Grid Tutorial

Laukik Chitnis
Sanjay Ranka
Outline

- Distributed Applications
  - Distributed Computing – the paradigm
  - Problems and Solutions
- Web Services and Grid Services
  - Introduction, Concepts, Programming
- The Grid and its middleware
  - Grid Security
  - Information services
  - Data Management
  - Job Management
  - VDS
- Resource Management
- Hands-on exercise
Nature of Large Scale Distributed Applications

- Distributed data
  - Naturally collected at multiple units of an organization

- Distributed Resources
  - Resources are distributed across multiple units

- Data and Resource Ownership Issues
  - Differential access and privacy issues
Examples of Distributed Applications

- High Energy Physics applications
  - Monte Carlo simulations
  - CMS experiment
- Finding interesting astronomical patterns
  - Sloan Digital Sky Survey
- Coastal ocean monitoring and predicting
  - SURA Coastal Ocean Observing and Prediction (SCOOP)
- Prime number generator
  - Cracking DES

Cannot be done on a single machine
You want to divide the application and run it on a distributed and decentralized environment
CMS organization

Source: http://cmsinfo.cern.ch/
Open Science Grid (OSG)

- A consortium of Universities and National Laboratories to building a sustainable grid infrastructure for Science in the U.S.
  - Argonne, Fermilab, SLAC, Brookhaven, Berkeley Lab, Jeff. Lab
  - UW Madison, U Florida, Purdue, Chicago, Caltech, Harvard, etc.

- Shared resources, benefiting broad set of disciplines
- Incorporation of advanced networking
- Focus on general services, operations, end-to-end performance

Source: http://www.opensciencegrid.org/
# SDSS - Sloan Digital Sky Survey

![Galaxy Image](http://www.sdss.org/)

<table>
<thead>
<tr>
<th></th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Area (square-degrees)</td>
<td>7000</td>
</tr>
<tr>
<td>Storage (gigabytes)</td>
<td>1540</td>
</tr>
<tr>
<td>Compute (CPU-hours on 500 MHz PIII with 1 gigabyte RAM)</td>
<td>7000</td>
</tr>
</tbody>
</table>

## Sample workload from HEPGRID2001 workload generator

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time span covered by workload generator in the HEPGRID2001 model</td>
<td>1 year</td>
</tr>
<tr>
<td>Number of physicists submitting jobs</td>
<td>500</td>
</tr>
<tr>
<td>Number of jobs in workload</td>
<td>341/day</td>
</tr>
<tr>
<td>Average size of a job input set</td>
<td>$10^7$ products</td>
</tr>
<tr>
<td>Average size of a job input set</td>
<td>1.3 TB</td>
</tr>
<tr>
<td>CPU capacity needed to analyze requested products in jobs</td>
<td>960,000 S195</td>
</tr>
<tr>
<td>RAW products requested by all jobs</td>
<td>$4.5 \times 10^{8}$/year</td>
</tr>
<tr>
<td>ESD products requested by all jobs</td>
<td>$3.0 \times 10^{11}$/year</td>
</tr>
<tr>
<td>AOD products requested by all jobs</td>
<td>$9.4 \times 10^{11}$/year</td>
</tr>
<tr>
<td>Average number of times that a single product is requested</td>
<td>40</td>
</tr>
<tr>
<td>CPU capacity needed to derive all requested products once</td>
<td>433,000 S195</td>
</tr>
<tr>
<td>Different data products derived at least once</td>
<td>$31$/event/year</td>
</tr>
<tr>
<td>ESD products derived if all derived only once</td>
<td>$4.3 \times 10^{9}$/year</td>
</tr>
<tr>
<td>AOD products derived if all derived only once</td>
<td>$2.7 \times 10^{10}$/year</td>
</tr>
<tr>
<td>Size of RAW products</td>
<td>1000 TB/year</td>
</tr>
<tr>
<td>Size of ESD products derived if all derived only once</td>
<td>2166 TB/year</td>
</tr>
<tr>
<td>Size of AOD products derived if all derived only once</td>
<td>269 TB/year</td>
</tr>
</tbody>
</table>
So, what are the issues?

- Communication
  - RPC
  - MPI
- Remote Process Management
  - Mobile Agents
  - Client/Server paradigm
- Naming
  - Globally unique names
  - Name resolution, assignment and maintenance
- Synchronization
- Security
Solutions

- Grid Computing!
- Is it new?
  - Not really
- Can we do the same with web services?
  - Absolutely
- But, what about these grid services?
Why do you want a grid?

- Different perspectives
  - User: I want to run my scientific application on the grid so that I can get results in 10 hours instead of 10 days
  - Organization: Our next big experiment will generate tera-bytes of data and we want to distribute, share and analyze the data
  - Organization: We want to tap into the existing grids and share resources
Distributing Application – A User Perspective

I need

- More CPU cycles
- More disk space
- More bandwidth
- Better software tools
- All of the above

Alternatives to grid

- Simple CPU cycle stealer
- Simple SRM (Storage Resource Manager)

Run my app in 10 hrs that usually takes 10 days on my pentium
Distributed Systems - Sys admin perspective

- How do I distribute the load on the machines?
- How do I reduce the overhead on the central server?
- How do I manage local and remote users?
- What should be the policies?
Why Grid? – Organizational Perspective

- Federation of scientists – distributing, sharing and analyzing data
- Tapping into existing grids
- Cost-effective: A grid can be built from commodity software and hardware without spending millions on the next super duper computer.
- Reliability: If a site fails, we can simply move our jobs to another site (this can be seen as a user perspective as well)

Where do you want to run your job today?
Distributed Application Requirements

- Requires
  - A lot of resources
  - Reservation of resources at a particular time
  - Monitoring of status of the submitted jobs to multiple sites
  - Storage that is not easily available at a single place
So, you do want a Grid for your applications!

Harnessing the enormous power of the networked heterogeneous systems

Before we plunge into the details of grid services let’s first see the motivation for them (the web services)
Web Services
Web Services

A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards. (W3C definition)
Web services

- In plain words, they provide a good mechanism to connect heterogeneous systems with WSDL, XML, SOAP etc.
A Sample interaction

Key things to note:
Request/response

Buyer

login

login ok

buy a book

price information

confirm purchase

Amazon.com
The Web Service state machine

1. Service advertises itself in the registry

2. Client looks up the service in the registry

3. Client interacts with the service
Why web services?

- Can we not do above interaction with traditional client/server protocols
- Yes!
- But,
  - We want to talk to any service in the same language in the same way
So, the solution

- XML + WSDL + SOAP + HTTP + UDDI
  + ...

- The key components are WSDL and SOAP and of course XML!
XML (Extensible Markup Language)

- A language for describing data
- Platform independent and self-describing
- Good for distributed computing where heterogeneous systems co-exist
An example XML file

```xml
<?xml version="1.0"?>
<contact-info>
  <name>John Smith</name>
  <company>University of Florida</company>
  <phone>352-392-1200</phone>
</contact-info>
</xml>

- Let’s dissect this example to understand different elements of xml
A Minimal XML declaration
- `<xml version="1.0"?>`

We can also specify the encoding of the text of the document.
- `<xml version="1.0" encoding="UTF-8"?>`

Note that the XML declaration is case-sensitive
Understanding XML – Tags and Elements

- The tags mark the start and end of the elements which form the basic units of an XML document

```
<contact-info>
  <name>John Smith</name>
</contact-info>
```

- Note that elements cannot overlap
XML tags and elements

- **Wrong**
  - `<a> ... <b> ... </a> ... </b>`

- **Right**
  - `<a> ... <b> ... </b> ... </a>`

- Again, note that the element and attribute names are case-sensitive
Attributes

- An attribute specifies a property of an element
- It has a name and a value
  - `<a href="http://www.ufl.edu"> UFL </a>`
- Attribute values are specified in quotes while names are not
Any text can be included in an XML document. It just has to conform to the encoding.
<?xml version="1.0"?>
<contact-info>
  <name>John Smith</name>
  <company>University of Florida</company>
  <phone>352-392-1200</phone>
</contact-info>
</xml>
Web services architecture

Processes
Discovery, Aggregation, Choreography...

Descriptions
Web Services Descriptions (WSDL)

Messages
SOAP Extensions
Reliability, Correlation, Transactions ....

SOAP

Communications
HTTP, SMTP, FTP, JMS, IIOP, ....

From W3C web services architecture document
Web Services Interaction

User → XML → Web Service

Resource (database, CPU, storage …)
Building blocks of a web service

- SOAP (the medium)
- WSDL (the description)
- XML (the language)
- UDDI (discovery protocol)
Web Services Architecture – a conceptual stack

- Service Discovery
- Service Publication
- Service Description and Implementation
- Messaging
- Underlying Network

UDDI

WSDL

SOAP messages written in XML
An example scenario

Tell me about your service

WSD (Web service description)

Send a SOAP message

Returned SOAP message

Amazon.com

Buyer

WSDL (Web service description)

<portType name="BookService">
  <operation name="buyBook">
    <input name="bookName" message="tns:bookName"/>
    <output name="price" message="tns:price"/>
  </operation>
</portType>
The Web Service state machine

1. Service advertises itself in the registry (UDDI)

2. Client looks up the service in the registry (UDDI) and gets a WSDL description

3. Client interacts with the service (SOAP + HTTP + XML)
The technologies

- **WSDL**
  - to describe the basic format of web service requests

- **SOAP**
  - defines a uniform way of passing XML-encoded data

- **XML and**
- **UDDI concepts**
  - for finding web services
SOAP (Simple Object Access Protocol)

- **SOAP** is a protocol specification that defines a uniform way of passing XML-encoded data.
- It also defines a way to perform remote procedure calls (RPCs) using HTTP as the underlying communication protocol.
- It is the underlying protocol for all web services.
XML messaging using SOAP

Application

SOAP

Network protocols (HTTP …)

Web Service

SOAP

Network protocols (HTTP …)
SOAP Usage example

- Say, you want to talk to a small web service that gives personalized “hello” messages
  
  ```java
  public interface Hello {
    public String greeting(String name);
  }
  ```

- Now, you know the interface. But, you do not know how it is implemented. All you want to do in your favorite language is to call the function

  ```java
  greeting ("world")
  ```

  and expect a response like

  “hello world”
So, how do we talk to the service in a platform-independent way? – XML !!!

The Request

```xml
<Hello>
  <greeting>
    <name>John</name>
  </greeting>
</Hello>
```

The Response

```xml
<Hello>
  <greetingResponse>
    <name>Hello John</name>
  </greetingResponse>
</Hello>
```
SOAP

- SOAP allows you to send and receive such messages in proper format
- A real SOAP message for the example might look like
xmns:xsi="http://www.w3.org/1999/XMLSchema-instance"
xmns:xsd="http://www.w3.org/1999/XMLSchema">
  <SOAP-ENV:Header>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <ns1:greeting xmlns:ns1="Hello"
                  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <name xsi:type="xsd:string">John</name>
    </ns1:greeting>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
WSDL (Web Service Description Language)

- Now, we know how to send messages. But, how do we find out about the web service interface? Answer: WSDL !!!

- **WSDL** provides a way for service providers to describe the basic format of web service requests over different protocols or encodings.

- It provides the following information about the service:
  - What the service can do
  - Where it resides
  - How to invoke it
WSDL

- Provides a platform and language independent abstraction of the service
- It specifies the port types, messages exchanged and how data is encoded etc.
- A port is defined by associating a network address with a reusable binding; a collection of ports define a service
Contents of a WSDL document

<definitions>
  <types>
    definition of types ........
  </types>
  <message>
    definition of a message ....
  </message>
  <portType>
    definition of a port .......
  </portType>
  <binding>
    definition of a binding ....
  </binding>
</definitions>
portType

- WSDL portType is the most important element of the document
- It defines the operations that can be performed on a web service

```xml
<portType name="Hello">
  <operation name="greeting">
    <input name="name" message="greetingRequest"/>
    <output name="response" message="greetingResponse"/>
  </operation>
</portType>
```
Messages and Types

- But, to describe the operation we also have to identify the messages that need to be sent and type of the elements that are sent in the messages

  ```xml
  <message name="greetingRequest">
    <part name="input" type="xs:string"/>
  </message>

  <message name="greetingResponse">
    <part name="value" type="xs:string"/>
  </message>
  
  As you can see, various types can be specified for the messages
The binding element describes the way the message needs to be encoded (usually using SOAP)

```xml
<binding type="Hello" name="myBinding">  <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http" />
<operation>  <soap:operation soapAction="http://example.com/Hello"/>
    <input>  <soap:body use="literal"/> </input>
    <output>  <soap:body use="literal"/> </output>
  </operation>
</binding>
```
<definitions name="Hello" targetNamespace="http://hello.com"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns="http://schemas.xmlsoap.org/wsdl/">
<message name="greetingRequest">
  <part name="input" type="xs:string"/>
</message>
<message name="greetingResponse">
  <part name="value" type="xs:string"/>
</message>
<portType name="Hello">
  <operation name="greeting">
    <input name="name" message="greetingRequest"/>
    <output name="response" message="greetingResponse"/>
  </operation>
</portType>
<binding type="Hello" name="myBinding">
  <soap:binding
    style="document"
    transport="http://schemas.xmlsoap.org/soap/http" />
  <operation>
    <soap:operation
      soapAction="http://example.com/Hello"/>
    <input>
      <soap:body use="literal"/>
    </input>
    <output>
      <soap:body use="literal"/>
    </output>
  </operation>
</binding>
</definitions>
XML Parsers

- **Validating**
  - in addition to checking well-formedness, the parser verifies that the document conforms to a specific DTD (either internal or external to the XML file being parsed).

- **Non-validating**
  - the parser does not check a document against any DTD (Document Type Definition); only checks that the document is *well-formed* (that it is properly marked up according to XML syntax rules).
Some XML Parsers

- **Validating**
  - **Xerces** - The Apache XML Project is maintaining XML parsers in Java, C++, and Perl
  - IBM's XML Parser for Java
  - Oracle XML Parser

- **Non-validating**
  - expat
  - LT XML
  - Xparse
UDDI (Universal Description, Discovery and Integration)

- A protocol for finding web services
- Registries of web services can be maintained
- The primary purpose is to find services with certain qualities
- **BusinessEntity**
  - Information about a company (name of the company, contact info etc.)
  - Kind of company
- **BusinessEntity** contains **BusinessService** elements that represent the services the company offers
- Each **BusinessService** contains **BindingTemplate**s that describe the services

<table>
<thead>
<tr>
<th>BusinessEntity (company description)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BusinessService (Service Description)</td>
</tr>
<tr>
<td>BindingTemplate (service technical description) modeled as a tModel</td>
</tr>
</tbody>
</table>
Web Services programming

- There are two major methodologies
- Bottom-up and Top-down

**Bottom-up:**
- Write the java code
- Generate java interfaces
- Generate WSDL from the interfaces

**Top-down:**
- Write the WSDL interface
- Generate java interfaces
- Generate java stubs and add functionality
So, what are these grid services anyway?

Grid services are web services that are customized to grid environment.

Similar to web services they provide the glue to interact with heterogeneous systems.

Why do we need them?

What do they provide?
Web Services vs Grid Services

- Though web services are great, some key things that are required on the grid are missing
  - State management
  - Global Service Naming
  - Reference resolution
  - more …
Web Services vs Grid Services

- Wait a minute! I can do all those things with web services, can’t I?
- YES! You can
- But,
  - The standards don’t provide (yet) the required mechanisms. Work is being done to figure out the best way to do these things
Stateful web services example

Buyer

login

Amazon.com

login ok, your shopping cart id is 0x800

logout

login and my id is 0x800

Your shopping cart has …
Problems

- No standard on how to do this
- Client needs to have special code
- Some protocol specific features like cookies can be used
OGSI

- OGSI defines the mechanisms to create grid services
- Introduces the notion of grid services and their special properties particularly the global pointer GSH (Grid Service Handle)
GSH and GSR

- Key concepts in OGSI

Resolving GSH (From OGSI draft)
Other OGSI concepts

- GWSDL (extensions to WSDL)
- Service data elements (SDEs) for a stateful service.
- GSH and GSR
- Factory
- HandleResolver
- Notifications
Grid Services as seen by OGSI

- Connect to the grid service
- Ask the server to create an instance for you
- Get a unique global pointer to it
- Interact with the service
WS-RF

- Confusion and Criticism from web services folks
- Implied Resource pattern
- Modeling stateful resource with Web services
WS-Resource

- Provides a means of expressing the relationship between stateful resources and web services
- The WS-Resource has an XML resource property document defined using XML schema.
- The requestor can determine the WS-Resource type by retrieving the portType
- Web service programming paradigm is used to interact with the resource
Programming Grid Services (with GT3)

- Basic steps involved in creating a grid service
  - Create the interface using WSDL
    - Specify the portTypes, messages and data encoding
  - Generate Stubs
  - Add functionality
  - Compile and Build the code using Globus libraries
  - Create a GAR (Grid Archive)
  - Deploy it
Grids
and
Grid Middleware
What is a Grid?

Simply put, a Grid is a congregation of different sites (clusters) collaborating to increase productivity.

For example, **Grid3**

A total of 35 sites
Over 3500 CPUs
Update: Grid3 is now assimilated into Open Science Grid

source: GridCat (http://osg-cat.grid.iu.edu/)
Broad Division of Grid

- **Data Grids**
  - Managing and manipulating large amounts of data. Main objective is to share large amounts of data that is otherwise impossible without the grid

- **Compute Grids**
  - For compute-intensive tasks. Emphasis is on the federation of CPU cycles and distribution of compute intensive tasks

There is no consensus on such categorizations and it only aids in understanding the requirements
Grid Building Blocks

- **Hardware**
  - Computational Clusters
  - Storage Devices
  - Networks

- **Software components**
  - Monitoring Infrastructure
  - Job execution infrastructure
  - Grid-wide authentication mechanism
Dell Cluster at UFlorida’s High Performance Center

A typical cluster rack

Cluster Management “frontend”

I/O Servers typically RAID fileserver

Tape Backup robots

Disk Arrays

A few Headnodes, gatekeepers and other service nodes

The bulk are Worker Nodes

Dell Cluster at UFlorida’s High Performance Center

Some slides are from Jorge Rodriguez’s presentation on “Building, Monitoring and Maintaining a Grid”
A head node is used as a gateway to submit all the jobs to the cluster.

Scheduling decisions are made on the gateway considering resource allocation and load balancing.

- Mostly using batch job scheduling systems
  - Such as PBS, Condor
- Execution of jobs occur on one or more of the worker nodes

Head node usually also handles

- Authentication
- Enforcing policies
Storage on the cluster

- Storage devices may vary from a single file server to large scale mass storage systems

- Typical hardware components
  - Servers: Linux, RAID controllers…
  - Disk Array
    - IDE, SCSI,…
    - RAID

- Local Access
  - Volumes mounted across compute cluster
    - nfs, gpfs, afs…
  - Volume Virtualization
    - dCache
    - pnfs

- Remote Access
  - gridftp: globus-url-copy
  - SRM interface
    - space reservation
    - request scheduling
Cluster networking

- **Internal Connectivity**
  - The worker nodes have private IP addresses, accessible only to servers inside a facility
    - Some sites allow outbound connectivity via Network Address Translation
  - Typical technologies used
    - Ethernet (0.1, 1 & 10 Gbps)
    - High performance, Low Latency interconnects
      - Myrinet: 2, 10 Gps
      - Infiniband: max at 120Gps

- **External connectivity**
  - Connection to Wide Area Network
  - Typically achieved via same switching fabric as internal interconnects
  - Only the head node of the cluster is exposed to the outside world
A typical cluster networking

Head Node/Frontend Server

I/O Node + Storage

Worker Nodes

Network Switch

WAN
Grid Middleware
A typical grid application, especially the ones in HEP and Astronomy experiments, consist of:

- Large datasets as inputs
- “Transformations” which work on the input datasets
- The output datasets

The emphasis is on the sharing of the large datasets

The transformations are usually long and can be parallelized
A (small) step in a typical HEP/Astronomy Grid Application

Input datasets → Transformation → Output datasets
A Typical Grid Application Workflow
(Small Montage Workflow)

~1200 node workflow, 7 levels

Mosaic of M42 created on the Teragrid using Pegasus

Source: Montage, Pegasus
Simple Expectations from the Grid

- Simply put, we need the grid to do the following operations for each transformation:
  - Find the input datasets
  - Apply the transformations (process the input)
  - Store the output datasets
    - and publish its “presence” so that collaborating scientists can find it
Qualities expected from the grid

- And of course, we would like these operations to be performed:
  - Efficiently
    - As quickly as possible
  - Seamlessly
    - for easy collaboration
  - Fairly
    - fair to all collaborators
  - Securely
    - security against loss (fault tolerance), unauthorized access
In an effort to view the ‘Grid as a Workstation’, a set of grid software and services act as middleware between the user and the grid of machines.

These services can be roughly categorized as follows:
- Security services
- Information Services
- Data Management
- Job Management
- Virtual Data System
Services offered in a Grid

- Resource Management Services
- Information Services
- Security Services
- Virtual Data System
- Data Management Services

Note that all the other services use security services.

In the following sections, we will study each of these in detail.
Grid Security
Security Services

- Forms the underlying communication medium for all the services
- Secure Authentication and Authorization
- Single Sign-on
  - User need not explicitly authenticate himself every time a service is requested
- Uniform Credentials
- Ex: GSI (Globus Security Infrastructure)
Creates a proxy for single-sign on
A GSI certificate includes four primary pieces of information:

- A subject name, which identifies the person or object that the certificate represents.
- The public key belonging to the subject.
- The identity of a **Certificate Authority (CA)** that has signed the certificate to certify that the public key and the identity both belong to the subject.
- The digital signature of the named CA.

A third party (a CA) is used to certify the link between the public key and the subject in the certificate. In order to trust the certificate and its contents, the CA's certificate must be trusted. The link between the CA and its certificate must be established via some non-cryptographic means, or else the system is not trustworthy.
Grid certificate

- Simply speaking, you need a certificate that authenticates you and which can be used as a single sign-on in systems like Globus.

- To get a personal grid certificate, usually the following steps are involved:
  - Import the CA certificate into your browser.
  - Request your certificate.
  - Retrieving your certificate.
  - Exporting your key pair for use by Globus grid-proxy-init.
DOEGrids certificate

- Import the DOEGrids Certificate Chain
  1. Go to: http://pki1.doeegrids.org
  2. Select "Retrieval" tab
  3. Select "Import CA Certificate Chain" from the menu.
  4. Under "Users", select the radio button "Import CA Certificate Chain into your browser".
  5. Click the "Submit" button.

source: http://www.doeegrids.org
Requesting a Grid Certificate

- Request your certificate.
  - Point your browser to https://pki1.doebridgs.org
  - Select Enrollment tab.
  - Fill in the New User Form.
  - When you click 'submit' the key pair will be generated.
Retrieving your Grid Certificate

- Go to [http://pki1.doegrids.org/srchCert.html](http://pki1.doegrids.org/srchCert.html)
  - This is a search interface to locate your certificate in the DOEGrids CA repository.
  - You MUST use the same browser on the same computer that you used to request the certificate, because it is this browser that is holding your private key in its certificate repository.

- In the "Subject Name" section, check the box marked "Show certificates with a subject name matching the following", and enter your last name in the "Common Name" box. Make sure Match Method of Partial is selected. Scroll (way down) to the bottom of the page and click Find.

- At least one certificate - hopefully yours! - should be listed. Click Details next to your certificate. At the very bottom of the Details page, click "Import Your Certificate".
Export your Grid Certificate

- The interface for this varies from browser to browser.
  - Internet Explorer starts with "Tools -> Internet Options -> Content";
  - Netscape Communicator has a "Security" button on the top menu bar;
  - Mozilla starts with "Edit -> Preferences -> Privacy and Security -> Certificates".

- The exported file will probably have the extension .p12 or .pfx. Guard this file carefully. Store it off your computer, or remove it once you are finished with this process.
Grid Certificate

- Exporting your key pair for use by Globus grid-proxy-init.
  - Copy the above PKCS#12 file to the computer where you will run `grid-proxy-init`.
  - Extract your certificate (which contains the public key) and the private key:
    - Certificate:
      ```sh
      openssl pkcs12 -in YourCert.p12 -clcerts -nokeys -out
      $HOME/.globus/usercert.pem
      ```
    - To get the encrypted private key:
      ```sh
      openssl pkcs12 -in YourCert.p12 -nocerts -out
      $HOME/.globus/userkey.pem
      ```
  - You must set the mode on your userkey.pem file to read/write only by the owner, otherwise grid-proxy-init will not use it
    ```sh
    chmod go-rw $HOME/.globus/userkey.pem
    ```
Once your user certificate is in place, you need to create a grid proxy which is used for accessing the Grid.

In Globus, you can do this using `grid-proxy-init`.

A proxy is like a temporary ticket to use the Grid, default in the above case being 12 hours.

Once this is done, you should be able to run “grid jobs”

`globus-job-run site-name /bin/hostname`
The working of GSI can be better understood if we consider the following “actors”:

- The user laptop (used for requesting and importing the user certificate)
- The user client machine (which requests the grid services)
- The Grid site (usually the head node of a cluster that provides the grid service)
- The Certificate Authority (CA that issues the user certificate)
- The VO admin (for adding authorized access to grid sites)
1. Import CA certificate chain

2. Request user certificate (include sponsor and VO information). Public key generated by browser included.

3. CA confirms with the sponsor (with a phone call!).

4. Issues a user certificate (assigns a subject name to the user)

5. User imports the issued certificate (The browser holds the private key and confirms the authenticity of the cert)

6. The .p12 file is copied to the client machine

Continued..
7. User requests the VO admin access to required grid servers/services.

8. VO admin adds the subject of the user certificate to the \textit{gridmap-file} of each of the required head nodes (ex. Using VOMS).

9. User creates a proxy for use with GSI using \textit{grid-proxy-init}.

10. User requests a head node of a grid site for some service. Ex. \textit{globus-run}

Continued..
Client machine

10. User requests a head node of a grid site for some service. Ex. *globus-run*

   The proxy generated in the previous step is forwarded to the grid site


Grid site

8. VO admin adds the subject of the user certificate to the *gridmap-file*.

11. The grid site verifies the identity of the user by matching the subject of the proxy received with the request with the entry in its *gridmap-file*.

12. If authenticated, the requested service is performed.

13. User gets the response from the grid site.
Steps 1 thru 7 talk about acquiring a Grid user certificate
- This is a one-time affair;
- Certificates, once obtained, are usually renewed once every year

Step 8 enables authorization of a user to access a set of grid sites

Step 9 creates a temporary ticket for the user to perform grid operations

Step 10-13 are repeated for every job a user executes on the grid
Gridmap file

- A *gridmap* file at each site maps the grid id of a user to a local id
  - The grid id of the user is his/her subject in the grid user certificate
  - The local id is site-specific;
  - multiple grid ids can be mapped to a single local id
    - Usually a local id exists for each VO participating in that grid effort
  - The local ids are then used to implement site specific policies
    - Priorities etc.
**Gridmap file entry**

- The *gridmap-file* is maintained by the site administrator.
- Each entry maps a Grid DN (distinguished name of the user; subject name) to local user names.

```
# Distinguished Name                      Local username
#
"/DC=org/DC=doegrids/OU=People/CN=Laukik Chitnis 712960"    ivdgl
"/DC=org/DC=doegrids/OU=People/CN=Richard Cavanaugh 710220"  grid3
"/DC=org/DC=doegrids/OU=People/CN=JangUk In 712961"    ivdgl
"/DC=org/DC=doegrids/OU=People/CN=Jorge Rodriguez 690211"  osg
```
OSG Authentication: Grid3 Style

VOMS server @ iGOC

VOMS server @ LLab

VOMS server @ OLab

DN mappings

gridmap-file

gridmap-file

gridmap-file

mapping of user’s grid credentials (DN) to local site group account

iVDGL, GADU...

LColab, Lexp1

Oexp1, Aexp2...

user DNs

user DNs

user DNs

VOMS= Virtual Organization Management System
DN=Distinguished Name
edg= European Data Grid (EU grid project)

This slide is adapted from Jorge Rodriguez’s presentation on “Building, Monitoring and Maintaining a Grid”
Submit a simple job to a remote site; check if the remote gatekeeper “knows” you – has your Grid identity authorized to run jobs:

- $globusrun -a -r <sitename>/jobmanager-fork
  - GRAM Authentication test successful

Look for the “test successful” message. This means you are “authorized” to use the <sitename> “resource”.

If this still fails, there is a problem with your Grid certificate or your entry in the gridmap-file on that remote site.
Grid Monitoring
And
Information Services
A grid is complex and quite big
- To efficiently use it, monitoring of the resources is needed

Why?
- To check the availability of different grid sites
- Discovery of different grid services
- To check the status of “jobs”
- Maintain information on “health” of the different sites for informed decision making
  - scheduling
Type of Information

The information can be broadly categorized as
- Static, and
- Dynamic

Static information about a site includes
- Its resource tallies
  - Number of worker nodes, processors
  - Storage capacities
  - Architecture and Operating systems
- Path of default staging area for grid users
  - $app, $data
Dynamic Information

- The dynamic content includes
  - The number of jobs running on each site
  - The CPU utilization of different worker nodes
  - Overall site “availability”
- Such time-varying information is more critical for efficient scheduling of grid jobs
- Monitoring such information presents the question of trade-off between accuracy (pertinence) and intrusion by the monitoring activities in the processor time share.
Site Level Infrastructure

- MonALISA server
- MIS-Core Infrastructure
- MDS

stor_stat
Ganglia
GIP
job_state
...

Collector

Monitoring Information Consumer API

Monitoring information DataBase

Historical information DataBase

https: Web Services
GRAM: jobman-mis
GINI, SOAP, WDSL...

GridCat
ACDC
MonALISA
Discovery Service
others...

This slide is adapted from Jorge Rodriguez’s presentation on “Building, Monitoring and Maintaining a Grid”
OSG Monitoring & Information System

**Site Level infrastructure**
- Scripts & tools collectors
- Databases
- Application APIs

**Grid Level Clients**
- User interfaces & APIs
- Graphical Displays

- MonALISA server
  - MonALISA client
  - Metrics DataViewer

- OSG MIS-CI
  - ACDC job Monitor
  - GridCat

- MDS: Generic Info. Provider
  - BDII

- Gridcat
- MonALISA
- ACDC job Monitor
- Metrics DataViewer
- BDII
Tools provide basic information about OSG resources
- Resource catalog: official tally of OSG sites
- Resource discovery: what services are available, where are they and how do I access it
- Metrics Information: Usage of resources over time

- Used to assess scheduling priorities
  - Where and when should I send my jobs?
  - Where can I put my output?
- Used to monitor health and status of the Grid
GridCat

http://osg-cat.grid.iu.edu

http://www.ivdgl.org/grid3/gridcat

Source: http://www.ivdgl.org/gridcat/home/
Globus MDS + BDII

- LCG-2 currently uses LDAP based
  - GT Monitoring and Discovery Service (MDS2) with
  - Berkley Database Information Indexes (BDII)

- MDS2 uses backend scripts (information providers) to provide information

- BDII stores information in a Berkeley Database backend database
MDS-BDI I

Merci Slides on MDS-BDI I adapted from EGEE presentation
MDS-BDI I

- Local GIIS runs on CEs and SEs at each site and report dynamic and static information regarding the status and availability of the services.

- At each site, a site GIIS collects the information of all resources given by the GIISes.

- Any site can run a BDII. It collects the information coming from the GIISes and collects it in a database.
• **Local GRISes** run on CEs and SEs at each site and report dynamic and static information regarding the status and availability of the services

   `ldapsearch -x -h <hostname> -p 2135 -b "mds-vo-name=local,o=grid"

   At each site, a **site GIIS** collects the information of all resources given by the GRISs

   `ldapsearch -x -h <hostname> -p 2135 -b "mds-vo-name=<name>,o=grid"

• Each site can run a **BDII**

   It collects the information coming from the GIISs and collects it in a data base

   `ldapsearch -x -h <hostname> -p 2170 -b "o=grid"`
The GLUE (Grid Laboratory Uniform Environment) information model is an information schema initially developed in 2002 to foster interoperation between Grids.

GLUE itself is a collection of attributes with a name, multiplicity, type and description of the content.

```plaintext
objectClass ( 1.3.6.1.4.1.8005.100.4.1.1
    NAME 'GlueSE'
    DESC 'Info for the Storage Service'
    SUP 'GlueSETop'
    STRUCTURAL
    MUST (GlueSEUniqueID
        MAY (GlueSEName $ GlueSEPort $ GlueSEHostingSL))
```
Types of monitoring systems

Based on the communication patterns between the sensors, the monitoring systems can be classified as using:

- Gossip mechanism
  - Multicast/multiple unicast between all/group of sensors/nodes

- Producer-Consumer mechanism
  - Node types differentiated as
    - Producer: Sensor/node which is the source of data (stream)
    - Consumer: This node uses the information
    - Mediator: Node which does aggregation etc. (typically is a combination of producer and consumer)
GMA

- The Grid Monitoring Architecture as proposed by the Global Grid Forum
- Proposes the Producer-Consumer model
- Includes a registry service for registration of producers
- Registry service supports match-making
R-GMA

- The relational-GMA by the European DataGrid
- Uses RDBMS to model the GMA
- Has different types of producers (stream, continuous stream,..)

Globus’ Monitoring and Discovery System (MDS)

“Classic” MDS Architecture (MDS-1)

Clients query organization server for current information.

- **Client 1**
- **Client 2**
- **Client 3**

**LDAP Organization Server**
Directory contains info from A and B

- **gram-reporter** Resource A
- **gram-reporter** Resource B

gram-reporters periodically update LDAP server’s information.
“Standard” MDS Architecture (MDS-2)

- Clients 1 and 2 request info directly from resources.
- Client 3 uses GIIS for searching collective information.
- Cache contains info from A and B

Source: http://www.globus.org/toolkit/mds/
Ganglia

- UC Berkley
- Gossip based mechanism
- leverages XML for data representation, XDR for compact, portable data transport, and RRDtool for data storage and visualization.
- Concept of gmond and gmetad
  - gmond: basic monitoring threads
  - gmetad: provides aggregation and hierarchical views
- Pros: Stable and widely used
- Cons: Uses multicast within a group and each node stores all the information of all nodes in the group

Source: [http://ganglia.sourceforge.net/](http://ganglia.sourceforge.net/)
Gems

- Grid Enabled Monitoring Service (GEMS)
- Gossip mechanism
- Concepts of grouping and layering
- Provides hierarchical views
- Pros
  - Scalable idea

Source: [http://www.hcs.ufl.edu/gems/](http://www.hcs.ufl.edu/gems/)
MonALI SA

Source: http://monalisa.caltech.edu/
Astrolabe

- Decentralized hierarchical database
- A table for each zone
  - For example, the "/USA/Cornell/pc3" agent creates the "/", "/USA", and "/USA/Cornell" zones if they did not exist already.
- Associated with each zone is an attribute list which contains the information associated with the zone.
- Propagates information using gossip
- If the two agents are in the same zone, the state exchanged relates to MIBs in that zone; if they are in different zones, they exchange state associated with the MIBs of their least common ancestor zone.
NWS

- UCSB
- Producer-Consumer mechanism
- One of the first monitoring systems to do some forecasting
  - though the forecasting techniques have not been integrated into any current grid toolkits
Data Management

- Grids are usually used for analyzing and manipulating large amounts of data.
- Data management services provide a flexible mechanism to move and share data.
- Ex: GridFTP, RLS (Replica Location Service)
Data Management

- Motivating factors
  - Large number of data files
    - Distributed across different sites
  - Big files!
    - ~TB, PB??
- Data is a first-class citizen in the Grid!
Data Management

- So what are the issues?
  1. **What** is the data?
  2. **Where** are the data files?
  3. **How** to move them?
Issues in Data Management

- Lets tackle them in a **bottom-up** fashion:
  1. **What** is the data?
  2. **Where** are the data files?
  3. **How** to move them?
Data Movement

- Issues
  - How to move data
    - Robustly
    - Securely
    - Faster

- Solutions
  - GridFTP
globus-url-copy

- GridFTP-compliant client from the Globus team
- Copy files from one location to another
  - The locations can be URLs, typically a gsiftp:// URL or a file:// URL
To move a file from remote GridFTP-enabled server to local machine

$globus-url-copy file:/home/lchitnis/file1
gsiftp://ufgrid.phys.ufl.edu/data/file1

Similarly, to transfer the file from local machine to remote server, just reverse the URLs

$globus-url-copy file:/home/lchitnis/file2
gsiftp://ufgrid.phys.ufl.edu/data/file2
Performance tuning

- You can monitor the performance of the transfer using `-vb` flag

  ```
globus-url-copy -vb file:/tmp/file1
gsiftp://ufgrid.phys.ufl.edu/ppadala/grid/file1
  543899 bytes 710.19 KB/sec avg 512.95 KB/sec inst
  ```

- By default only 1 data channel used
  - Multiple channels can be used to boost the transfer rate using the `-p` option

  ```
globus-url-copy -vb -p 4 file:/tmp/bigfile
gsiftp://ufgrid.phys.ufl.edu:15000/ppadala/grid/bfile
  765328908 bytes 4980.25 KB/sec avg 5201.24 KB/sec inst
  ```

Examples in this chapter are adapted from David Gehrig’s presentation on “Grid Data Management”
Faster transfers with globus-url-copy

- Use larger TCP windows to enhance the performance using the \(-tcp-bs\) option

```bash
$globus-url-copy -vb -p 4 -tcp-bs 524288
gsiftp://ufgrid.phys.ufl.edu:15000/lchitnis/verybigfile
file:/tmp/lchitnis/verybigfile
514392064 bytes      6609.67 KB/sec avg      8639.71 KB/sec inst
```

- Using large memory buffers further enhances the performance \((-bs flag)\)

```bash
$globus-url-copy -vb -p 4 -bs 524288 -tcp-bs 1048576
gsiftp://ufgrid.phys.ufl.edu:15000/lchitnis/largefile
file:/tmp/largefile
523304960 bytes      7300.56 KB/sec avg      9311.99 KB/sec inst
```
Issues in Data Management

- Let's tackle them in a bottom-up fashion:
  1. **What** is the data?
  2. **Where** are the data files?
  3. **How** to move them?
Where are the files?

- File catalogues
  - Replica Location Service (RLS)
  - Phedex
  - RefDB/PupDB
Requirements from a file catalogue

- Abstract out the logical file name (LFN) for a physical file
  - maintain the mappings between the LFNs and the PFNs (*physical file names*)
- Maintain the location information of a file
  - Or at least the information of the site on which it can be found in case of remote file
Data Replication

- In order to avoid "hotspots", the data files can be replicated to more than one locations
  - effective use of the grid resources
  - So, each LFN can effectively have more than 1 PFNs
- Replication avoids single point of failure
- Replication of data files can be done
  - Manually, or
  - Automatically – considering the demand for a file, the transfer bandwidth consideration, etc.
Each RLS server usually runs

- LRC
  - Local replica catalog
  - Catalog of what files you have (directly know physical location about) and mappings to URL(s) or PFNs (physical file names)

and/or

- RLI
  - Replica location index
  - Catalog of what files (LFNs) other LRCs know about
Network of RLS servers *inform* each other

- Each site has LRC with mappings of LFNs to PFNs
  - usually contains the “local” mappings
  - E.g. site A has the following mapping in its LRC

\[
X \rightarrow \text{gsiftp://dataserver.A:32768/fileX}
\]

- LRC catalog at each site tells remote RLIs what LFNs it has mappings for
  - E.g. Site A tells Site B that it has the location of X
  - So now Site B has the mapping

\[
X \rightarrow \text{rls://dataserver.A:32768}
\]
LRC of Site A reports to RLI of Site B

Site A

LRC


Site B

LRC

Y -> gsiftp://dataserver.B/fileY

RLI

Y -> rls://dataserver.B/
X -> rls://dataserver.A:32768/
Replication using RLS

- The Data file which needs to be replicated can be copied to the 2nd location, using GridFTP
  - E.g. file X is copied from Site A to a new Site C
- In the previous case, note that the LRC of Site A is programmed to report to RLI of Site B
- If the new site C is also set up to report to the RLI of Site B, site B will know both locations of file X
  - And so will any other LRCs/RLIs querying RLI on site B
LRC of Site C reports the (replicated) X to RLI of Site B.
RLS use-case scenario

Typical way to query RLS network and find files in your Grid

- Ask your local LRC "*Give me the PFN for the LFN X*"
  - If it knows about X, it answers "*The PFN of X is gsiftp://dataserver.A:32768/fileX*"
  - If it doesn’t, it returns an error code 1
- So now ask an RLI "*Which site has the PFN for file X?*
  - If it answers "*The RLS server at Site C knows about this file*"
    - Go ask the Site C RLS server "*I am told you know about the file X. Please tell me the URL for it*
    - It answers "*The PFN of X is gsiftp://dataserver.C/fileCx*"
RLS client

Two ways for clients to interact with RLS Server

- globus-rls-cli simple command-line tool
  - query
  - create new mappings

- Code your own client by coding against API
  - Java
  - C
  - Python
**RLS client**

- Simple query to LRC to find a PFN for LFN
  - Note that multiple PFNs may be returned for 1 LFN

$ globus-rls-cli query lrc lfn smallfile rls://dataserver.mysite
smallfile: file://mysite/tmp/smallfile1
smallfile: file://remotesite/data/smallfile2

- If LFN is not found, the return code is set to 1

$ globus-rls-cli query lrc lfn "nothere"
rls://dataserver.mysite
LFN doesn't exist: nothere
$echo $?
1
To obtain PFNs for more than one LFN at a time

$globus-rls-cli bulk query lrc lfn smallfile largefile rls://dataserver.mysite:27000
smallfile: gsiftp://dataserver.mysite/tmp/smallfile1
largefile: gsiftp://dataserver.remotesite:21000/data/largefile1

Wildcard search is supported

$ globus-rls-cli query wildcard lrc lfn "*file"
rls://dataserver.mysite:27000
smallfile: gsiftp://dataserver.mysite/tmp/smallfile1
largefile: gsiftp://dataserver.remotesite:21000/data/largefile1
RLS Queries

Simple query to RLI to locate a LFN to LRC map

$>globus-rls-cli query rli lfn myfile
rls://dataserver.mysite
myfile: rls://dataserver.remotesite:32768

then query that LRC for the PFN

$>globus-rls-cli query lrc lfn myfile
rls://dataserver.remotesite:32768
myfile: gsiftp://dataserver.remotesite:32768/data/yourfile1
Creation, addition and deletion of mappings

- Use `globus-rls-cli`
  - with **create** to create 1st mapping for a LFN
    
    ```
    $ globus-rls-cli create file1 gsiftp://dataserver/file1 rls://dataserver
    ```
  
  - with **add** to add more mappings for a LFN
    
    ```
    $ globus-rls-cli add file1 file://dataserver/file1 rls://dataserver
    ```

  - with **delete** to remove a mapping for a LFN
    
    ```
    $ globus-rls-cli delete file1 file://file1 rls://dataserver
    ```

  - when last mapping is deleted for a LFN the LFN is also deleted
  - cannot have LFN in LRC without a mapping
Issues in Data Management

- Let's tackle them in a bottom-up fashion:
  1. **What** is the data?
  2. **Where** are the data files?
  3. **How** to move them?
Metadata Catalogue

- Store data about...data!
- Aids in data-discovery
- A user wants to run transformations on a set of input files
  - However, the user might not be knowing the exact LFNs assigned to the files
  - But he knows some properties of the input
Metadata Catalogue

- One scenario useful in a Data Grid
  - data generated/collected into files at some detector site
  - *location* of data files published into RLS
    - `datafile5` -> `gsiftp://dataserver.exptsite/data/datafile5`
  - *existence* of data files and important metadata published into metadata catalog
    - `datafile` ->
      - data starts at GPS time 675370245
      - data is for the LHC experiment
      - conducted at CERN,
      - file contains 30 seconds of data
      - the calibration parameters are $\alpha = 11.345$ and $\beta = -1.356$
Data Management

Use-case scenario

To run an application that analyzes the data on the Grid

- Query metadata catalog for LFNs that contain data of interest
  - “Show me files where $10 < \alpha < 15$ and $\beta < -1$ for GPS times from 601024240 to 701024340”
    - datafile4
    - datafile5
    - datafile7
    - datafile8
Query RLI catalog to find out where those LFNs/files are known about

$ globus-rls-cli query rli lfn datafile4
  rls://dataserver
  datafile4: rls://dataserver.exptsite/

Query LRC catalog to get URLs for those files of interest

$ globus-rls-cli query lrc lfn datafile4:
  rls://dataserver.exptsite/
  datafile4:
    gsiftp://dataserver.exptsite/data/datafile4
Move files from storage to analysis site using GridFTP

$globus-url-copy -p 4
gsiftp://databaseserver.exptsite/data/datafile4
gsiftp://databaseserver.analysissite/input/input1
Other Grid Data Management Solutions

- Storage Resource Broker (SRB)
  - http://www.sdsc.edu/srb/
- Lightweight Data Replicator (LDR)
  - http://www.lsc-group.phys.uwm.edu/LDR
Job Management
Job Management Services

- Provide a uniform, standard interface to remote resources including CPU, Storage and Bandwidth
- Main component is the remote job manager
- Ex: GRAM (Globus Resource Allocation Manager)
Job Management on a Grid

Narration: note the different local schedulers
Submitting jobs with Globus Commands

- Authentication check
  - First, check that the gatekeeper “knows” you – has your Grid identity authorized to run jobs:
    - `$globusrun -a -r <sitename>/jobmanager-fork`
      - GRAM Authentication test successful
  - Look for the “test successful” message. This means you are “authorized” to use the `<sitename>“resource”.
  - If this still fails, there is a problem with your Grid certificate
Globus job submission

- A very simple way of submitting a job on the grid is with the command “globus-job-run”:
  - `gk1$ globus-job-run gk2 /bin/hostname`
  - `gk2`

- You’ve just submitted a “job” (the command “hostname”) to the GRAM gatekeeper on site `gk2`, from the “submit host” `gk1`!

- Trivial, perhaps, but a building block to more powerful capabilities.
Commands to run a job

- **globus-job-run**
  - globus-job-run runs in the foreground and defaults to sending output to your terminal.
  - In its basic form, it is roughly equivalent to rsh, but has considerably more functionality for running complex jobs on the Grid.

- **globus-job-submit**
  - globus-job-submit is for submitting jobs to a remote batch job scheduler such as LSF or the Portable Batch System (PBS).
  - With globus-job-submit, you can submit a job, log out, and log back in later to collect the output. That is, globus-job-submit runs in the background and defaults to sending output to the machine running the command.
  - Retrieve output with *globus-job-get-output*, and then clean up with *globus-job-clean*.
Commands to run jobs on grid

- **globusrun**
  - globusrun can run jobs either in the foreground or background, and can send output to your terminal or to the machine running the command.
  - The trend in Globus software development is toward considering globusrun as middleware, which can be used by application specific shell scripts to manage job submission.
  - In fact, globus-job-run and globus-job-submit are simply shell scripts that use globusrun for job submission, but present a simpler interface to users.
Managing your jobs

- We need something more than just the basic functionality of the globus job submission commands
- Some desired features
  - Job tracking
  - Submission of a set of inter-dependant jobs
  - Check-pointing and Job resubmission capability
  - Matchmaking for selecting appropriate resource for executing the job
- Lets have a look at some job management middleware...
Job management

- Various job management systems available that schedule jobs at cluster level
  - PBS
  - LSF
  - Condor
- We will consider Condor as a case study
Condor

- Condor has several unique mechanisms such as:
  - ClassAd Matchmaking
  - Process checkpoint/restart/migration
  - Remote System Calls

- Managing a large number of jobs
  - You specify the jobs in a file and submit them to Condor, which runs them all and keeps you notified on their progress
  - Mechanisms to help you manage huge numbers of jobs (1000’s), all the data, etc.
  - Condor can handle inter-job dependencies (DAGMan)
  - Condor users can set job priorities
  - Condor administrators can set user priorities
Condor-G

- Condor-G is a specialization of Condor. It is also known as the “Globus universe” or “Grid universe”.
- Condor-G can submit jobs to Globus resources, just like globus-job-run.
- Condor-G benefits from all the wonderful Condor features, like a real job queue.
- The gatekeeper doesn’t have to submit to a Condor pool.
  - It could be PBS, LSF,…
Personal Condor

- Keeps an eye on your jobs and will keep you posted on their progress
- Implements your policy on the execution order of the jobs
- Keeps a log of your job activities
- Adds fault tolerance to your jobs
- Implements your policy on when the jobs can run on your workstation
Starting with Condor

- Choosing a “Universe” for your job
  - Just use VANILLA for now
  - This isn’t a grid job, but almost everything applies, without the complication of the grid

- Make your job “batch-ready”

- Creating a `submit description` file

- Run `condor_submit` on your submit description file
Submit Description file

- A plain ASCII text file
- Tells Condor about your job:
  - Which executable, universe, input, output and error files to use, command-line arguments, environment variables, any special requirements or preferences (more on this later)
- Can describe many jobs at once (a “cluster”) each with different input, arguments, output, etc.
# Simple condor_submit input file
# (Lines beginning with # are comments)
# NOTE: the words on the left side are
# not case sensitive, but filenames are!
Universe   = vanilla
Executable = exe_name
Queue
You give *condor_submit* the name of the submit file you have created

*condor_submit* parses the file, checks for errors, and creates a “ClassAd” that describes your job(s)

Sends your job’s ClassAd(s) and executable to the *condor_schedd*, which stores the job in its queue

- Atomic operation, two-phase commit

View the queue with *condor_q*
condor_q

% condor_submit my_job.submit-file
Submitting job(s).
1 job(s) submitted to cluster 1.

% condor_q

-- Submitter: grinpc03.phys.ufl.edu : <128.105.165.34:1027> :  
ID      OWNER            SUBMITTED     RUN_TIME ST PRI SIZE  
CMD
1.0  juin           6/16 06:52   0+00:00:00 I  0   0.0  
my_job

1 jobs; 1 idle, 0 running, 0 held

%
Condor_rm

- If you want to remove a job from the Condor queue, you use `condor_rm`
- You can only remove jobs that you own (you can’t run `condor_rm` on someone else’s jobs unless you are root)
- You can give specific job ID’s (cluster or cluster.proc), or you can remove all of your jobs with the “-a” option.
Temporarily halt a Job

- Use *condor_hold* to place a job on hold
  - Kills job if currently running
  - Will not attempt to restart job until released
- Use *condor_release* to remove a hold and permit job to be scheduled again
Condor’s Process Checkpointing mechanism saves all the state of a process into a checkpoint file
- Memory, CPU, I/O, etc.

The process can then be restarted *from right where it left off*.

Typically no changes to your job’s source code needed – however, your job must be *relinked* with Condor’s Standard Universe support library.
Globus universe jobs

```
universe = globus
globusscheduler =
citgrid3.citcms.cacr.edu/jobmanager
executable = exe_name
queue
```

- Submits job to a remote cluster
- Remote job manager can be any cluster scheduler
  - PBS
  - LSF
- Authentication using globus infrastructure (X509 certificates)
Condor-G matchmaking

- With matchmaking, globus universe jobs can use requirements and rank:
  
  Executable = foo
  Universe = globus
  Globusscheduler = $$\text{(GatekeeperUrl)}$$
  Requirements = arch == LINUX
  Rank = NumberOfNodes
  Queue

- The $$x$$ syntax inserts information from the target ClassAd when a match is made.

- Where do these target ClassAds representing Globus gatekeepers come from? Several options:
  - Simple script on gatekeeper publishes an ad via `condoradvertise` command-line utility
  - Program to query Globus MDS and convert information into ClassAd (method used by EDG)
Virtual Data System
Virtual Data System

- What must we “virtualize” to compute on the Grid?
  - Location-independent computing: represent all workflow in abstract terms
  - Declarations not tied to specific entities:
    - Sites
    - Physical file names
    - Schedulers
Need for VDS

- For the process that is to be executed or the transformation that is to be applied, we need to ask the following questions:
  - What is to be done?
  - How it is to be done?
  - Where is it to be done?

- Virtual Data System allows us to tackle what is to be done in a transparent manner.
Virtual data
- defining data by the logical workflow needed to create it virtualizes it with respect to location, existence, failure, and representation

VDL – Virtual Data Language
- A language (text and XML) that defines the functions and function calls of a virtual data workflow
The Workflow

- Consider the following set of operations on a dataset:

  data1 → Op1 → data2 → Op2 → data3

Some slides in this chapter are adapted from Mike Wilde’s presentation on “Virtual Data Concepts”
The workflow

TR op1 (in a1, out a2) {
    argument stdin = ${a1};
    argument stdout = ${a2};
}

TR op2 (in a1, out a2) {
    argument stdin = ${a1};
    argument stdout = ${a2};
}

DV dv_1 -> op1 (a1=@{in:data1}, a2=@{out:data2});
DV dv_2 -> op2 (a1=@{in:data2}, a2=@{out:data3});
Understanding the workflow

TR op1 (in a1, out a2) {
    argument stdin = ${a1};
    argument stdout = ${a2};
}

TR op2 (in a1, out a2) {
    argument stdin = ${a1};
    argument stdout = ${a2};
}

DV dv_1 -> op1 (a1=@{in:data1}, a2=@{out:data2});
DV dv_2 -> op2 (a1=@{in:data2}, a2=@{out:data3});
Understanding VDL

- Transformations can be thought of as a function wrapper for an application
  - The arguments can be thought of as ‘formal’

- Derivations are the “calls” to the function
  - with actual parameters replacing the formal parameters
Abstract Workflow Generation

- Such a VDL file on “compilation” (using something like vdlc), generates the abstract workflow (.dax file)

- Note that specifics are still missing
  - such as the physical file name,
  - the execution site
“Concrete” workflow

The abstract workflow, when filled in with specific details, can be called the concrete workflow

- Details such as execution site, executable, I/O
- Other services can be utilized in this process
  - RLS for resolving file names

This concrete workflow is usually handed over to the scheduler for efficient execution scheduling

- If used with Condor Dagman, it is usually in the form of .dag file
Executing VDL Workflows

Workflow specification

VDL Program → Virtual Data catalog → Virtual Data Workflow Generator → Abstract workflow
Executing VDL Workflows

**Workflow specification**

1. **VDL Program** → **Virtual Data catalog**
2. **Virtual Data catalog** → **Virtual Data Workflow Generator**
3. **Virtual Data Workflow Generator** → **Abstract workflow**
4. **Abstract workflow** → **Planner**
5. **Planner** → **Create Execution Plan**

- **DAGman**
  - **DAG**
  - *(Concrete workflow)*
Executing VDL Workflows

**Abstract**

VDL (Virtual Data Language) workflows are generated by a Virtual Data Workflow Generator, which takes a VDL program as input. The abstract workflow is then created using Planner. DAGman and Condor-G are used for grid workflow execution. For job execution on Grid sites, Condor-G is employed.

**Workflow specification**

- VDL Program
  - Virtual Data catalog
  - Virtual Data Workflow Generator
  - Abstract workflow
  - Planner
- DAGman and Condor-G
  - For job execution on Grid sites
Grid Scheduling
The Problem of Grid Scheduling

- Decentralised ownership
  - No one controls the grid

- Heterogeneous composition
  - Difficult to guarantee execution environments

- Dynamic availability of resources
  - Ubiquitous monitoring infrastructure needed

- Complex policies
  - Issues of trust
  - Lack of accounting infrastructure
  - May change with time
A Real Life Example

Merge two grids into a single multi-VO “Inter-Grid”

- How to ensure that
  - neither VO is harmed?
  - both VOs actually benefit?
  - there are answers to questions like:
    - “With what probability will my job be scheduled and complete before before my conference deadline?”

- Clear need for a scheduling middleware!
Emerging Challenge: Intra-VO Management

- Today’s Grid
  - Few Production Managers

- Tomorrow’s Grid
  - Few Production Managers
  - Many Analysis Users

- How to ensure that
  - “Handles” exist for the VO to “throttle” different priorities
    - Production vs. Analysis
    - User-A vs. User-B
  - The VO is able to
    - “Inventory” all resources currently available to it over some time period
    - Strategically plan for its own use of those resources during that time period
Quality of Service

- For grid computing to become economically viable, a Quality of Service is needed
  - “Can the grid possibly handle my request within my required time window?”
  - If not, why not? When might it be able to accommodate such a request?
  - If yes, with what probability?
But, grid computing today typically:
- Relies on a “greedy” job placement strategies
  - Works well in a resource rich (user poor) environment
  - Assumes no correlation between job placement choices
- Provides no QoS

As a grid becomes resource limited,
- QoS becomes more important!
- “greedy” strategies not always a good choice
  - Strong correlation between job placement choices
Some Requirements for Effective Grid Scheduling

- Information requirements
  - Past & future dependencies of the application
    - Persistent storage of workflows
  - Resource usage estimation
  - Policies
    - Expected to vary slowly over time
  - Global views of job descriptions
  - Request Tracking and Usage Statistics
    - State information important
Requirements for effective Grid Scheduling (cont..)

- Resource Properties and Status
  - Expected to vary slowly with time
- Grid weather
  - Accurate and pertinent parameter reporting important
- Replica management

- System requirements
  - Distributed, fault-tolerant scheduling
  - Customisability
  - Interoperability with other scheduling systems
  - Quality of Service

Merci

Slides for requirements for scheduling and SPHINX
Prepared by Rick Cavanaugh for CHEP 2004
Incorporate Requirements into a Framework

Assume the GriPhyN Virtual Data Toolkit:
- Client (request/job submission)
  - Globus clients
  - Condor-G/DAGMan
  - Chimera Virtual Data System
- Server (resource gatekeeper)
  - MonALISA Monitoring Service
  - Globus services
  - RLS (Replica Location Service)
Incorporate Requirements into a Framework

Framework design principles:

- Information driven
- Flexible client-server model
- General, but pragmatic and simple
  - Implement now; learn; extend over time
- Avoid adding middleware requirements on grid resources
  - Take what is offered!

Assume the GriPhyN Virtual Data Toolkit:

- Client (request/job submission)
  - Globus clients
  - Condor-G/DAGMan
  - Chimera Virtual Data System
- Server (resource gatekeeper)
  - MonALISA Monitoring Service
  - Globus services
  - RLS (Replica Location Service)
The Sphinx Framework

Sphinx Server

Data Warehouse

Request Processing

Data Management

Information Gathering

Sphinx Client

Chimera Virtual Data System

Condor-G/DAGMan

VDT Client

Globus Resource

Replica Location Service

MonALISA Monitoring Service

VDT Server Site

Clarens

WS Backbone
Sphinx Scheduling Server

- Functions as the Nerve Centre
- Data Warehouse
  - Policies, Account Information, Grid Weather, Resource Properties and Status, Request Tracking, Workflows, etc
- Control Process
  - Finite State Machine
    - Different modules modify jobs, graphs, workflows, etc and change their state
  - Flexible
  - Extensible
Pegasus

- Pegasus – Planning for Execution in Grid
- Pegasus is a configurable system that can plan, schedule and execute complex workflows on the Grid.
  - Algorithmic and AI based techniques are used.
- Pegasus takes an abstract workflow as input. The abstract workflow describes the transformations and data in terms of their logical names.
- It then queries the Replica Location Service (RLS) for existence of any materialized data. If any derived data exists then it is reused and a workflow reduction is done.

Merci

Slides on Pegasus adapted from a presentation by Gaurang Mehta in CondorWeek2004
Pegasus (cont)

- It then locates physical locations for both components (transformations and data)
  - Uses Globus Replica Location Service (RLS) and the Transformation Catalog (TC)
- Finds appropriate resources to execute
  - Via Globus Monitoring and Discovery Service (MDS)
- Adds the stage-in jobs to transfer raw and materialized input files to the computation sites.
- Adds the stage out jobs to transfer derived data to the user selected storage location.
  - Both input and output staging is done Globus GridFtp
- Publishes newly derived data products for reuse
  - RLS, Chimera virtual data catalog (VDC)
Current Pegasus system

- Current Pegasus implementation plans the entire workflow before submitting it for execution. (Full ahead)
- Deferred planning in progress
Pegasus (cont)

- Pegasus generates the concrete workflow in Condor Dagman format and submits them to Dagman/Condor-G for execution on the Grid.

- These concrete Dags have the concrete location of the data and the site where the computation is to be performed.

- Condor-G submits these jobs via Globus-Gram to remote schedulers running Condor, PBS, LSF and Sun Grid Engine.