Large Scale Computing Future

*From “classical” Grids to Clouds*

Frédéric Desprez
LIP ENS Lyon / INRIA
GRAAL Research Team

In the forties … the central computer

- Feb., 14 1946
- ENIAC
- 18,000 tubes, 30 tons, 170 m²
- 2,000 tubes replaced every months by 6 technicians
Earth simulator (2002)

- First computer to reach the Teraflops (10^{12} flops)
- Target Application: CFD-Weather, Climate
- 640 NEC SX/6 (mod)
  - 5120 Vector CPUs
- 40TeraFlops (peak)
- $400 million
  - $20-$30/y maintenance
- Size of a large concert hall
- Homogeneous, Centralized, Proprietary, Expensive!

IBM Roadrunner (2008)

- First computer to reach the Petaflops (10^{15} flops)
- Roadrunner runs on
  - 6,948 dual-core AMD Opteron chips on IBM Model LS21 blade servers,
  - 12,960 Cell engines (same as PS3) on IBM Model QS22 blade servers.
- With 80 terabytes of memory, the Roadrunner system and is housed in 288 IBM BladeCentre racks occupying 6,000 square feet.
- 10,000 connections, both infiniband and gigabit Ethernet, with 57 miles of fibre–optic cable.
Google cluster 1997

Google Servers today

• 36 data centers containing
  > 800K servers

• 40 servers/rack
Electric Power Grids and Computational Grids

- Term coined by Ian Foster in the early 90s
- **Power Grid analogy**
  - Power producers: machines, software, networks, storage systems
  - Power consumers: user applications
- **Applications draw power from the Grid the way appliances draw electricity from the power utility**
  - Seamless, High-performance, Ubiquitous, Dependable
- **Why the Computational Grid is like the Electric Power Grid**
  - Electric power is ubiquitous
  - Don’t need to know the source of the power (transformer, generator) or the power company that serves it
- **Why the Computational Grid is different from the Electric Power Grid**
  - Wider spectrum of performance
  - Wider spectrum of services
  - Access governed by more complicated issues: Security, Performance

Credits: D. Petcu

Some books and a lot of hype!
Why Grids?

- First need: supercomputing at a national or international scale
- Large size problems (grand challenge problems) need a collaboration between several codes/supercomputing centers
- Always a need of more computing power, memory capacity, and disk storage
- The power of any single resource is always small compared to the aggregation of several resources
- Network connectivity increased fast!

Many available resources
- Many clusters
- Supercomputers
- Millions of PC and workstations connected
- Sharing or renting resources

Increasing complexity of applications
- Multi-scale
- Multi-disciplinary
- Huge data set produced
- Heterogeneity

GRID Computing

- Allowing communities (virtual organizations) to share distributed resources to face common goals without
  - Centralized control
  - Global knowledge
  - Trust relations

- Resources (supercomputers, visualization systems, sensors, instruments, people) are integrated through middleware systems that ease the use of this resources (or at least try to hide their complexity)
Elements of the Problem

- **Sharing resources**
  - Computers, storage, sensors, network, ...
  - Their share is always conditional: trust problems, access policies, negotiation, payment, ...

- **Coordinated problem solving**
  - More than “classical” client–server approach: data analysis, computation, distributed collaborations

- **Dynamic and multi-institutional Virtual Organizations (VOs)**
  - Overlap over classical organization structures
  - Small or large, static or dynamic

One example of a large VO: CERN’s Large Hadron Collider

1800 physicists, 150 institutes, 32 countries

100 PB of data around 2010; 50,000 CPUs?
Let’s try to classify Grids

- **Information**
  - Sharing knowledge

- **Storage Grid**
  - Large scale data management

- **Computing Grid**
  - Aggregating computing power

* « A distributed system is a collection of independent computers that appear to the users of the system as a single computer ».  
  A. Tanenbaum, Prentice-Hall, 1994

Huge Applications Needs
Increasingly complicated problems

- Biologists want to simulate thousands of molecular drug candidates to see how they would interact with specific proteins.

- Earth scientists keep track of the level of atmospheric ozone with satellite observations. For this task alone, they download, from space to ground, about 100 Gigabytes of raw images per day.

- High Energy Physics will soon produce about 10 Petabytes of data per year. This data will record the result of collisions of extremely energetic fundamental particles. Thousands of physicists in dozens of universities around the world will want to analyze this data.

- Unlocking the secrets of the human genome would be impossible without the computerized analysis of massive amounts of data, including the sequence of the three billion chemical units that comprise our DNA.

Credits: R. Stevens

Different problems to be solved on Grids

- Computer-centric problems
  - Need teraflops, lots of them!
  - The Grid combines large computational resources

- Data-centric problems
  - The Grid is used to collect, store and analyze data maintained in geographically distributed repositories, digital libraries, and databases

- Community-centric problems
  - Collaborative applications --enable and enhance human-to-human interactions.
  - Provide a “virtual shared space”
Large Hadron Collider (LHC)

- Higher energy collisions are the key to further discoveries of more massive particles (E=mc²)
- One particle predicted by theorists remains elusive: the Higgs boson
- The LHC is the most powerful instrument ever built to investigate elementary particles
- Beams of protons collision at an energy of 14 TeV with a 27 km circumference instrument.
- Data
  - 40 million collisions per second
  - After filtering, 100 collisions of interest per second
  - A Megabyte of data digitised for each collision = recording rate of 0.1 Gigabytes/sec
  - 1010 collisions recorded each year
  - = 10 Petabytes/year of data

![Diagram](Image)

Large Hadron Collider (LHC)

- There is a "bunch crossing" every 25 nsecs.
- There are 100 "triggers" per second
- Each triggered event is ~1 MByte in size

- Physicists work on analysis "channels".
- Each institute will have ~10 physicists working on one or more channels; data for these channels should be cached by the institute server

Credits: Harvey Newman, Caltech
**Large Synoptic Survey Telescope (LSST)**

- Ground-based 8.4-meter, 10 square-degree-field telescope
- will provide digital imaging of faint astronomical objects across the entire sky, night after night
- 3 gigapixels detector for wide field imaging
- LSST will cover the available sky every three nights, opening a movie-like window on objects that change or move on rapid timescales:
  - exploding supernovae, potentially hazardous near-Earth asteroids, and distant Kuiper Belt Objects.
  - images used to trace billions of remote galaxies and measure the distortions in their shapes produced by lumps of Dark Matter, providing multiple tests of the mysterious Dark Energy.
- Data
  - >30 TB of data/night!

**CFD and Medicine (arterial flow) – George Karniadakis– Brown**

- Strong relationship between blood flow pattern and formation of arterial disease such as atherosclerotic plaques
- Disease develops preferentially in separated and re-circulating flow regions such as vessel bifurcations
- 1D results feed 3D simulations, providing flow rate and pressure for boundary conditions
- Very clever multiscale approach
- Couples resources weakly in real time, but requires co-scheduling
- MPIg, partly supported by TeraGrid, used for intra-site and inter-site communications.

Credits: N. Wilkins–Diehr
• Goal is understanding earthquakes and to mitigate risks of loss of life and property damage.
• Spans the gamut from largest simulations to midsize jobs to huge number of small jobs.
• For largest runs (Cybershake), where they examine high frequency modes (short wavelength, so higher resolution) of particular interest to civil engineers, need large distributed memory runs using the Track2 machines at TACC, NICS. 2000–64,000 cores of Ranger, Kraken.
• To improve the velocity model that goes into the large simulations, need mid-range core counts jobs doing full 3-D tomography (Tera3D); DTF and other clusters (e.g. Abe); Need large data available on disk (100 TB)

Output is large data sets stored at NCSA, or SDSC’s GPFS, IRODS. Moving to DOE machine at Argonne. TG provided help with essential data transfer.

Excellent example of coordinated ASTA support- CUI (SDSC) and Urbanic (PSC) interface with consultants at NICS, TACC, &NCSA to smooth migration of code. Improved performance 4x.

Credits: N. Wilkins-Diehr

Google

• Single search query touches 700 to up to 1k machines in less than 0.25sec
• There are more than 200 Google File System clusters
• The largest BigTable instance manages about 6 petabytes of data spread across thousands of machines

• MapReduce is increasing used within Google
  • 29,000 jobs in August 2004 and 2.2 million in September 2007
  • Average time to complete a job has dropped from 634 seconds to 395 seconds
  • Output of MapReduce tasks has risen from 193 terabytes to 14,018 terabytes
  • Typical day will run about 100,000 MapReduce jobs each occupies about 400 servers
  • takes about 5 to 10 minutes to finish
Distributed Entertainment

Everquest

- 45 communal “world servers” (26 high-end PCs per server) supporting 430,000 players
- Real-time interaction, individualized database management, back channel communication between players
- Data management adapted to span both client PC and server to mitigate communication delays
- Game masters interact with players for real-time game management

Next generation Grids will include new technologies

New devices

- PDAs, sensors, cars, clothes, smart dust, smart bandaids, ...

Wired and Wireless

- HPWREN, Roadnet (Hans-Werner Braun, Frank Vernon et al.)
  - 45 Mbps between Mount Laguna telescope and SDSU, wireless access to Pala, Rincon, La Jolla Indian Reservations, etc.
  - Roadnet expanding to waterways, etc.
Overview of (some) Grid Platforms

The TeraGrid Strategy

- Building a distributed system of unprecedented scale
  - 40+ teraflops compute
  - 1+ petabyte storage
  - 10–40Gb/s networking

- Creating a unified user environment across heterogeneous resources
  - Single user support resources
  - Single authentication point
  - Common software functionality
  - Common job management infrastructure
  - Globally-accessible data storage

- Create a unified national HPC infrastructure that is both heterogeneous and extensible

Credits: N. Wilkins–Diehr
Diversity of Resources (not exhaustive)

- **Very Powerful Tightly Coupled Distributed Memory**
  - Ranger (TACC): Sun Constellation, 62,976 cores, 579 Tflops, 123 TB RAM
  - Kraken (NICS): Cray XTS, 66,048 cores, 608 Tflops, > 1 Ptflops in 2009

- **Shared Memory**
  - Cobalt (NCSA): Altix, 8 Tflops, 3 TB shared memory
  - Pople (PSC): Altix, 5 Tflops, 1.5 TB shared memory

- **Clusters with Infiniband**
  - Abe (NCSA): 90 Tflops
  - Lonestar (TACC): 61 Tflops
  - QueenBee (LONI): 51 Tflops

- **Condor Pool (Loosely Coupled)**
  - Purdue—up to 22,000 CPUs

- **Visualization Resources**
  - TeraDRE (Purdue): 48 node nVIDIA GPUs
  - Spur (TACC): 32 nVIDIA GPUs

- **Storage Resources**
  - GPFS-WAN (SDSC)
  - Lustre-WAN (IU)
  - Various archival resources

Credits: N. Wilkins-Diehr
Experimental Grids

They are complex systems:
  Large scale, Deep stack of complicated software

Several research issues:
  Security, Performance, Fault tolerance, Scalability, Load Balancing, Coordination, Message passing, Data storage, Programming, Algorithms, Communication protocols and architecture, Deployment, Accounting, etc.

How to test and compare?
  • Fault tolerance protocols
  • Security mechanisms
  • Networking protocols
  • etc.

Large scale distributed & parallel systems raise research issues but also methodological challenges
What about production platforms used for experimental purpose?

**No reproducible experimental conditions:**
- Scientific studies require reproducible experimental conditions

**Not designed for experiments:**
- Many researchers run short length, highly parallel & distributed algos
- Preparation and execution of experiments are highly interactive

**Not optimized for experiments:**
- Experimental platforms should exhibit a low utilization rate to allow researchers executing large collocated experiments

**Not reconfigurable:**
- Many projects require experiments on OS and networks,
- Some projects require the installation of specific hardware

→Nowhere to test networking/OS/middleware ideas, to measure real application performance

---

**Exploration tools**

![Diagram showing the spectrum of exploration tools from Model Protocol proof to Real systems, with log(realism) and log(cost & coordination) axes.](image)

- Math
- Simulation
- Emulation
- Instrument
- Real systems

**Model**
- Math
- Simulation
- Emulation
- Instrument
- Real systems

**Protocol proof**
- SimGrid
- MicroGrid
- GridSim
- Bricks
- NS, etc.

**Data Grid eXplorer**
- DAS3
- PlanetLab
- Yahoo Cloud
- OneLab, PANLab
- SensLab, etc.

**Emulab**
- Emulab

**Grid’5000**
- Grid’5000

**Instrument**
- EGEE
- DEISA
- TERAGRID
- NAREGI
Original Vision of Grid’5000: Analogy with Physics Instruments

The Cosmotron. This was the first accelerator in the world to send particles to energies in the billion electron volt, or GeV, region, 1953.

Grid’5000 key design decisions

- Offer to the users only very low level/variety of software functionalities (and responsibilities)
- Keep Grid’5000 software as simple (minimal) as possible to allow:
  - A robust platform,
  - A large variety of experiments,
  - Users developing and tuning their experimental environments
- Allow experiments at any level of the software stack
- Allow users to run their experiments on the same hardware several times
- Run experiments in an isolated way
- Keep safe Grid’5000 and Internet Security!
Grid’5000 software stack

- Deploy an experimental computing infrastructure to allow any kind of experiments on large scale distributed and parallel systems
- Experiments of any kind of grids (Virtual Supercomputer, Desktop Grid, ..) or Clouds

Grid’5000 experimental model

A highly controllable and reconfigurable experimental platform
Let users:

1) reserve experimental resources
2) create their software stack including injectors and probes
3) deploy their software environment on the reserved experimental resources and run their experiments
Grid’5000 network

**Strong involvement of Renater!**

- First version: 1Gbps (IP premium Qos)
- Sec. version: 10 Gbps (dedicated lambda)
- Third version: X10 Gbps dedicated lambdas + traffic isolation (users will be able to reserve network resources for their experiments)
FAIL: FAult Injection Language

- High-level language for fault scenario
- Fault-injection middleware
- Also suitable for stress testing and user simulation

Dynamic Energy Consumption Monitoring

- Separate system monitoring Grid’5000
- Controllable and external energy sensors
  --> on-line energy consumption measurements
  --> required for developing energy saving algorithms
Example 1: Accelerating Research

Solving large instances of combinatorial optimization problems attracts the attention of researcher because of its algorithmic challenge. It always requires a huge number of computational resources.

Many success stories in combinatorial optimizations:

- First:
  - solve the n-Queens problem for n=25, in 2005
- one of the most promising one, in 2008:
  - Grid’5000 was used to design and improve the algorithm (MOGO) used in the first computer victory against a professional Go player (5 Dan) on a 9x9 plate in the last Paris tournament! (it’s close to the Dan!)

Example 2: Drastically improve precision

Bronze Standard method addressing the issue of medical image algo. evaluation
- Application on estimation of the spatial rigid transformation between two images (convenient to align two different images of a same patient acquired separately).

- Complex workflow of computations on large number of data sets.
- Typically require 10s to 100s of 3D images pairs. 15 minutes per image pair.

- The application is executed with MOTEUR (workflow engine)
- Several degrees of parallelism are tested:

  Execution time (seconds)
Conclusion on Experimental Grids

- Experimental platforms (and observation instruments) are essential in the computer science methodology – like in other sciences!
- Grid’5000, DAS3 and PlanetLab were among the first computer science Instruments.
- Research in Cloud and Service computing needs experimental platforms too (see all the platforms announced recently)
- Having a real scale platform is not enough as a scientific tool
- There are some requirements:
  - Experiment isolation
  - Capability to reproduce experimental conditions
  - Flexibility through high degree of reconfiguration
  - The strong control of experiment preparation and running
  - Precise measurement devices
  - Deep on-line monitoring (essential to help observations understanding)
- Building such experimental platforms is very difficult!
- It’s important to share our experiences, as users, as developers of experimental platform software and as builders of experimental platforms

Volunteer Computing Platforms

- 1 billion PCs
  - 55% privately owned
  - most are on Internet
- If 100M participate:
  - > 100 PetaFLOPs, 1 Exabyte (10^18) storage
- Consumer products drive technology

Credits: D.H. Anderson, UC Berkeley
## Volunteer computing

<table>
<thead>
<tr>
<th>Project</th>
<th>start</th>
<th>where</th>
<th>area</th>
<th>#hosts</th>
</tr>
</thead>
<tbody>
<tr>
<td>GIMPS</td>
<td>1994</td>
<td></td>
<td>math</td>
<td>10,000</td>
</tr>
<tr>
<td>distributed.net</td>
<td>1995</td>
<td></td>
<td>cryptography</td>
<td>100,000</td>
</tr>
<tr>
<td>SETI@home I</td>
<td>1999</td>
<td>UCB</td>
<td>SETI</td>
<td>600,000</td>
</tr>
<tr>
<td>Folding@home</td>
<td>1999</td>
<td>Stanford</td>
<td>biology</td>
<td>200,000</td>
</tr>
<tr>
<td>United Devices</td>
<td>2002</td>
<td></td>
<td>commercial</td>
<td>200,000</td>
</tr>
<tr>
<td>CPDN</td>
<td>2003</td>
<td>Oxford</td>
<td>climate change</td>
<td>150,000</td>
</tr>
<tr>
<td>LHC@home</td>
<td>2004</td>
<td>CERN</td>
<td>physics</td>
<td>60,000</td>
</tr>
<tr>
<td>Predictor@home</td>
<td>2004</td>
<td>Scripps</td>
<td>biology</td>
<td>100,000</td>
</tr>
<tr>
<td>WCG</td>
<td>2004</td>
<td>IBM</td>
<td>biomedicine</td>
<td>200,000</td>
</tr>
<tr>
<td>Einstein@home</td>
<td>2005</td>
<td>LIGO</td>
<td>astrophysics</td>
<td>200,000</td>
</tr>
<tr>
<td>SETI@home II</td>
<td>2005</td>
<td>UCB</td>
<td>SETI</td>
<td>850,000</td>
</tr>
<tr>
<td>Rosetta@home</td>
<td>2005</td>
<td>U. Wash</td>
<td>biology</td>
<td>100,000</td>
</tr>
<tr>
<td>SIMAP</td>
<td>2005</td>
<td>T.U. Munich</td>
<td>bioinformatics</td>
<td>10,000</td>
</tr>
</tbody>
</table>

Total of BOINC-based projects: 660,000 participants, 1,000,000 hosts, 450 TeraFLOPS

Credits: D.H. Anderson, UC Berkeley

## What's different about volunteer computing?

- **Must attract and retain volunteers**
  - Credit
  - Community features
  - Easy installation; autonomic

- **Volunteers are unreliable**
  - one solution: redundant computing

- **Heterogeneous, dynamic resource pool**

Credits: D.H. Anderson, UC Berkeley
Volunteer Computing Systems

- Systems with many more processors than any single such machine that perform massively parallel computations: Volunteer Computing Systems
- Those are systems to which resource owners can (generously) contribute (idle) cycles
- The most famous: SETI@home
- Luckily, there are many interesting applications with no communication requirements: Embarrassingly parallel applications
  - Tons of independent tasks to do
  - Doesn’t really matter when the last one is done as long as I do some number of tasks per days
  - Perhaps it doesn’t even really matter if a few tasks do not complete
  - Examples:
    - Monte-Carlo Simulations
      - Do a bunch of simulations, take an average over “a lot” of samples
      - Computing Pi is the basic Monte-Carlo example
    - Example: Bioinformatics
      - I have a bunch of procaryotes genomes, and I want to do all pair-wise alignments
  - For such applications, everything under the sun could potentially be used to make some progress
    - Some processor will compute 1000 tasks, and some other will compute 1 task, but that 1 is still useful in the big scheme of things because #tasks >> #hosts

Credits: H. Casanova

Desktop “Grids”

- Instead of purchasing a cluster, one can reuse the otherwise wasted CPU cycles of desktop machines
- Either in people’s home
  - e.g., SETI@home, AIDS@home, etc.
  - but needs some incentive, which is non-trivial
- Or in an “enterprise”
  - easy to convince a CEO of potential savings
  - not quite “volunteer”
  - e.g., United Device

- Desktop Grid Client
  - Harvesting of “otherwise wasted CPU cycles”
  - Done via a “client” that runs on the user’s machine
  - The client monitors user activity to know when it can run the desktop grid application
    - e.g., as a screen saver

Credits: H. Casanova
SETI@home

- http://setiathome.berkeley.edu/

- As of 26/11/08:
  - 905,601 users
    - Organized in 54,470 teams
  - 2,139,856 registered computers
  - In 252 countries

- Computes 528 TFlop/sec
  - Would place it at about #3 on the Top500 list today
  - But of course it’s not amenable to “cluster” applications

Credits: H. Casanova
SETI@home architecture

SETI@Home

Centralized Master-worker architecture

SETI@home clients

- Fixed-rate data processing task
- Low bandwidth/computation ratio
- embarrassingly parallel application

Credits: H. Casanova

Number of internet users

TOP 20 COUNTRIES WITH HIGHEST NUMBER OF INTERNET USERS

<table>
<thead>
<tr>
<th>#</th>
<th>Country or Region</th>
<th>Population, 2009 EST</th>
<th>Users Latest Data</th>
<th>% Population (Penetration)</th>
<th>Growth 2000-2009</th>
<th>% of World Users</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>China</td>
<td>1,330,444,665</td>
<td>298,000,000</td>
<td>22.4 %</td>
<td>1,244.4 %</td>
<td>18.7 %</td>
</tr>
<tr>
<td>2</td>
<td>United States</td>
<td>304,228,257</td>
<td>227,190,969</td>
<td>74.7 %</td>
<td>138.3 %</td>
<td>14.2 %</td>
</tr>
<tr>
<td>3</td>
<td>Japan</td>
<td>127,288,419</td>
<td>94,000,000</td>
<td>73.8 %</td>
<td>99.7 %</td>
<td>6.9 %</td>
</tr>
<tr>
<td>4</td>
<td>India</td>
<td>1,147,988,888</td>
<td>81,000,000</td>
<td>7.1 %</td>
<td>1,520.0 %</td>
<td>6.1 %</td>
</tr>
<tr>
<td>5</td>
<td>Brazil</td>
<td>196,342,587</td>
<td>67,510,400</td>
<td>34.4 %</td>
<td>1,260.2 %</td>
<td>4.2 %</td>
</tr>
<tr>
<td>6</td>
<td>Germany</td>
<td>82,369,540</td>
<td>55,221,169</td>
<td>67.0 %</td>
<td>130.1 %</td>
<td>3.5 %</td>
</tr>
<tr>
<td>7</td>
<td>United Kingdom</td>
<td>60,943,912</td>
<td>42,752,600</td>
<td>71.8 %</td>
<td>164.1 %</td>
<td>2.7 %</td>
</tr>
<tr>
<td>8</td>
<td>France</td>
<td>62,150,775</td>
<td>40,658,353</td>
<td>65.7 %</td>
<td>360.7 %</td>
<td>2.0 %</td>
</tr>
<tr>
<td>9</td>
<td>Russia</td>
<td>140,702,094</td>
<td>38,000,000</td>
<td>27.0 %</td>
<td>1,125.6 %</td>
<td>2.4 %</td>
</tr>
<tr>
<td>10</td>
<td>Korea South</td>
<td>48,379,392</td>
<td>36,794,800</td>
<td>76.1 %</td>
<td>93.3</td>
<td>2.3 %</td>
</tr>
<tr>
<td>11</td>
<td>Spain</td>
<td>40,491,051</td>
<td>28,552,604</td>
<td>70.5 %</td>
<td>429.9 %</td>
<td>1.8 %</td>
</tr>
<tr>
<td>12</td>
<td>Italy</td>
<td>58,145,321</td>
<td>28,388,026</td>
<td>48.8 %</td>
<td>115 %</td>
<td>1.8 %</td>
</tr>
<tr>
<td>13</td>
<td>Mexico</td>
<td>105,555,400</td>
<td>37,400,000</td>
<td>24.9 %</td>
<td>510.2 %</td>
<td>1.7 %</td>
</tr>
<tr>
<td>14</td>
<td>Turkey</td>
<td>75,793,836</td>
<td>26,500,000</td>
<td>35.0 %</td>
<td>1,226.0 %</td>
<td>1.7 %</td>
</tr>
<tr>
<td>15</td>
<td>Indonesia</td>
<td>237,752,355</td>
<td>25,000,000</td>
<td>10.5 %</td>
<td>1,150.0 %</td>
<td>1.6 %</td>
</tr>
<tr>
<td>16</td>
<td>Canada</td>
<td>33,212,695</td>
<td>23,999,000</td>
<td>72.3 %</td>
<td>69.0 %</td>
<td>1.5 %</td>
</tr>
<tr>
<td>17</td>
<td>Iran</td>
<td>65,875,223</td>
<td>23,750,000</td>
<td>34.9 %</td>
<td>1,100.0 %</td>
<td>1.4 %</td>
</tr>
<tr>
<td>18</td>
<td>Vietnam</td>
<td>86,116,866</td>
<td>20,903,374</td>
<td>24.3 %</td>
<td>10,396.7 %</td>
<td>1.3 %</td>
</tr>
<tr>
<td>19</td>
<td>Poland</td>
<td>38,500,698</td>
<td>20,020,362</td>
<td>52.0 %</td>
<td>915.0 %</td>
<td>1.3 %</td>
</tr>
<tr>
<td>20</td>
<td>Argentina</td>
<td>40,681,981</td>
<td>20,000,000</td>
<td>49.4 %</td>
<td>700.0 %</td>
<td>1.3 %</td>
</tr>
</tbody>
</table>

TOP 20 Countries: 4,285,550,022
Rest of the World: 2,423,459,446
Total World - Users: 6,710,009,068

One View of Requirements

Identity & authentication
Authorization & policy
Resource discovery
Resource characterization
Resource allocation
(Co-)reservation, workflow
Distributed algorithms
Remote data access
High-speed data transfer
Performance guarantees
Monitoring

Adaptation
Intrusion detection
Resource management
Accounting & payment
Fault management
System evolution
Etc.
Etc.
...

Credits: Globus team
Another View: “Three Obstacles to Making Grid Computing Routine”

- **New approaches to problem solving**
  - Data Grids, distributed computing, peer-to-peer, collaboration grids, ...

- **Structuring and writing programs**
  - Abstractions, tools

- **Enabling resource sharing across distinct institutions**
  - Resource discovery, access, reservation, allocation; authentication, authorization, policy; communication; fault detection and notification; ...

Credits: Globus team

**Programming & Systems Problems**

- **The programming problem**
  - Facilitate development of sophisticated apps
  - Facilitate code sharing
  - Requires **programming environments**
    - APIs, SDKs, tools

- **The systems problem**
  - Facilitate coordinated use of diverse resources
  - Facilitate infrastructure sharing
    - e.g., certificate authorities, information services
  - Requires **systems**
    - protocols, services

Credits: Globus team
What do we need (user point of view)?

- **Single sign-on**
  - authentication to any resource of the Grid gives access to all resources

- **Single compute space**
  - one scheduler for all Grid resources

- **Single data space**
  - can address files and data from any Grid resources

- **Single development environment**
  - Grid tools and libraries that work on all grid resources

Credits: Globus team

The Systems Problem: Resource Sharing Mechanisms That Address security and policy concerns of resource owners and users

- Are flexible enough to deal with many resource types and sharing modalities
- Scale to large number of resources, many participants, many program components
- Operate efficiently when dealing with large amounts of data & computation

Credits: Globus team
Aspects of the Systems Problem

• Need for interoperability when different groups want to share resources
  • Diverse components, policies, mechanisms
  • E.g., standard notions of identity, means of communication, resource descriptions

• Need for shared infrastructure services to avoid repeated development, installation
  • E.g., one port/service/protocol for remote access to computing, not one per tool/appln
  • E.g., Certificate Authorities: expensive to run

• A common need for protocols & services

Hence, a Protocol-Oriented View of Grid Architecture, that Emphasizes ...

• Development of Grid protocols & services
  • Protocol-mediated access to remote resources
  • New services: e.g., resource brokering
  • “On the Grid” = speak Intergrid protocols
  • Mostly (extensions to) existing protocols

• Development of Grid APIs & SDKs
  • Interfaces to Grid protocols & services
  • Facilitate application development by supplying higher-level abstractions

• The (hugely successful) model is the Internet

Credits: Globus team
Layered Grid Architecture (By Analogy to Internet Architecture)

“Coordinating multiple resources”: ubiquitous infrastructure services, app-specific distributed services

“Sharing single resources”: negotiating access, controlling use

“Talking to things”: communication (Internet protocols) & security

“Controlling things locally”: Access to, & control of, resources

The Security Problem

- Resources being used may be extremely valuable & the problems being solved extremely sensitive

- Resources are often located in distinct administrative domains
  - Each resource may have own policies & procedures

- The set of resources used by a single computation may be large, dynamic, and/or unpredictable
  - Not just client/server

- It must be broadly available & applicable
  - Standard, well-tested, well-understood protocols
  - Integration with wide variety of tools
Resource Layer – Protocols & Services

- Grid Resource Allocation Management (GRAM)
  - Remote allocation, reservation, monitoring, control of compute resources

- GridFTP protocol (FTP extensions)
  - High-performance data access & transport

- Grid Resource Information Service (GRIS)
  - Access to structure & state information

- Others emerging: Catalog access, code repository access, accounting, etc.

- All built on connectivity layer: GSI & IP

Credits: Globus team

The Resource Management Problem

- Enabling secure, controlled remote access to computational resources and management of remote computation
  - Authentication and authorization
  - Resource discovery & characterization
  - Reservation and allocation
  - Computation monitoring and control

Credits: Globus team
Collective Layer – Protocols & Services

- Index servers aka metadirectory services
  - Custom views on dynamic resource collections assembled by a community
- Resource brokers (e.g., Condor Matchmaker)
  - Resource discovery and allocation
- Replica catalogs
- Replication services
- Co-reservation and co-allocation services
- Workflow management services
- Etc.

Credits: Globus team
Example: High-Throughput Computing System

- **App**: High Throughput Computing System
  - **Collective (App)**: Dynamic checkpoint, job management, failover, staging
  - **Collective (Generic)**: Brokering, certificate authorities
  - **Resource**: Access to data, access to computers, access to network performance data
  - **Connect**: Communication, service discovery (DNS), authentication, authorization, delegation
  - **Fabric**: Storage systems, schedulers

Example: Data Grid Architecture

- **App**: Discipline-Specific Data Grid Application
  - **Collective (App)**: Coherency control, replica selection, task management, virtual data catalog, virtual data code catalog, ...
  - **Collective (Generic)**: Replica catalog, replica management, co-allocation, certificate authorities, metadata catalogs
  - **Resource**: Access to data, access to computers, access to network performance data, ...
  - **Connect**: Communication, service discovery (DNS), authentication, authorization, delegation
  - **Fabric**: Storage systems, clusters, networks, network caches, ...

Credits: Globus team
Adaptivity

- Performance contracts for applications
- But a Grid system is highly dynamic
- Adaptivity is one of the keys of success as applications should continue to be executed efficiently even if
  - Resource performance evolve
  - Faults occur (network, processors),
  - Old software components can be thrown away,
  - New devices are added,
  - Hardware and software elements are upgraded (even during execution!)
- Strong need of research work for algorithms that are fault tolerant and which can adapt to varying execution conditions

Which approach for grid programming

- MPICH-G2: classical message passing
- CoG kits, GridProt: web portals
- Condor-G, Pegasus, Taverna, …: workflows management
- Legion, Proactive: object model for the grid
- NetSolve/Ninf/DIET: Network Enabled Solver systems (GridRPC)
- Components approach
- Web services

<table>
<thead>
<tr>
<th>Application Toolkits</th>
<th>Distributed Computing Toolkit</th>
<th>Data-Intensive Applications Toolkit</th>
<th>Collaborative Applications Toolkit</th>
<th>Remote Visualization Applications Toolkit</th>
<th>Problem Solving Applications Toolkit</th>
<th>Remote Instrumentation Applications Toolkit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grid Services (Middleware)</td>
<td>Resource-independent and application-independent services authentication, authorization, resource location, resource allocation, events, accounting, remote data access, information, policy, fault detection</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Grid Fabric (Resources)</td>
<td>Resource-specific implementations of basic services E.g., Transport protocols, name servers, differentiated services, CPU schedulers, public key infrastructure, site accounting, directory service, OS bypass</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Beware of the Cloud Hype!
Cloud Computing

What is Cloud Computing?

An emerging computing paradigm where data and services reside in massively scalable data centers and can be ubiquitously accessed from any connected devices over the internet.

4+ billion phones by 2010 [Source: Nokia]

Web 2.0-enabled PCs, TVs, etc.

Businesses, from startups to enterprises

Cloud Computing

- Two key concepts
  - Processing 1000x more data does not have to be 1000x harder
  - Cycles and bytes, not hardware are the new commodity

- Cloud computing is
  - Providing services on virtual machines allocated on top of a large physical machine pool
  - A method to address scalability and availability concerns for large scale applications
  - Democratized distributed computing

- Functionnalities
  - SaaS: Software as a Service
  - HaaS: Hardware as a Service
  - DaaS: Data as a Service
  - PaaS: Platform as a Service
  - IaaS: Infrastructure as a Service
IBM Cloud

A massively scalable and flexible computing platform of the future, built on IBM and open source software, for hosting Web 2.0 and SOA applications.

Cloud computing infrastructure to support the academic initiative can be delivered either as hosted or onsite solution.

Credit: IBM Corp.

Amazon Elastic Compute Cloud

A set of APIs and business models which give developer-level access to Amazon's infrastructure and content:

Data As A Service
- Amazon E-Commerce Service
- Amazon Historical Pricing

Search As A Service
- Alexa Web Information Service
- Alexa Top Sites
- Alexa Site Thumbnail
- Alexa Web Search Platform

Infrastructure As A Service
- Amazon Simple Queue Service
- Amazon Simple Storage Service
- Amazon Elastic Compute Cloud

People As A Service
- Amazon Mechanical Trunk

Credits: Jeff Barr, Amazon
Amazon Elastic Compute Cloud

- Provides on-demand processing power
- Virtual machine images

• Virtual Compute Cloud
• Elastic Capacity
• 1.7 GHz x86
• 1.7 GB RAM
• 160 GB Disk
• 250 MB/Second Network
• Network Security Model

Time or Traffic-based Scaling, Load testing, Simulation and Analysis, Rendering, Software as a Service Platform, Hosting

$.10 per server hour

$.10 - $.18 per GB data transfer

Credits: Jeff Barr, Amazon

Amazon EC2 Concepts

- Amazon Machine Image (AMI):
  • Bootable root disk
  • Pre-defined or user-built
  • Catalog of user-built AMIs
  • OS: Fedora, Centos, Gentoo, Debian, Ubuntu, Windows Server
  • App Stack: LAMP, mpiBLAST, Hadoop

- Instance:
  • Running copy of an AMI
  • Launch in less than 2 minutes
  • Start/stop programmatically

- Network Security Model:
  • Explicit access control
  • Security groups

- Inter-service bandwidth is free

Credits: Jeff Barr, Amazon
Amazon Simple Queue Service

- Efficient, reliable load distribution layer
- Pay by the message

Scalable Queuing
Elastic Capacity
Reliable, Simple, Secure

Inter-process messaging, data buffering, architecture component

$0.10 per 1000 messages

$0.10 - $0.18 per GB data transfer

Amazon SQS Concepts

- Queues:
  - Named message container
  - Persistent

- Messages:
  - Up to 256KB of data per message
  - Peek / Lock access model

- Scalable:
  - Unlimited number of queues per account
  - Unlimited number of messages per queue

Credits: Jeff Barr, Amazon
Amazon Simple Storage Service

- Virtually infinite storage capacity
- Provides permanence layer when EC2 nodes are not running

![Diagram showing Object-Based Storage features and costs]

Amazon S3 Concepts

- **Objects:**
  - Opaque data to be stored (1 byte … 5 Gigabytes)
  - Authentication and access controls

- **Buckets:**
  - Object container – any number of objects
  - 100 buckets per account / buckets are “owned”

- **Keys:**
  - Unique object identifier within bucket
  - Up to 1024 bytes long
  - Flat object storage model

- **Standards-Based Interfaces:**
  - REST and SOAP
  - URL-Addressability – every object has a URL

Credits: Jeff Barr, Amazon
Amazon Elastic Compute Cloud (Amazon EC2)

- Create an Amazon Machine Image (AMI) containing all your software, including your operating system and associated configuration settings, applications, libraries, etc. Think of this as zipping up the contents of your hard drive. We provide all the necessary tools to create and package your AMI.
- Upload this AMI to the Amazon S3 (Amazon Simple Storage Service) service. This gives us reliable, secure access to your AMI.
- Register your AMI with Amazon EC2. This allows us to verify that your AMI has been uploaded correctly and to allocate a unique identifier for it.
- Use this AMI ID and the Amazon EC2 web service APIs to run, monitor, and terminate as many instances of this AMI as required. Currently, we provide command line tools and Java libraries, and you may also directly access our SOAP or Query based APIs.
- You can also skip the first three steps and choose to launch an AMI that is provided by Amazon or shared by another user.
- While instances are running, you are billed for the computing and network resources that they consume.

http://www.amazon.com/ec2

Google App Engine

- Run your web applications on Google's infrastructure.
  - Google App Engine enables developers to build web applications on the same scalable systems that power our own applications.
- It’s easy to scale.
  - Google App Engine makes it easy to design scalable applications that grow from one to millions of users without infrastructure headaches.
- No assembly required.
  - Google App Engine exposes a fully-integrated development environment.
- It's free to get started.
  - Every Google App Engine application will have enough CPU, bandwidth, and storage to serve around 5 million monthly page views.

http://code.google.com/appengine/
Clouds vs Grids

• GRIDs
  • open standards (OGF, …)
  • publicly funded & operated (slow evolution)
  • no central management
  • interoperability important
  • geographically distributed; locally owned and managed
  • share (usually modest) local resources
  • scientific research, high-end users

• CLOUDs
  • no standardized interfaces
  • privately funded & operated (fast evolution)
  • managed by a single entity
  • no interoperability
  • geographically distributed; centrally owned and managed
  • make huge systems available
  • enterprise applications, information processing, data mining

Credits: G. Fox

Programming the Cloud: Google MapReduce

• Developed by Google in 2003
• Programming model: dataflow programming
  • Input & Output: each a set of key/value pairs
  • Programmer specifies two functions:
    ▪ map (in_key, in_value) -> list(out_key, intermediate_value)
      • Processes input key/value pair
      • Produces set of intermediate pairs
    ▪ reduce (out_key, list(intermediate_value)) -> list(out_value)
      • Combines all intermediate values for a particular key
      • Produces a set of merged output values (usually just one)
  • Inspired by similar primitives in LISP and other languages
  • Example uses:
    ▪ distributed grep     web link–graph reversal
    ▪ distributed sort    web access log stats
    ▪ term–vector per host inverted index construction
    ▪ document clustering  machine learning
    ▪ statistical machine translation ...

• Open–source version
  • Hadoop (java implementation of MapReduce + GFS + Bigtable)

Credits: J. Dean, S. Ghemawat, Google, Inc.
Google MapReduce: Task Granularity And Pipelining

- Fine granularity tasks: many more map tasks than machines
  - Minimizes time for fault recovery
  - Can pipeline shuffling with map execution
  - Better dynamic load balancing

- Often use 200,000 map/5000 reduce tasks w/ 2000 machines

---

Google File System

- Built for Google large scale data management
- One big problem: hardware failure!
- GFS concepts
  - Data spread evenly throughout cluster
  - Replicated 3x (locality aware replication, replica migrates)
  - Master machine detects failure and rebalances data on the fly
- Some features
  - Data automatically distributed to nodes at load time
  - Loss of nodes causes automatic data rebalance
Google Bigtable

- A Google database layer
- Data model: a big map
- <Row, Column, Timestamp> triple for key - lookup, insert, and delete API

<table>
<thead>
<tr>
<th>Project name</th>
<th>Table size (TB)</th>
<th>Compression ratio</th>
<th># Cells (billions)</th>
<th># Column Families</th>
<th># Locality Groups</th>
<th>% in memory</th>
<th>Latency-sensitive?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Crawl</td>
<td>800</td>
<td>11%</td>
<td>1000</td>
<td>16</td>
<td>8</td>
<td>0%</td>
<td>No</td>
</tr>
<tr>
<td>Crawl</td>
<td>50</td>
<td>33%</td>
<td>200</td>
<td>2</td>
<td>2</td>
<td>0%</td>
<td>No</td>
</tr>
<tr>
<td>Google Analytics</td>
<td>20</td>
<td>29%</td>
<td>10</td>
<td>1</td>
<td>1</td>
<td>0%</td>
<td>Yes</td>
</tr>
<tr>
<td>Google Analytics</td>
<td>200</td>
<td>14%</td>
<td>8</td>
<td>1</td>
<td>1</td>
<td>0%</td>
<td>Yes</td>
</tr>
<tr>
<td>Google Base</td>
<td>2</td>
<td>31%</td>
<td>10</td>
<td>29</td>
<td>3</td>
<td>15%</td>
<td>Yes</td>
</tr>
<tr>
<td>Google Earth</td>
<td>0.5</td>
<td>64%</td>
<td>8</td>
<td>7</td>
<td>2</td>
<td>33%</td>
<td>Yes</td>
</tr>
<tr>
<td>Google Earth</td>
<td>70</td>
<td>–</td>
<td>9</td>
<td>8</td>
<td>3</td>
<td>0%</td>
<td>No</td>
</tr>
<tr>
<td>Orkut</td>
<td>9</td>
<td>–</td>
<td>0.9</td>
<td>8</td>
<td>5</td>
<td>1%</td>
<td>Yes</td>
</tr>
<tr>
<td>Personalized Search</td>
<td>4</td>
<td>47%</td>
<td>6</td>
<td>93</td>
<td>11</td>
<td>5%</td>
<td>Yes</td>
</tr>
</tbody>
</table>

One application example: FlickrProducts
FlickrProducts architecture

**FlickrProducts architecture**

![FlickrProducts Diagram]

---

**GREEN-NET**

*Power Aware Software Frameworks for High Performance Data Transport and Computing in Large Scale Distributed Systems*

- Cost of IT: 1.5 billion machines / 5-10% of world electrical usage
- HPC race (Petaflops, exaflops machine) – Top500: > 1 M CPUs
- How to save energy without compromising Quality of Experiment? -> Studying and Designing Energy Efficient Large Scale Distributed Systems!

**GREEN-NET Project:**

- Studying and Energy usage analysis of large scale systems (Grid5000) and Thermal energy envelop studies
- Proposing Large scale On/Off models for energy savings and assuming network presence
- Validating models with real distributed systems traces
- Adapting a scheduling framework (OAR)
- Large scale trust delegation framework for tasks migration

**GREEN-NET Partners:** EPI RESO (Lyon), IRIT (Toulouse), EPI MESCAL (Grenoble), Virginia Tech (USA)

[http://www.ens-lyon.fr/LIP/RESO/Projects/GREEN-NET](http://www.ens-lyon.fr/LIP/RESO/Projects/GREEN-NET)
Conclusions

- Grid moved from research projects and PhD thesis to production and large scale deployment
- Many incarnation of the Grid (from volunteer Desktop computing systems to Clouds)
  - Large scale grids available for ambitious projects
  - Clouds to be rented for everyone

- Still “some” work to do
  - Standardization of APIs
  - Interoperability between platforms
  - How should we program the Grid?
  - Some nice little research issues related to models, heterogeneity management, adaptivity, fault tolerance, resource management, large scale data management, virtual machines management, energy consumption, hybrid architectures (GPU, multicore), …
Questions ?