs in grid
- Condor-G
 - Condor job management layer to Globus



Running GT2

- Run grid-proxy-init
 - User enters passphrase to decrypt private key
 - Private key signs proxy certificate
 - Proxy certificate placed in /tmp
 - Traffic managed by gatekeeper
 - Process acts like inetd for available services
 - Handles authentication issues
- Run desired command
 - eg. `globus-job-run <host> <program> [args]`

What Is a Web Service?

- Program-program interoperability standard
 - Allows for construction of loosely coupled applications
- Platform and language independent
- You don't need previous knowledge of a service to use it
- Uses standards-based protocols
 - WSDL – Description of the Web service interface
 - UDDI – Directory of available services
 - SOAP – Network procedure call
 - HTTP – Transport protocol (gets through ~ anything)
 - XML – Payload data format for messages

■ See Hatzidakis, Nash, Lawrence



Web Services Invocation

- Find a service by querying the UDDI registry
- Registry replies with a list of servers
- Ask web service to describe its invocation format
- Service replies with WSDL definition
- Send SOAP request message to service
- Service replies with SOAP response message
 - Contains requested data or error info



What Is the GGF?

- Global Grid Forum <http://www.ggf.org>
 - “ to promote and support the development, deployment, and implementation of Grid technologies and applications via the creation and documentation of "best practices" – technical specifications, user experiences, and implementation guidelines.”
- Developed two key specifications for us
 - OGSA
 - OGSI



Goals of OGSI and OGSA

- Provide a grid framework
- Leverage existing standards
- Services are heterogeneous
- Services are dynamic
- Services are searchable
- Services are callable by any system on the grid
- Services are independent of implementation



What Is OGSA?

- Open Grid Services Architecture
- Defines what Grid Services are
 - Enhanced Web Services
- Defines a set of core and higher-level features
 - Factory, Registry, Discovery, Lifecycle, Authentication
 - Policy, Monitoring, Management, SLA, Hierarchy, QOS
 - Query Service Data, Notification, Reliable Invocation
- Architecture layers can be implemented by different products, ISVs, open source communities, *etc.*
- Pick and choose the “best” implementation



OGSA Programming Model

- All OGSA services adhere to specific service interfaces and behaviours
- Service interface defined by GWSDL
 - Grid WSDL
 - OGSA has identified several WSDL extensions



What Is OGSI?

- Open Grid Services Infrastructure
- Formal specification for OGSA concepts
- What do we need to build grid services?
 - Stateful Web services
 - Life cycle functions
 - Naming functions
 - Service Data
 - Notification of state change



OGSI Stateful Web Services

- Normal Web services are not stateful
 - This is not helpful in a grid environment
- We need smarter instances
 - Can be long-running
 - Can be used in a chain of related operations
 - Need to be able to query, pause, stop jobs, *etc.*
- Introduce Factory approach
 - Creates and manages pool of running instances
 - Can be unique or shared amongst clients



OGSI Lifecycle

- Create and destroy service instances
 - Actual mechanism not defined by OGSI
- Can create *via*:
 - Direct invocation of method on service
 - Factory
- Can destroy *via* direct invocation of method
- Can allow to die
 - Don't respond to the keep-alive notification



OGSI Naming

- Web Services addressed *via* a URI
 - `http://my.server.com/applications/serviceA`
 - Permanent, unique
 - But OGSA calls it a Grid Service Handle (GSH)
- A GSH is insufficient to talk to an instance
 - Resolve GSH to a Grid Service Reference (GSR)
 - This is temporary as an instance could die at any time
 - GSR is defined by a WSDL grammar (methods, *etc.*)
 - Use WSDL because GT3 implemented SOAP / HTTP



OGSI Service Data

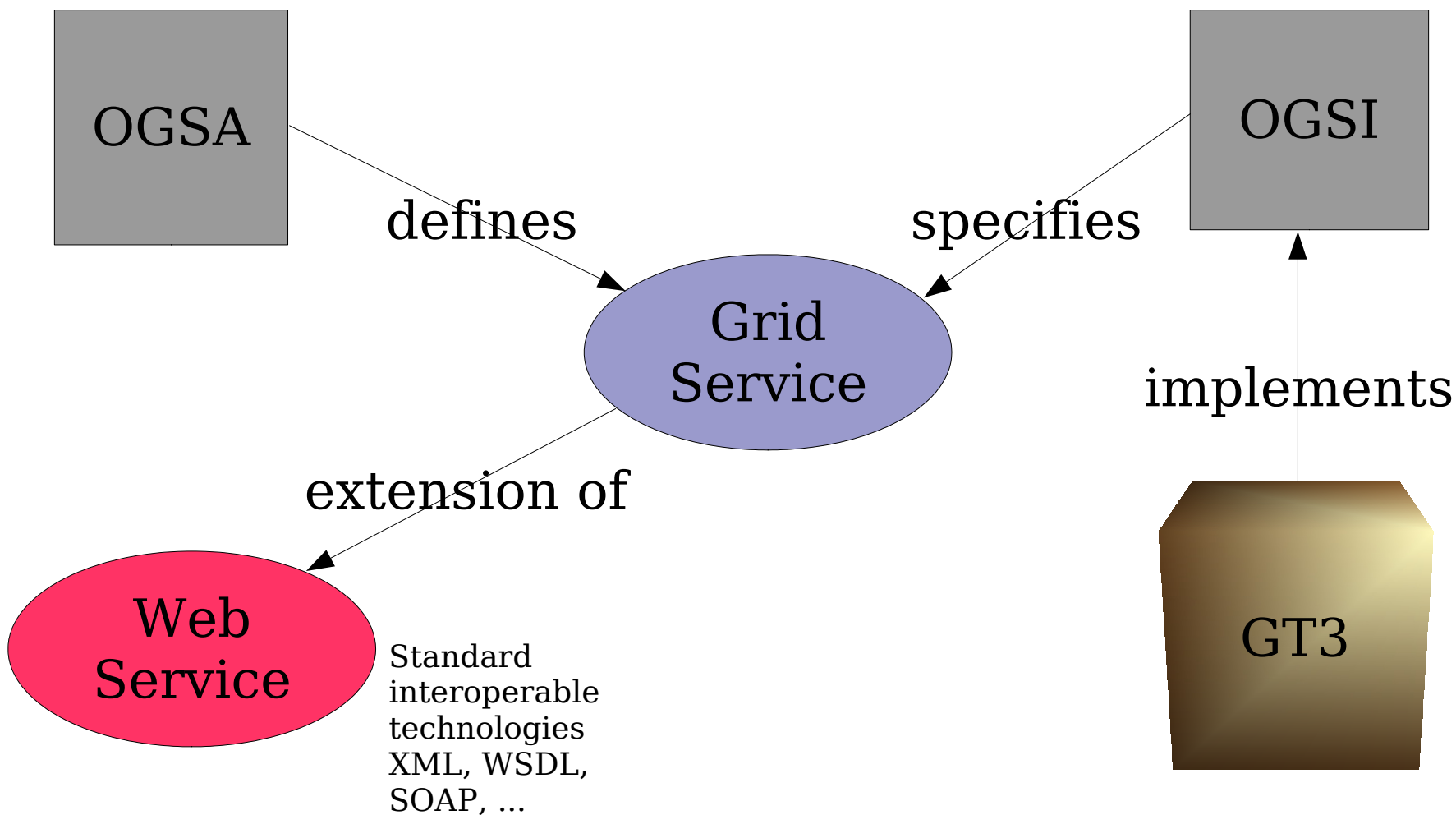
- One of the most important concepts
 - Storage area associated with a grid service
 - Collection of Service Data Elements (SDE)
 - Zero or more SDEs per instance
 - List of types defined by OGSI
 - Externally expose state data to service requesters
- Query
- Update
- Fire notification to subscribers on change



OGSI Notification

- Register and deliver async events
- Notification Source
 - Delivers notification messages
- Notification Sink
 - Receives notification messages

OGSA/OGSI/GT3 Relationship



From GT3 Tutorial, Sotomayor



What Is the Globus 3 Toolkit?

- Here we are finally!
- Open source implementation of OGSI 1.0
- Usable Web services-based grid services toolkit
- Runtime environment called the container
- Written in Java
- Some GT2 components have no GT3 equiv
 - Full support for GT2.4, written in C
 - GT3 job manager launches as separate process
 - Deprecating over time



GT3 Core Services

- Implements all OGSI interfaces
 - GridService, Factory, Notification, HandleResolver
 - Admin, Logging, Management, *etc.*
- Run time environment for services
 - Between application and plumbing
 - Embedded: library for any J2SE application
 - Standalone: lightweight server (test, development)
 - Web: runs in a J2EE servlet engine
 - EJB: expose stateful Entity and Session beans as services



GT3 Security Services

- Transport- and message-level security
 - Transport being deprecated
 - Message based on WS-Security, XML Signature
 - At SOAP level
 - Per-session or per-message
- SSL, X.509 certificates
- Based on JAAS
 - Authentication, Authorization Service Framework



GT3 Base Services

- Database services
 - Persisting Service Data in native XML db
- Managed Job Service
 - Notify, subscribe, pause, stop, query jobs
- Index Service
 - Query services around grid
- Reliable File Transfer
 - Guaranteed large file transfers *via* restart
- Replica Location Service



Writing a Simple Grid Service

- Define all aspects of the service interface
 - Java Interface and/or GWSDL
- Generate grid service support code
 - Server and client stub classes
- Implement the service guts
- Fortunately this process is very automated
 - ant is your friend



Our Simple Grid Service

- Message of the Day service
- Differentiate it from a normal Web Service
 - Stateful server keeps track of which messages sent
 - Messages delivered in order
 - Will not repeat until all MOTDs delivered
- Client calls instance's getMOTD()



Define a Service Interface

- Two ways:
- Bottom-up
 - Write a Java Interface
 - Run a tool against it to generate WSDL
 - Be careful! Some complex Java types do not map well into WSDL.
 - Good for exposing legacy code w/o reprogramming
- Top-down
 - Write a GWSDL Port Type definition
 - Run two tools against it to generate full WSDL



Top-Down Service Definition

- Write the abstract definition of the service
 - Types, Messages and PortType in GWSDL
 - Better for more complex service data types
 - Annoying XML syntax
- Run through the GWSDL2WSDL tool
 - Creates WSDL 1.1 portType definition
- Run through the generateBindings tool
 - Creates the wsdl:binding and wsdl:service parts of portType definition



Generate Service Support Code

- Generate Java stubs
 - Handles marshalling of data to/from XML
 - Server & client side JAX-RPC-compliant interfaces
- All done by ant



Implement the Service

- Write a Java class that does the job
 - Implement the service interface
 - Private attributes make it stateful
 - Private methods in the class are OK
- Simple constructor
 - Calls `super()` to pass to `GridServiceImpl`
- Implement service methods
 - String `getMOTD()`



Deploy the Service

- Write a deployment descriptor
 - Tells the Web server how the service is published
 - Extra fields define factory attributes
- Create a GAR
 - Grid ARchive, a JAR with grid-specific extras
 - Best to copy/paste an existing ant task
 - target="makeGar"
- Deplot the GAR into a hosting environment
 - ant deploy -Dgar.name=<path to GAR>



Write a Service Client

- Command-line client application
 - Good idea to pass service info on command line
 - Or use Index Service to discover
- Make a GSH from the factory service URI
 - Build up with hostname and factory path
- Locate the factory using the GSH
- Create a service instance with the factory
- Call a remote method on the instance
- Destroy the instance



Prepare the Server for Clients

- Start the container
 - ant startContainer
- If using factory, don't need to create an instance
- If not using a factory, create a service instance
 - java org.globus.ogsa.client.CreateService <serverURL>/<factory service name>
 - ServerURL = <http://hostname:8080/>
 - Factory = /<path to factory>
- Container now running, waiting for requests



Test the Client

- Launch the client
- Pass on the command line
 - Instance reference (GSR)
 - Like handle, but with /hash-123.... on end
- Client pulls and displays one MOTD
- Try it with your wireless laptop!



References

- *The Grid: Blueprint for a New Computing Infrastructure*
 - Foster et al, ISBN 1558604758
- High-level grid papers from www.globus.org
 - The Anatomy of the Grid (anatomy.pdf)
 - The Physiology of the Grid (ogsa.pdf)
- GT3 papers from www.globus.org
 - gt3-faq.html, hunt around for many more
- *Globus V3 Programmer's Tutorial*, Borja Sotomayor
 - www.casa-sotomayor.net/gt3-tutorial



References 2

- Redbooks from www.ibm.com/redbooks
 - *Grid Computing with Globus (SG24-6895)*
 - *Enabling Applications for Grid Computing with Globus (GT2.2) (SG24-6936)*
 - *Globus Toolkit 3 Early Experiences (redp3697.pdf)*
 - *Globus Toolkit 3 with WSAD and WebSphere 5*
 - Working title
 - In process at this time