ICENI: An Integrated Grid Environment for Component Applications

Steven Newhouse

Technical Director
London e-Science Centre
Imperial College, London

Grid2001 - 12th November 2001

Contents

• The Grid and e-Science
• ICENI – Imperial College E-Science Networked Infrastructure
• The need for higher-level meta-data
• Exploiting the meta-data
• Component Based Linear Solver
• Summary & Acknowledgements
From Computational Grids to e-Science

- The Grid: complex networked resources
  - Don’t know which resources will be available and when
  - Potentially fragile resource connectivity and availability
  - Link instruments / storage / computation / HPC
- E-Science: higher-level applied use of these resources
  - Complex applications have to retain performance
    - Analysis of very large distributed data sets
    - Coupled execution for multi-physics or visualisation
  - Ease of use for domain specialists
    - For the Developer – higher level abstractions
    - For the User – higher level interaction, e.g. portals, PSE’s
  - Better utilisation of resources and improve throughput
  - Support scientific collaboration and knowledge management

Building Computational Communities

- Need to federate resources from real organisations
  - Express ownership and retain control
  - Grid resources are ‘not’ free – accountability / payment!
  - Manage the resources as a single unit
- Computational Communities (a.k.a. Virtual Organisations)
  - Share and combine heterogeneous resources
  - Ensure transparency by hiding complexity from the user
  - Optimise resource utilisation for all jobs and users
  - Easy to contribute (and withdraw) resources
- Collect and retain information relating to:
  - The resources
  - The applications
  - The users
ICENI

The Iceni, under Queen Boudicca, united the tribes of South-East England in a revolt against the occupying Roman forces in AD60.

- IC E-Science Networked Infrastructure
- Developed by Grid Middleware Group, London E-Science Centre
- Collect and provide relevant Grid meta-data
- Use to define and develop higher-level services

Open Extensible Technologies

- XML used to define:
  - Resource, application, user meta-data
  - Protocol between different services in the system
- Java used to construct the framework, run-time representation and interfaces to the assembled components
- Jini used to provide a wide-area transport layer that supports dynamic discovery and join
- Examining SOAP, WSDL, UDDI, .NET, SunONE etc.
Need Higher Level Knowledge

- Resources:
  - Availability, capability, environment, access

- Application:
  - Composition, behaviour, performance
  Related talk: Anthony Mayer,
  Thursday 2.00, A201/205

- User:
  - Who, what, when, where

Managing Internal Resources

- The resources advertise their capabilities to a private domain
- Extensible resources abstractions:
  - Computational resources
  - Storage resources
  - Software resources
  - ...
- Annotate using XML
The Domain Manager

Sole route between the private and the public areas of the infrastructure

- Private ➔ Public
  - Imposes local usage requirements and access
  - and publicly advertises its resources

- Public ➔ Private
  - Authenticates requests to use the local resources (through the Identity Manager)

Trusting Organisations & Users

- Recognise three entities: individuals, groups and organisations (or domains)
- Entities verified through the Identity Manager using public key certificates (X.509)
- Locally managed access control list determine which entities have access to local resources
- Non-local users can be mapped to individual, single, or guest accounts (resource dependant policy)
Interaction with the Computational Community

The Computational Community allows Users & Domain Managers to advertise their capabilities and requirements.

The Resource Browser GUI
Resources in a Computational Community

A resource might be:

- **Available** → shown with all its public attributes and access policies. It’s possible to get the values of the dynamic attribute(s) and to connect to the resource.

- **Temporary unavailable** → only the access policies are shown.

- **Always unavailable** → no information is shown.

ICENI Architecture

[Diagram showing the ICENI Architecture with various components such as Identity Manager, Domain Manager, Policy Manager, Resource Manager, and resource classes such as CR (Computational Resource) and SR (Software Resource).]
Higher Levels of Abstraction

- **Developer**
  - Create numerical implementations
  - Provide performance & implementation meta-data
  - Extensible abstract objects or interfaces
- **Scientist**
  - Add domain specific knowledge to implementations
  - Define (new?) component interfaces
  - Specify component meta-data
- **End-User**
  - Customise component instances
  - Application defined by interacting component instances
  - Use components in local (or remote) repositories

Separating Meta-Information from Implementation
User Information

- **Security**
  - Need secure trusted identity for use on the Grid
  - Need to define what it can be used for
- **Application**
  - Develop through visual programming
  - Drag & drop functionality from elsewhere
- **Policy**
  - Restrict where the application runs
- **Accounting**
  - Which project (account) and how much?
  - Defining resource exchange rates
Exploiting The Meta-data

- The Computational Community contains information:
  - On resource capability, usage and access policy
  - On the user’s job and their application’s capability
- Uses this information to ensure:
  - The user’s job is run as specified (e.g. deadline)
  - All resources are fully utilised (if possible)
  - The mapping of jobs to resources is ‘optimal’

Higher-level Grid Services

- Application Mapper
  - Uses resource and application information to optimise resource selection for an application
    (e.g. are 16 PC processors better than 8 Alpha processors?)
- Resource Broker
  - Uses computational economics to balance priorities
    (e.g. 16 underused PC processors may cost less for longer than 8 over subscribed Alpha processors)
  - Factor queue length into resource selection
    (e.g. Will a slow unloaded resource provide shorter job turnaround time than a faster but heavily loaded resource?)
Example: Linear Solver

Local Computational Community

<table>
<thead>
<tr>
<th>System</th>
<th>Processor</th>
<th>Number</th>
<th>Language</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>PC (Linux)</td>
<td>AMD 900MHz</td>
<td>1</td>
<td>Java + C LAPACK</td>
<td>LU + BCG</td>
</tr>
<tr>
<td>Atlas</td>
<td>Alpha 667MHz</td>
<td>1,4,9</td>
<td>ScaLAPACK</td>
<td>LU + BCG</td>
</tr>
<tr>
<td>AP3000</td>
<td>UltraSparcII 300MHz</td>
<td>4,9,16</td>
<td>ScaLAPACK</td>
<td>LU + BCG</td>
</tr>
</tbody>
</table>
## Dynamic Resource Cost

<table>
<thead>
<tr>
<th>System</th>
<th>Cost Model 1</th>
<th>Cost Model 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>PC (Linux)</td>
<td>80</td>
<td>80</td>
</tr>
<tr>
<td>Atlas</td>
<td>475</td>
<td>425</td>
</tr>
<tr>
<td>AP3000</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>

## Influence of Resource Selection Policy on Execution Times

![Graph showing execution times and resource selection policies](image-url)
Cost savings relative to minimum time

Layered Grid Architecture

<table>
<thead>
<tr>
<th>Application Toolkits</th>
<th>Grid Services</th>
<th>Grid Fabric</th>
<th>Grid Resources</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlobusView</td>
<td>Nexus</td>
<td>Condor</td>
<td>LSF</td>
</tr>
<tr>
<td>DUROC</td>
<td>GRAM</td>
<td>MPICH</td>
<td>PBS</td>
</tr>
<tr>
<td>MPI</td>
<td>GridFTP</td>
<td>HPC++</td>
<td>NQE</td>
</tr>
<tr>
<td>Condor-G</td>
<td>GSI</td>
<td>Nimrod-4</td>
<td>Linux</td>
</tr>
<tr>
<td>HPC++</td>
<td>GSI-FTP</td>
<td>GLOBUS-600</td>
<td>NT</td>
</tr>
<tr>
<td>Nimrod-4</td>
<td>HBM</td>
<td>GRAM</td>
<td>Sanders</td>
</tr>
<tr>
<td>Globus-600</td>
<td>GASS</td>
<td>GRAM</td>
<td>DiffServ</td>
</tr>
</tbody>
</table>

Tested Status: globuscribe
Exploit Layered Architecture

Application Toolkits

ICENI

Grid Services

Grid Fabric

Integration with the Grid

Portals (JServlet)  HTC
Applied Portals

SRB  Globus  Java  CoG

Globus  CoG  Grid Fabric

Grid Services

ICENI

Application Toolkits

Public Computational Community

Private Computational Community

Resource Browser

Gateway between private and public regions

Private  Public

Application Mapper

Resource Broker
Summary

- To enable the e-Scientist to fully exploit the resources in the Grid we need information relating to the:
  - Resources
  - Applications
  - Users
- Represent the meta-data in an open extensible manner using XML derived syntax.
- Collect and hold the meta-data in a Java/Jini Grid middleware which has inherent fault-tolerance.
- Allow higher level services to use this meta-data to transparently guide resource allocation within the computational community for defined goals.

Acknowledgements

- London e-Science Centre, Grid Middleware Group:
  - John Darlington, Steven Newhouse, Tony Field
  - Anthony Mayer, Nathalie Furmento, Stephen McGough,
  - James Stanton, Yong Xie
- Further information:
  - http://www-icpc.doc.ic.ac.uk/components/
  - http://www.lesc.ic.ac.uk/
- Contact: icpc-sw@doc.ic.ac.uk
- Funding: EPSRC GR/N13371
- Related talk: Anthony Mayer, Thursday 2.00, A201/205