Scalability and Heterogeneity

Colin Perkins

http://csperkins.org/teaching/2004-2005/gc5/

UNIVERSITY *of* GLASGOW

Lecture Outline

- Review of Traditional Distributed Systems
- How is Grid Computing Different?
  - Aspects of Heterogeneity
  - Aspects of Scalability
  - Implications for System Design
- Preparation for Tutorial 1

The aims of today:

- To understand why grid computing is difficult, raise a number of issues to consider throughout the module
- To give more examples of Grid computing systems

Review of Traditional Distributed Systems

"A distributed system is a collection of independent computers that appears to its users as a single coherent system."

[Tanenbaum & van Steen, 2002]

- The machines are autonomous, but the users think they're dealing with a single system
- Typically distributed systems are used to share resources within an organization:
  - Homogeneity eases management, fault tolerance, scheduling, authentication
  - E.g. a departmental fileserver, database of exam marks

What if we break the assumption of homogeneity?

…we move from distributed systems to grid computing!

What is Grid Computing?

Infrastructure for Internet-scale Distributed Systems

- A software system, implemented in terms of a middleware layer, that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities
- A system that allows sharing of remote resources as-if they were local, across geographical and organizational boundaries
- A large and widely distributed collection of independent systems that appears to its users as a single coherent system

There are many definitions, depending who you ask…

How is Grid Computing Different?

- A computational grid integrates disparate resources into a single virtual organization
  - Varying applications using the services of the Grid
  - Large and varied amounts of data to be processed
  - Varying classes of user, with different rights and responsibilities
  - Running on a range of networks, using varying hardware and software
  - Across different administrative and legal domains
- Implies a more **heterogeneous** environment and **greater scaling** than traditional distributed systems

- How does this affect system design?

Aspects of Heterogeneity

Heterogeneity comes from several factors:

- Users, applications and data
- Software and the hardware on which it runs
- Interconnection networks
- Organizations

Heterogeneous Users, Applications & Data

- Large scale grid computing started to service the needs of the e-science community

- The EGEE project ("Enabling Grids for e-Science in Europe")
  - A typical e-science grid development project
  - European Union funding: €30 million
  - Building a computational grid for physics, health and bio-sciences, earth sciences, astronomy, etc.
  - Share resources of 70 sites in 27 countries; aiming for thousands of active users, wide range of applications, lots of data
- The Grid2003 Production Grid for physics and astronomy [1]

Consider diversity of user locations and environment, job processing, data, trust and security models…

Heterogeneous Users, Applications & Data

- Many of the concepts of grid computing finding their way into commercial applications

- Google, Amazon or iTunes
  - Large database/commerce sites; significant financial value
  - Accessed directly via web-site, or embedded in an application
  - Worldwide user community; millions of users and transactions
- Business process automation
  - Automatic inventory processing, ordering management
  - E.g. airline reservation systems, stock trading and financial modelling

Consider diversity of user locations and environment, job processing, data, trust and security models…

Heterogeneous Hardware and Software

- EGEE aiming to allow users at 70 different sites to share data, run distributed computational jobs
- Google is reputed to manage a distributed system of ~100,000 hosts around the world; caching the entire web in memory
- In large-scale grids like these, you *cannot* standardize on a particular hardware or software environment
  - By the time you've synchronised the system software, some hardware will have failed, requirements will have changed
  - Client software will vary widely
  - Likely multiple versions of server and client software in use

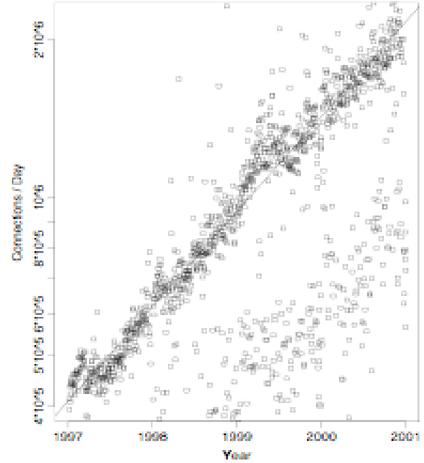
Design for compatibility, interoperability and cross-platform operation

Heterogeneous Networks

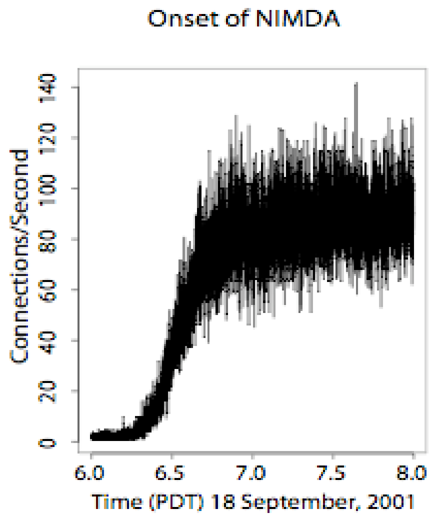
- Grids are widely distributed systems connected by the Internet
- What are the characteristics of the Internet?
  - Big and complex
  - Best effort service; no performance guarantees
  - Fragmented ownership
- Implication: the variation in the network will affect how we build a computational grid
  - Paper [2] discusses how network heterogeneity affects design and modelling of new protocols

Heterogeneous Networks

- Big, and getting bigger:
  - Size of the network has more than doubled *every* year since early 1980s
  - Approximately 100 million hosts at end of year 2000
  - What happens if 0.1% of hosts behave atypically?

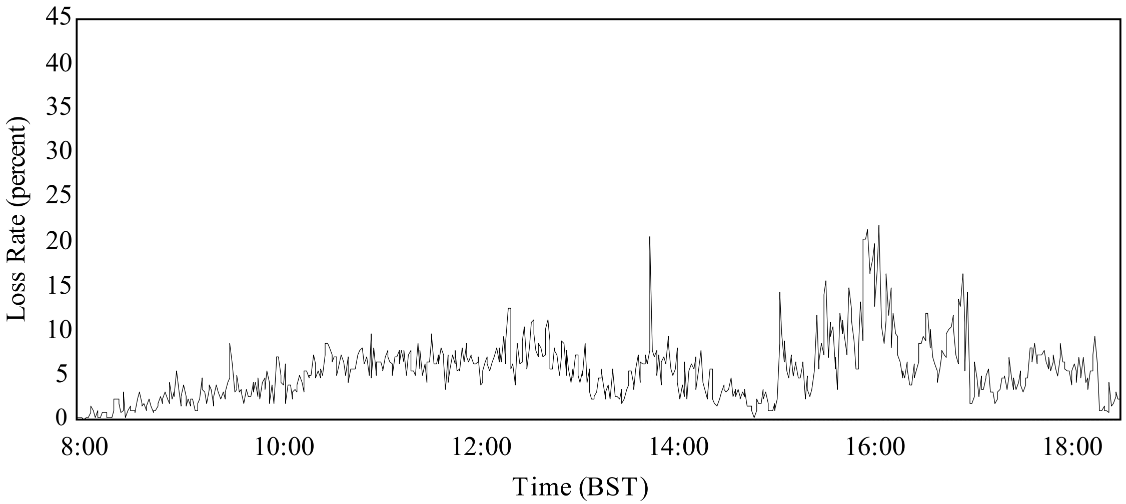


- Traffic patterns shift rapidly:
  - World-wide web: doubled every ~7 weeks for 2 years
  - Mbone: some sites reported >50% traffic in 1995 was multicast; now virtually none
  - Peer-to-peer: many reports of network congestion due to Napster, Kazaa, BitTorrent, etc.
  - Worms and malicious traffic: Nimda; from release to 100 probes-per-second in 30 minutes



Heterogeneous Networks

- At least 6 orders of magnitude variation in link speed:
  - 9.6 kbps GSM wireless → 10 Gbps optical fibre
  - Link capacity growing faster than Moore's law
- At least 4 orders of magnitude variation in latency:
  - Sub-millisecond LAN connections; hundreds of milliseconds worldwide
  - Varies with queuing delay, network congestion, path changes
  - A fundamental limiting factor for synchronous protocols (e.g. web services)
- Wide variation in packet loss rates:



Implications of Heterogeneous Networks

- Systems and protocols must be adaptive and scalable
- Decentralisation is essential, to handle load
- Global synchronisation is difficult, tending to impossible, due to latency

Implications for systems

Your system works in the lab today…
Will it still work in a few months,
when you have thousands of users?

Widely distributed or peer-to-peer

Asynchronous and weakly consistent

Location transparent

Loss tolerant, rate/latency adaptive

Organizational Heterogeneity

- Goal is to share resources across organizational boundaries, to form new *virtual organizations*
- How to authenticate users and resources?
  - Who do you trust to do the authentication?
  - Do you trust users to delegate authority?
    - To other users?
    - To software agents?
  - Do you trust the servers? The data?
- How to provide, control and limit access?
  - Full user accounts or a limited subset of functionality
  - Firewalls
  - Malicious users/applications
- Who sets the acceptable use policy?
  - Is it consistent worldwide? Can/should it be?

Significant implications on security infrastructure

How is Grid Computing Different?

- A computational grid integrates disparate resources into a single virtual organization
  - Varying applications using the services of the Grid
  - Large and varied amounts of data to be processed
  - Varying classes of user, with different rights and responsibilities
  - Running on a range of networks, using varying hardware and software
  - Across different administrative and legal domains
- Implies a more **heterogeneous** environment and **greater scaling** than traditional distributed systems

- How does this affect system design?

Aspects of Scalability

When building a grid, need to consider how it will scale in terms of:

- Data Storage and Distribution
- Software
- Scheduling
- Robustness and Fault Tolerance
- System Management

Scalability of Data Storage & Distribution

Storage is cheap:

- Apple Xserve RAID: 3.5Tbytes = £8,799
  5.25×17×18.4 inches

- Consider the storage available on a large distributed system…

Grid computing applications produce a lot of data:

- The ATLAS experiment at CERN will produce 1.3 petabytes/year of raw data (a stack of CDROMs 10 miles high…)
  - Simulation and analysis software routinely produces data files around 2 gigabytes in size
- Measurements on Grid2003 show 2 terabytes/day transferred to support experiments on a 27 site grid
  - Continuous 200 Mbits/second transfer rate

Scalability of Data Storage & Distribution

- How to manage, distribute this much data?
  - Do you move the data to the job? Or the job to the data?
  - Are you allowed to move the data?
    - Copyright, confidentiality, privacy, legal reasons
    - E.g. Grid computing for oil exploration: governments won't let geological data out of the country – remote access to terabytes of data in Africa from the US?
  - How to transfer large datasets?
    - Manually?
    - Automatically and transparently? How?
- How to index and search this much data?
- Need interoperable and standardised data formats
  - Long term archival and curation; efficient short term access
- How to do data fusion across heterogeneous databases/sources?
  - Transparent database queries across multiple systems, worldwide
- How to maintain data provenance?

Scalable Scheduling

- Job scheduling on single and parallel computers well understood
- Evolving towards job scheduling for clusters:
  - VAX/VMS clustering in the mid-1980s
  - Condor, OpenPBS, Sun Grid Engine, etc. more recently

- How to move to Internet-scale job scheduling?
  - Policy compliance
  - Load balancing
  - Co-scheduling
  - System monitoring and failure handling
  - Distributing jobs and data
  - Location transparency

An open research problem…

See also paper [3]

Naming, Addressing and Middleware

- How to write an application that runs over thousands of hosts, when you don't know which hosts it's using?

- Need a naming scheme and communication protocol that works independently of location
  - Can't use DNS or IP addresses directly; tied to organizational structure, network topology
- Peer-to-peer protocols solve some of these problems:
  - Distributed hash tables/content addressable networks and event notification systems built on them
  - e.g. Pastry and Scribe
- Lots of research; no standards yet

Paper [3] addresses some of these issues in more depth

Robustness and Fault Tolerance

- Systems fail; an internet-scale distributed system might never be completely operational
  - If a system is large enough, statistically likely something will have failed
    - Grid2003 reports job failure and restart rates of 30% in some cases… [1]
  - How big can a system be before failures become overwhelmingly likely?

- How to detect and recover from failures?
  - Routing around failures?
  - Recovering from failure while a job is running?
  - Avoiding cascading failures?
- Distributed systems and parallel computing has given us many useful algorithms
  - Complicated by the scale of computational grids, and cross organizational management issues

See paper [4]

System Configuration Management

- Independent of job scheduling and resource management, need to manage the configuration of the grid
  - Operating system updates + patches
  - New versions of application software
  - Detecting and fixing hardware and software failures

- How to manage thousands of hosts?
- How to manage a system that's never completely functional?

- Build applications that monitor the system, and reconfigure it as needed – leads to the idea of **autonomic computing** [5]

Summary

- The two biggest challenges to designing a computational grid are **heterogeneity** and **scalability**
- These distinguish grids from traditional distributed systems

- Have asked lots of questions… the reading list will raise more issues
- The rest of the module will try to answer some of these questions; others are open research topics…

- Next week: discussion of current standard architectures and protocols for grid computing

References

[1] I. Foster *et al.*, “The Grid2003 Production Grid: Principles and Practice”, Proceedings of the 13th IEEE Intl. Symp. on High Performance Distributed Computing, 2004.

[2] S. Floyd and V. Paxson, “Difficulties in Simulating the Internet”, IEEE/ACM Transactions on Networking, Vol. 9, No. 4, August 2001.

[3] J. A. Crowcroft, S. M. Hand, T. L. Harris, A. J. Herbert, M. A. Parker and I. A. Pratt, “FutureGRID: A Program for long-term research into GRID systems architecture”, Proceedings of the UK e-Science All Hands Meeting, Sept 2003.

[4] M. Amin, “Toward Self-Healing Infrastructure Systems”, IEEE Computer, August 2000.

[5] J. O. Kephart and D. M. Chess, “The Vision of Autonomic Computing”, IEEE Computer, January 2003.

Preparation for Tutorial 1

We will be discussing two papers on Monday:

- "Computational Grids"
- "The Anatomy of the Grid"

Between now and Monday:

- You should all read **both** papers
- Prepare a summary of each paper, explain "what is a grid?"
  - Work in groups to do this, discuss the papers in advance
  - Use the material from Research Techniques to help you prepare

On Monday, two people will be chosen at random for each paper:

- They will stand in front of the class and present the paper (5 minutes)
- Then, the rest of the class will then discuss the paper, to see if they agree with that view of grid computing