CMSC 818S
Computational Grids

Alan Sussman
September 5, 2002

Administrivia

• Class introductions
• Project
  – MPI project coming soon
    • it will count, but only for a small part of your grade
  – sign up for a Linux cluster account if you don’t have one already (redleader.umiacs.umd.edu)
• Next class will be led by Henrique Andrade, on Globus toolkit
  – chapter/paper available by tomorrow – see Readings web page
• We’ll talk about choosing topics to present next Thursday

What is a Grid?

• The hardware and software infrastructure that provides computing resources that are:
  – dependable – performance guarantees
  – consistent – standard services/interfaces
  – pervasive – available everywhere supported
  – inexpensive – economic argument
• “Coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations” – from The Anatomy of the Grid (Foster and Tuecke, Argonne/UI-Chicago)

Why Grids?

• To provide more computing power
  – from new technology – take advantage of new hardware, wherever it is located
  – to share available resources on demand (like time-sharing)
  – to increase utilization of underused resources (cycle stealing)
  – to share computational results – collaboratories
  – take advantage of new tools and techniques
    • e.g., network enabled solvers (Netsolve @UTK, Ninf in Japan)
    • teleimmersion – collaborative data exploration/analysis (Discover @ Rutgers)

Analogy to the Electric Power Grid

• Salient characteristics
  – heterogeneity
    • generators/outlets vs. machines/networks
  – consumers have different requirements
    • power consumption/service guarantees/$$$ vs. computational requirements/QOS/$$$
  – want to take advantage of economies of scale
  – politics
    • local control, but interfaced to uncontrolled environment – need standards

Applications

• Distributed supercomputing
  – to maximize available resources for large problems
    • within an organization, or across organizations
  – examples include distributed interactive simulation, simulating complex physical processes
  – challenges include co-scheduling, scalability of services to large numbers of nodes, tolerating latency, and obtaining high performance across heterogeneous systems
Applications (cont.)

- High-throughput computing
  - to schedule large numbers of loosely coupled or independent tasks onto available resources
  - examples include Condor (Wisconsin), LSF (Platform Computing)

Applications (cont.)

- On-demand computing
  - to provide access to non-local resources – computation, software, data, sensors, etc.
  - often driven by cost-performance rather than absolute performance
  - examples
    - software/computation - NEOS (Argonne), NetSolve (UTK), Ninf (Japan)
    - sensors – telemicroscopy
    - data – on-demand meteorological satellite data processing

Applications (cont.)

- Data intensive computing
  - Data analysis applications
  - Examples include:
    - GriPhyN – Grid Physics Network (griphyn.org)
    - Sloan Digital Sky Survey
    - Weather forecasting – using remote sensing data from satellites
    - Petroleum reservoir simulation data analysis
  - Challenges include scheduling/configuring storage and network resources

Applications (cont.)

- Collaborative computing
  - to enhance human-human interaction
  - to provide a virtual shared space
  - to share resources such as data archives or ongoing simulation results
  - examples include BoilerMaker (Argonne), CAVE5D (virtual reality hardware/software), NICE (Illinois)
  - challenges include realtime requirements for human interaction, and UI issues

Grid Community Examples

- Government – e.g., NSF Supercomputing Centers, DOE labs, etc. – national grid
- HMO – hospitals in a metro area – private grid
- Materials science collaboratory – university researchers at many sites – virtual grid
- Computational market economy – consumer/producer market – public grid
  - companies are doing this – e.g., Entropia, Parabon
  - free efforts too – e.g., SETI@Home, Mersenne primes

Using Grids

- Enabling Grid programming
  - to allow apps to adapt to changes in resource availability, and deal with resource heterogeneity in general
  - need standards for apps, programming models, tools, services – the Global Grid Forum (GGF, gridform.org)
  - basic Grid services talked about last
Using Grids (cont.)

• Tool developers
  – compilers, libraries, etc. to implement prog.
    models and services used by app developers
  – examples include Condor, Netsolve, MPICH-
    G2 (ISI/Argonne), CAVERN (UI-Chicago),
    DataCutter
  – built on top of basic Grid services – e.g.,
    Globus, Legion, TCP/IP
    • authentication, process management, data access,
      communication, resource location, fault detection,
      security, …

Using Grids (cont.)

• Application developers
  – Use tools (last slide) and basic Grid services
  – Want portability, efficiency, etc., just like in
    parallel (and sequential) world
  – Programming models include:
    • streams, shared memory w/multithreading, data
      parallelism, message passing, RPC, job
      scheduling(e.g., Condor), agents

Using Grids (cont.)

• End users
  – Want reliability, predictability, confidentiality, usability
    from grid applications
• System administrators
  – grids will often span multiple administrative domains
  – goal is to balance local policy requirements and needs
    of larger grid community
  – for scalability, must decentralize administration and
    automate cross-site issues (e.g., negotiating policy with
    other sites and users, accounting, etc.)

Grid architecture

• Scale up from end system to cluster to
  intranet to Internet
• Grid services
  – based on services that have been proved
    effective in previous environments, plus some
    new services
  – basic services include:
    • authentication, authorization, interprocess
      communication (IPC), resource
      acquisition/allocation, scheduling, accounting,
      payment, protection (from other users)

End Systems

• Computers, storage devices, sensors, etc.
• Small scale, homogeneous, integrated
• Basic services provided by OS
  – handles authentication, resource allocation, IPC, etc.
• Processor architecture, memory system, compiler highly
  integrated
  – enables high performance
• New features for integration into clusters, intranets,
  internets
  – OS support for clusters (distribute, not replicate, services)
  – better network integration – cheaper IPC
  – mobile code support, w/security – e.g., Java

Clusters

• Set of workstations/PCs connected by high-speed
  LAN – still homogeneous, under one
  administrative entity
• New issues
  – larger scale – hundreds to thousands of processors –
    complicates resource management
  – reduced integration – use commodity parts for low cost,
    so lower performance (e.g., IPC)
  – co-scheduling – so processes on different nodes don’t
    have to wait to communicate
• Mostly use same services as end user systems,
  with extensions, unless performance is critical
Intranets

- Grid belonging to a single organization
  - so has centralized admin control
  - but end systems and networks heterogeneous, individual systems separately administered, and no accurate global knowledge of system structure or current state
- End up with simpler, less tightly integrated set of services than for a cluster
  - data sharing (e.g., distributed FSs, DBs, Web services)
  - examples include DCE, DCOM, CORBA
  - interactions via RPC/RMI using standard protocols (layered on TCP/IP)
  - hard to get good performance

Intranets (cont.)

- Additional services over clusters include
  - resource discovery – what is available out there?
  - more security concerns due to reduced trust – e.g., use Kerberos
- Main issue is getting high performance
  - much research has been done here

Internets

- Span multiple organizations
  - large and heterogeneous like intranets
  - with no centralized control, geographic distribution of resources (network issues), and international issues (e.g., export controls)
- New approaches required for many basic services
  - security – cross domain authentication via public key cryptography, as in Globus GSI
  - coscheduling across multiple scheduling policies
- Hot topic now – the TeraGrid, an NSF funded project across several sites (NPACI, NCSA, Caltech, Argonne)
- Software projects include Globus, Legion