Computational Grids
Chapter 2 of The Grid: Blueprint for a New Computing Infrastructure, by Ian Foster and Carl Kesselman

Presented by Adam Bazinet
Six Questions

• Why do we need computational grids?

• What types of applications will grids be used for?

• Who will use grids?

• How will grids be used?

• What is involved in building a grid?

• What problems must be solved to make grids commonplace?
Reasons for Computational Grids

• Computational approaches to problem solving have proven their worth

• The average computing environment remains substantively deficient, often not providing the user with enough computational power

• Grid computing hopes to ameliorate this situation and revolutionize the way computational needs are met
Increasing Delivered Computation - How?

- End systems will continue to increase their computational capacity
- Easy demand-driven access to computational resources will improve
- Better utilization of idle cycles, as computers sit idle (but on) much of the time
- Better sharing of computational results will result in more efficient and productive collaborations
- Such new technology may lead to innovative new problem-solving techniques
Definition of Computational Grids

• By analogy with the electrical grid circa 1910

• A computational grid: a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities.

• It is this combination of features that will have a transformative effect on how computation is performed.
Five Classes of Grid Applications

• Distributed Supercomputing

• High-Throughput Computing

• On-Demand Computing

• Data-Intensive Computing

• Collaborative Computing
Four Grid Communities

- Government
- A Health Maintenance Organization
- Materials Science Collaboratory
- Computational Market Economy
People Using Grids

- Grid Developers
- Tool Developers
- Application Developers
- End Users
- System Administrators
Grid Architecture

• System types, in terms of increasing scale:

  • *end system*

  • *cluster*

  • *intranet*

  • *internet*
**End Systems**

- Comprises computers, storage systems, sensors, and other devices
- Small in scale, highly homogeneous, highly integrated
- Simplest and most intensively studied environment in which to provide basic services
- Do not necessarily integrate easily into larger clusters, intranets, and internets
Clusters

• A network of workstations connected by a high-speed LAN

• Typically homogeneous

• Principal complicating factors:
  1) increased physical scale
  2) reduced integration

• Software architectures for clusters may converge with those for end systems, as end-system architectures address issues of network operation and scale.
Intranets

• A grid comprising a potentially large number of resources that nevertheless belong to a single organization

• Principal complicating factors:
  1) heterogeneity
  2) separate administration
  3) lack of global knowledge

• Somewhat increased need for security

• A challenge to coordinate resources and get good performance
Internets

• A grid of internetworked systems that span multiple organizations.

• Principal complicating factors:
  1) lack of centralized control
  2) geographical distribution
  3) international issues

• Security is of primary concern

• Policy issues between organizations become important
Research Challenges

• Many and varied:
  1) the nature of applications may change
  2) new programming models and tools
  3) system architecture
  4) algorithms and problem-solving methods
  5) resource management
  6) security
  7) instrumentation and performance analysis
  8) end systems
  9) network protocols and infrastructure
The Lattice Project
A Computational Grid System

Presented by Adam Bazinet
The Lattice Project

• The Lattice Project is primarily aimed at effectively sharing computational resources between departments and institutions, starting with those in the University System of Maryland.

• The Grid is focused on computation, and we have not yet made efforts to enable large-scale data access, storage, or replication.

• The Grid has transitioned from a research project which started in 2003 to a production system that has been used by a number of researchers, racking up many hundreds of CPU years in the process.
Grid Software

• The backbone of the Grid system is Globus Toolkit software, which provides mechanisms for job submission, file transfer, and authentication and authorization of entities on the Grid.

• The most novel feature of The Lattice Project is our Globus-BOINC interface, which enables Grid jobs to flow into a BOINC pool. The Lattice BOINC Project (http://boinc.umiacs.umd.edu) is our active BOINC project for this purpose. Anyone can participate.

• We also work with scheduling software that controls local resources. Our resource base is currently composed of Condor pools and clusters running variants of PBS.
Grid Architecture

1. Grid Client
   - Client invokes a Grid service with a particular set of arguments, thus submitting a job to the Grid.

2. Grid Service
   - Grid service submits job to Grid scheduler on behalf of client.

3. Grid Scheduler
   - Scheduler sends job to an appropriate Grid resource.

4. BOINC Server
   - BOINC clients download workunits from the BOINC server, compute, and return the results.

   BOINC Client Pool

   Cluster Master Node
   - A queueing system such as Condor or PBS distributes work to cluster nodes.

   Cluster Nodes

   SMP
Grid Services

• Applications are Grid-enabled and made into a Grid service. Such trusted applications are then made available to run on Grid resources.

• To date, we have Grid-enabled approximately 25 applications, mostly life science applications, with notable exceptions. Only a subset of these have been run a significant amount.

• We have developed a software stack that allows us to Grid-enable applications quickly and easily. We call this software GSBL (Grid Services Base Library) and GSG (Grid Services Generator).
User Interfaces

• Our primary Grid interface is command-line based. Grid users log on to a specific machine, upload their input data, and then submit and monitor jobs using our tools.

• We also provide a Web interface for monitoring job status, which is located on The Lattice Project intranet.

• Future work might see job submission and other operations take place via a Web interface or portal of sorts. We may also make the command-line interface more widely available.
Job Search

Search for a job or set of jobs containing any of the parameters below.

<table>
<thead>
<tr>
<th>Job ID</th>
<th>Job Name</th>
<th>Start Time</th>
<th>Finish Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Within the last</td>
<td>Within the last</td>
</tr>
<tr>
<td></td>
<td></td>
<td>days hours</td>
<td>days hours</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(☑ Show unfinished)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>User Name</th>
<th>Job Status</th>
<th>Grid Service</th>
<th>Resource</th>
</tr>
</thead>
<tbody>
<tr>
<td>gt4admin</td>
<td>Idle</td>
<td>CARLI</td>
<td>PBS</td>
</tr>
<tr>
<td>freed</td>
<td>Running</td>
<td>Complab</td>
<td>BOINC</td>
</tr>
<tr>
<td>hollym</td>
<td>Removed</td>
<td>MDIV</td>
<td>Condor</td>
</tr>
<tr>
<td>fushman</td>
<td>Finished</td>
<td>HMMPIam</td>
<td>Fork</td>
</tr>
<tr>
<td>pknut777</td>
<td>Failed</td>
<td>Gsi</td>
<td></td>
</tr>
<tr>
<td>gridtest</td>
<td>Retrieved</td>
<td>MARXAN</td>
<td></td>
</tr>
<tr>
<td>demo</td>
<td>Failed Retrieve</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Operating System</th>
<th>Scheduler</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTEL PPC</td>
<td>LINUX WIN OSX</td>
<td><a href="https://128.8.115.115:8443/">https://128.8.115.115:8443/</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td><a href="https://128.8.120.35:8443/">https://128.8.120.35:8443/</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td><a href="https://131.118.1.37:8443/">https://131.118.1.37:8443/</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td><a href="https://128.8.10.42:8443/">https://128.8.10.42:8443/</a></td>
</tr>
</tbody>
</table>
### Job Status

This page is intended to help you monitor jobs that have been submitted to the Grid. Click the column header to sort the table.

<table>
<thead>
<tr>
<th>Job ID</th>
<th>Job Name</th>
<th>User ID</th>
<th>Grid Service</th>
<th>Job Status</th>
<th>Start Time</th>
<th>Finish Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>55318610.8734852526449487</td>
<td>NoDDCoblect122_batch1</td>
<td>lepusers</td>
<td>GARLI</td>
<td>Idle</td>
<td>2007-10-05 15:27:15</td>
<td>Not Finished</td>
</tr>
<tr>
<td>192021490.5563047721797734</td>
<td>NoDDCoblect122_batch4</td>
<td>lepusers</td>
<td>GARLI</td>
<td>Idle</td>
<td>2007-10-05 15:32:51</td>
<td>Not Finished</td>
</tr>
<tr>
<td>25274230.663135560834466</td>
<td>NoDDCoblect122_batch7</td>
<td>lepusers</td>
<td>GARLI</td>
<td>Idle</td>
<td>2007-10-05 15:38:34</td>
<td>Not Finished</td>
</tr>
<tr>
<td>300903930.4232310049235042</td>
<td>NoENOoblect122_batch1</td>
<td>lepusers</td>
<td>GARLI</td>
<td>Idle</td>
<td>2007-10-05 15:46:04</td>
<td>Not Finished</td>
</tr>
<tr>
<td>285851500.28441392065741633</td>
<td>NoENOoblect122_batch2</td>
<td>lepusers</td>
<td>GARLI</td>
<td>Idle</td>
<td>2007-10-05 15:47:59</td>
<td>Not Finished</td>
</tr>
<tr>
<td>319745870.7450886906716563</td>
<td>NoENOoblect122_batch5</td>
<td>lepusers</td>
<td>GARLI</td>
<td>Idle</td>
<td>2007-10-05 15:53:36</td>
<td>Not Finished</td>
</tr>
<tr>
<td>155012760.3862041884142373</td>
<td>NoENOoblect122_batch6</td>
<td>lepusers</td>
<td>GARLI</td>
<td>Idle</td>
<td>2007-10-05 15:55:25</td>
<td>Not Finished</td>
</tr>
<tr>
<td>120185750.6712593397034693</td>
<td>NoENOoblect122_batch7</td>
<td>lepusers</td>
<td>GARLI</td>
<td>Idle</td>
<td>2007-10-05 15:57:16</td>
<td>Not Finished</td>
</tr>
<tr>
<td>193890030.38653957104439485</td>
<td>NoENOoblect122_batch9</td>
<td>lepusers</td>
<td>GARLI</td>
<td>Idle</td>
<td>2007-10-05 16:01:08</td>
<td>Not Finished</td>
</tr>
<tr>
<td>125525010.27238711016874495</td>
<td>NoPERoblect122_batch1</td>
<td>lepusers</td>
<td>GARLI</td>
<td>Running</td>
<td>2007-10-06 20:27:50</td>
<td>Not Finished</td>
</tr>
<tr>
<td>67296680.7323232439232642</td>
<td>NoPERoblect122_batch2</td>
<td>lepusers</td>
<td>GARLI</td>
<td>Running</td>
<td>2007-10-06 20:33:57</td>
<td>Not Finished</td>
</tr>
<tr>
<td>54608600.7151218753272369</td>
<td>NoPERoblect122_batch6</td>
<td>lepusers</td>
<td>GARLI</td>
<td>Running</td>
<td>2007-10-06 20:58:26</td>
<td>Not Finished</td>
</tr>
<tr>
<td>120959170.9505456979838497</td>
<td>NoWGoblect122_batch2</td>
<td>lepusers</td>
<td>GARLI</td>
<td>Running</td>
<td>2007-10-06 21:35:29</td>
<td>Not Finished</td>
</tr>
<tr>
<td>263348430.7646646780565017</td>
<td>linsi_batch40</td>
<td>gridtest</td>
<td>GARLI</td>
<td>Running</td>
<td>2007-10-10 18:16:17</td>
<td>Not Finished</td>
</tr>
<tr>
<td>28788460.8681602501952815</td>
<td>linsi_batch41</td>
<td>gridtest</td>
<td>GARLI</td>
<td>Running</td>
<td>2007-10-10 18:38:47</td>
<td>Not Finished</td>
</tr>
<tr>
<td>169915760.6209327852586667</td>
<td>linsi_batch42</td>
<td>gridtest</td>
<td>GARLI</td>
<td>Running</td>
<td>2007-10-10 18:40:51</td>
<td>Not Finished</td>
</tr>
<tr>
<td>2812810.01971095155635405</td>
<td>linsi_batch43</td>
<td>gridtest</td>
<td>GARLI</td>
<td>Running</td>
<td>2007-10-10 18:42:57</td>
<td>Not Finished</td>
</tr>
<tr>
<td>111714820.841011800995614</td>
<td>linsi_batch44</td>
<td>gridtest</td>
<td>GARLI</td>
<td>Running</td>
<td>2007-10-10 18:45:07</td>
<td>Not Finished</td>
</tr>
<tr>
<td>Cluster</td>
<td>CPU Years</td>
<td>Lattice Jobs</td>
<td>Arch. &amp; OS</td>
<td>Idle</td>
<td>Running</td>
<td>Free CPUs</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------</td>
<td>--------------</td>
<td>------------</td>
<td>------</td>
<td>---------</td>
<td>------------</td>
</tr>
<tr>
<td>UMIACS Condor Pool</td>
<td>24.29</td>
<td>Lattice Jobs</td>
<td>Intel Linux</td>
<td>0</td>
<td>0</td>
<td>154</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Intel Win</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Sun4u Solaris28</td>
<td>0</td>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td>Gridiron Condor Pool</td>
<td>25.69</td>
<td>Lattice Jobs</td>
<td>Intel Linux</td>
<td>0</td>
<td>0</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Intel Win</td>
<td>0</td>
<td>0</td>
<td>87</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PPC OSX</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>CLFS Condor Pool</td>
<td>8.62</td>
<td>Lattice Jobs</td>
<td>Intel Linux</td>
<td>0</td>
<td>0</td>
<td>78</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Intel Win</td>
<td>0</td>
<td>0</td>
<td>36</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PPC OSX</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>Terpcondor Condor Pool</td>
<td>105.40</td>
<td>Lattice Jobs</td>
<td>Intel Linux</td>
<td>0</td>
<td>0</td>
<td>197</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Intel Win</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Lattice on BOINC</td>
<td>820.32</td>
<td>Lattice Jobs</td>
<td>Intel Linux</td>
<td>900</td>
<td>2500</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PPC OSX</td>
<td>0</td>
<td>0</td>
<td>33</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Intel Win</td>
<td>0</td>
<td>0</td>
<td>503</td>
</tr>
<tr>
<td>USM Condor Pool</td>
<td>0.14</td>
<td>Lattice Jobs</td>
<td>Intel Linux</td>
<td>0</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Intel Win</td>
<td>0</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Sun4u Solaris28</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Deepthought</td>
<td>3.15</td>
<td>Lattice Jobs</td>
<td>Intel Linux</td>
<td>0</td>
<td>0</td>
<td>116</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SEIL</td>
<td>23.33</td>
<td>Lattice Jobs</td>
<td>Intel Linux</td>
<td>0</td>
<td>0</td>
<td>308</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PPC OSX</td>
<td>0</td>
<td>0</td>
<td>44</td>
</tr>
<tr>
<td>Bluegrit</td>
<td>1.56</td>
<td>Lattice Jobs</td>
<td>PPC Linux</td>
<td>0</td>
<td>0</td>
<td>44</td>
</tr>
</tbody>
</table>

**Total Lattice Jobs**: Idle 900, Running 2500

**Total Free CPUs**: Linux 681, Windows 828, Mac OS X 112, Solaris 9, Grand Total 1630
The Lattice BOINC Project

- BOINC - Berkeley Open Infrastructure for Network Computing

- A platform for volunteer computing (public computing) or desktop grid computing

- A thin client pulls down work from a server, crunches it, and returns the results

- Users are compensated in the form of credit (not redeemable for cash prizes)

- We are the first to use the framework as part of a general purpose Grid system

- A potentially huge and valuable resource
The Lattice Project

About The Lattice Project
- Background and Introduction

Join The Lattice Project
1. Rules and Policies [Please read first!]
2. Getting Started
3. Create Account
4. Download BOINC - we require a core client >= 5.5.1!
5. When prompted, enter http://boinc.umiacs.umd.edu/

Returning participants
- Your account - view stats, modify preferences
- Teams - create or join a team
- Applications

Community
- Participant profiles
- Message boards
- Questions and answers

Project totals and leader boards
- Top participants
- Top computers
- Top teams
- Other statistics

User of the Day
[AF>France>Ouest>Midi-Pyrenees] Sebastien
Je suis technicien de laboratoire et membre de l'Alliance

Status - The project currently has no work available

Server Status:
- Globus Server: running
- Scheduler: running
- File Server: running
- Feeder: running

Results:
- Workunits: 7167
- Results ready to send
- Results in progress: 0

News
October 10, 2007 Choosing Applications
Some had asked for the ability to choose which applications they run. Well, now you can! If your system meets the minimum requirements to run an application, you will decide what applications you want to run.

October 8, 2007 Server Upgrade
The server upgrade is essentially complete, save for one feature. Please report any bugs.

October 4, 2007 Research Projects
A new page describes types of analyses run on The Lattice BOINC Project, application listing work in progress! Head on over to Research Projects

Old News
More Information

• The Lattice Project web site: http://lattice.umiacs.umd.edu/

• The Lattice BOINC Project web site: http://boinc.umiacs.umd.edu/

• Look on the site for publications about The Lattice Project and previous presentations, among other things.

• Please sign up for the Lattice BOINC Project and support your local University researchers!

• Please contact me for more information: pknut777@umiacs.umd.edu
The Data Grid:
Towards an Architecture for the Distributed Management and Analysis of Large Scientific Datasets, by Ann Chervenak et al.

Presented by Adam Bazinet
Large Data Collections

- In diverse problem domains, the volume of data is enormous and growing.
  - global climate change
  - high energy physics
  - computational genomics
  - and on...

- Such repositories are often geographically distributed

- The combination of large datasets, widely distributed users and resources, and computationally intensive analysis results in demands not satisfied by any existing data management infrastructure.
The Data Grid

- An extension and specialization of general “Grid” infrastructure

- Guiding design principles:
  1) mechanism neutrality
  2) policy neutrality
  3) compatibility with Grid infrastructure
  4) uniformity of information infrastructure

- These four principles led to the development of a layered architecture for the data grid (pictured on the next slide)
Data Grid Architecture

Figure 1: Major components and structure of the data grid architecture
Core Data Grid Services

• Fundamental basic services: **data access** and **metadata access**.

• The data access service provides mechanisms for accessing, managing, and initiating third-party transfers of data stored in storage systems.

• The metadata access service provides mechanisms for accessing and managing information *about* data stored in storage systems.
Storage Systems

- Applications should be presented with a uniform view of data

- This is accomplished by creating a logical *storage system* which provides functions for manipulating *file instances*, the basic unit of information

- Such a storage system can be implemented on top of a Unix file system, an HTTP server, a network cache, or even a distributed file system

- The storage system associates a set of properties with the file instance, like name, size, access restrictions, and so on
Data Access

• An API describes possible operations on storage systems and file instances - remote requests to read files, write files, determine file size, transfer files

• Data grid considerations increase complexity:
  - access functions must go through the remote security environment
  - robust performance requires reservation capabilities

• Characterizations of network traffic can help storage systems work together with data grid applications to coordinate data movement

• Transactions must be robust and fault tolerant
Metadata

• There must be a means of publishing and accessing metadata, which is information about file instances, the contents of file instances, and storage systems.

• *application metadata* may describe the information content of a dataset, the circumstances under which the data was obtained, and/or information instructing applications how to process the data.

• *replica metadata* includes information for mapping file instances to particular storage system locations.

• *system configuration metadata* describes the fabric of the data grid itself, e.g., network connectivity, details about storage systems, etc.
The Metadata Service

• Queries are posed to a **metadata service** that includes a metadata *repository* or *catalog*, which associates the query characteristics with *logical files*

• Once such logical files have been identified, the replica manager uses replica metadata to locate the physical file instance to be accessed

• Besides integrating different approaches to metadata storage and representation, the service must operate efficiently and be robust to failure

• Therefore, the metadata service should be structured hierarchically and distributed in order to achieve scalability, be robust, and enable local control over data

• The metadata service is implemented as an LDAP distributed directory
Other Basic Services

• An authentication and authorization infrastructure that supports multi-institutional operation

• Resource reservation and co-allocation mechanisms for storage systems and other resources such as networks to support performance guarantees

• Performance measurements and estimation techniques for resources like storage systems, networks, and computers

• Instrumentation services that enable the end-to-end instrumentation of storage transfers and other operations
Higher-Level Data Grid Components: Replica Management

- The **replica management service** creates or deletes copies of file instances, or *replicas*, within specified storage systems.

- A replica is simply a user-asserted correspondence between two physical files.

- Replicas may be created because the new storage location offers better performance or availability to or from a particular location.

- Entries in the replica manager catalog correspond to logical files and possibly collections of logical files -- associated with each of these are one or more replicas or file instances.

- Replica manager does not determine when or where replicas are created -- such policy decisions are left to the application.
Higher-Level Data Grid Components: Replica Selection

- The **replica selection service** chooses a replica that provides the application with data access characteristics that optimize some performance measure.

- The selection process may initiate the creation of a new replica.

- May consider access to subsets of a file instance, if that is advantageous.

- Data selection with subsetting may exploit Grid-enabled servers to perform such processing as part of data management.
Implementation Experiences

• Catalog design for metadata and replica management to support two application demonstrations:
  1) climate modeling
  2) data visualization

• Information in an LDAP catalog is organized in a Directory Information Tree

• In the data visualization application, they found that a large number of objects slowed directory searches, so they added support for specifying collections of logical files
Data Grids Today

- Storage Resource Broker (SRB)
  http://www.sdsc.edu/srb/index.php/Main_Page

- Leading software infrastructure for a Data Grid Management System (DGMS)

- Other technologies and initiatives?