Cluster Computing

Metacomputing: Harnessing Informal Supercomputers

- General Introduction
- The Evolution of Metacomputing
- Metacomputing Design Objectives and Issues
- Metacomputing Projects
- Emerging Metacomputing Environments
- Summary and Conclusions

2

General Introduction

- From CASA project, one of several U.S. Gigabit testbeds around 1989
- Metacomputer
  - a dynamic environment that has some informal pool of nodes that can join or leave the environment whenever they desire
    - The nodes refer to independent machines
    - A parallel computer (IBM SP2) can be viewed as a "metacomputer in a box"
    - An SMP parallel computer (Tera MTA or SUN E10000) cannot as individual computational nodes in an SMP are not independent
  - By Catlett & Smarr
    - "the use of powerful computing resources transparently available to the user via a networked environment"
  - a networked virtual supercomputer
  - Steps to realize a metacomputer
    - the integration of individual SW & HW resources into a combined networked resources
    - the implementation of middleware to provide a transparent view of the resources available
    - the development & optimization of distributed applications to take advantage of the resources

3

Why Do We Need Metacomputing?

- Our computational needs are infinite, whereas our financial resources are finite
  - users will always want more & more powerful computers
  - try & utilize the potentially hundreds of thousands of computers that are interconnected in some unified way
  - need seamless access to remote resources

4

What is Metacomputer?

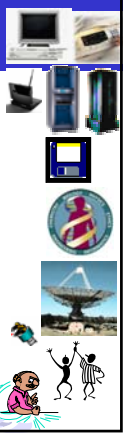
- Computational Grid
  - Equivalent to metacomputing environments
  - Describe a universal source of computing power
  - The means to provide pervasive access to advanced computational resources, databases, sensors, and people
  - Analogy to electricity grid
- Metacomputing encompasses
  - Seamless access to high performance resources
  - Parameter studies (embarrassingly parallel application)
  - The linkage of scientific instruments, analysis system, archival storage, and visualization (4-way metacomputing)
  - The general complex linkage of N distributed components

5

Towards Grid Computing … .

6

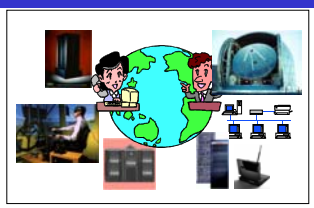
What is Grid ?



- An infrastructure that couples
  - Computers (PCs, workstations, clusters, traditional supercomputers, and even laptops, notebooks, mobile computers, PDA, and so on) …
  - Software (e.g., renting expensive special purpose applications on demand)
  - Databases (e.g., transparent access to human genome database)
  - Special Instruments (e.g., radio telescope--SETI@Home Searching for Life in galaxy, Austrophysics@Swinburne for pulsars)
  - People (may be even animals who knows ?)
- Across the Internet and presents them as an unified integrated (single) resource

7 http://www.csse.monash.edu.au/~rajkumar/ecogrid/

Conceptual view of the Grid



Leading to Portal (Super)Computing

8

Grid Application-Drivers

- Old and new applications getting enabled due to coupling of computers, databases, instruments, people, etc.
  - (distributed) Supercomputing
  - Collaborative engineering
  - High-throughput computing
    - large scale simulation & parameter studies
  - Remote software access / Renting Software
  - Data-intensive computing
  - On-demand computing

9

The Grid Impact

" The global computational grid is expected to drive the economy of the 21st century similar to the electric power grid that drove the economy of the 20th century"

10

The Parts of a Metacomputer

- A metacomputer
  - A virtual computer architecture
  - Its constituent components are individually not important
  - The key concept is how these components work together as a unified resources
  - Components
    - Processors and Memory
      - An array of processors with some dynamic memory associated
    - Networks and Communications Software
      - Physical connections and some communications software
    - Virtual Environment
      - Configure, manage, and maintain the metacomputing environment like an OS
    - Remote Data Access and Retrieval
      - Interacting between supercomputers across national or international networks (a major challenge for metacomputing environment)

11

The Evolution of Metacomputing

- Early examples
  - FAFNER
    - Running on any workstation with more than 4MB of memory
  - I-WAY
    - A means of unifying the resource of large supercomputing centers

12

FAFNER

- The RSA algorithm - combining prime numbers
  - factoring is computationally very expensive
- By Bellcore Labs. ,Syracus University and Co-Operating Systems in 1995
- A project of factoring via the Web
- Use parallel factoring algorithm (RSA130) which require no communications after the initial setup
- Use a numerical technique NFS (Number Field Sieve) factoring method using computational Web servers
- The consortium produced a Web interface to NFS
- Server-side CGI scripts (in Perl) invoked by contributors support services for the sieving step of the factorization
- Won award in TeraFlop challenge at SC95

13

FAFNER

- Factors for success
  - The NFS implementation allowed even single workstations with 4MB to perform useful work using small bounds and a small sieve
  - Support anonymous registration – users could contribute their hardware resources to the sieving effort without revealing their identity to anyone other than the local server administrator
  - A consortium of sites was recruited to run the CGI script package locally, forming a hierarchical network of RSA130 Web servers which reduced the potential administrator bottleneck and allowed sieving to proceed around the clock with minimal human intervention

14

I-WAY

- Information Wide Area Year (I-WAY)
  - an experimental high performance network linking many high performance computers & advanced visualization environment
  - conceived in early 1995 with the idea not to build a network, but to integrate existing high-bandwidth networks with telephone systems
  - connect 17 different U.S. sites by 10 networks
    - based on ATM technology
- I-POP (Point-of-Presence)
  - gateways to I-WAY
  - UNIX workstation
    - accessible via the Internet, operated within its site's firewall
    - had an ATM interface
  - I-POP provide uniform authentication, resource reservation, process creation, and communication functions across I-WAY resource

15

I-WAY

- I-Soft
  - a standard S/W environment of I-POP
  - help overcome issues such as heterogeneity, scalability, performance, and security
- Computational Resource Broker (CRB)
  - Resource scheduler
  - Consist user-to-CRB and CRB-to-local-scheduler protocols
  - A single central scheduler and multiple local scheduler daemons (one per I-POP)
    - Central scheduler maintain queues of jobs and tables representing the state of local machines, allocating jobs to machine and maintaining state information on the AFS file system
  - Authentication proxy, performing subsequent authentication to I-WAY resource on a user's behalf
    - Kerberos authentication and encryption

16

I-WAY

- AFS
  - Provide a shared file repository for software and scheduler information
  - A version of remote copy (ircp) to move data between machines
- Nexus
  - Low-level communication library
  - Support automatic configuration mechanism to choose appropriate configuration depending on the technology used
- Application driven
  - Supercomputer – Supercomputing
  - Remote Resource – Virtual Reality
  - Virtual Reality- Virtual Reality
  - Multisupercomputer – Multivirtual Reality
  - Video, Web, GII-Windows

17

Summary of FAFNER and I-WAY

- FAFNER
  - Forerunner of projects such as WebFlow
  - Work on any platform where a Web server could be run
  - Tailored to a particular factoring application that was in itself trivially parallel and was not dependent on a fast interconnect
  - Dependent on quite a lot of human intervention to distribute and collect sieving results
  - Lack of a number of features
    - Every client had to compile, link, and run a FAFNER daemon
    - Individual computational tasks were unable to communicate with one another
- I-WAY
  - I-soft was very influential on the approach used to design components employed in the Globus toolkit
  - Unify the resource at supercomputing sites
  - Cope with a range of diverse high performance applications that typically need a fast interconnect
  - Limited by the design of components that made up I-POP and I-Soft
  - A number of inappropriate features
    - I-POP is a single-points-of-failure

18

Metacomputer Design Objectives and Issues

- General Principles
  - Not interfere with the existing site administration or autonomy
  - Not compromise existing security of users or remote sites
  - Not need to replace existing OS, network protocols, or services
  - Allow remote sites to join or leave the environment whenever they choose
  - Not mandate the programming paradigms, languages, tools, or libraries that a user wants
  - Provide a reliable and fault tolerance infrastructure with no single point of failure
  - Provide support for heterogeneous components
  - Use standards, and existing technologies, and is able to interact with legacy applications
  - Provide appropriate synchronization and component program linkage

19

Metacomputer Design Objectives and Issues

- Underlying Hardware and Software Infrastructure
  - A metacomputing environment must be able to operate on top of the whole spectrum of current and emerging HW & SW technology
  - An ideal environment will provide access to the available resources in a seamless manner such that physical discontinuities such as difference between platforms, network protocols, and administrative boundaries become completely transparent

20

Metacomputer Design Objectives and Issues

- Middleware – The Metacomputing Environment
  - Administrative hierarchy
    - the way that each metacomputing environment divides itself up to cope with a potentially global extent
  - Communication services
    - needs to support protocols that are used for bulk-data transport, streaming data, group communications, and those used by distributed objects
  - Directory/registration services
    - provide the mechanism for registering and obtaining information about the metacomputer structure, resources, services, and status
  - Processes, threads, and concurrency control
    - share data and maintain consistency when multiple processes or threads have concurrent access to it

21

Metacomputer Design Objectives and Issues

- Middleware – The Metacomputing Environment
  - Time and clocks
    - time is an entity that we wish to measure accurately
    - algorithms have been developed that depends on clock synchronization
    - use for maintaining the consistency of distributed data or as a part of the Kerberos authentication protocol
  - Naming services
    - provide an uniform name space across the complex metacomputing environment
      - X.500, DNS naming schemes
  - Distributed file systems and caching
    - provide an uniform global namespace
    - support a range of file I/O protocols
    - require little or no program modification
    - provide means that enable performance optimization to be implemented

22

Metacomputer Design Objectives and Issues

- Middleware – The Metacomputing Environment
  - Security and authorization
    - confidentiality: prevent disclosure of data
    - integrity: prevent tampering with data
    - authorization: verify identity
    - accountability: knowing whom to blame
  - System status and fault tolerance
  - Resource management and scheduling
    - efficiently and effectively schedule the applications that need to utilize the available resource in the metacomputing environment

23

Metacomputer Design Objectives and Issues

- Middleware – The Metacomputing Environment
  - Programming tools and paradigms
    - include interface, APIs, and conversion tools so as to provide a rich development environment
    - support a range of programming paradigms
    - a suite of numerical and other commonly used libraries should be available
  - User and administrative GUI
    - intuitive and easy to use interface to the services and resources available
  - Availability
    - easily port on to a range of commonly used platforms, or use technologies that enable it to be platform neutral

24

Metacomputing Projects

- Globus (from Argonne National Laboratory)
  - provides a toolkit on a set of existing components to build metacomputing environments
- Legion (from the University of Virginia)
  - provides a high-level unified object model out of new and existing components to build a metasystem
- Webflow (from Syracuse University)
  - provides a Web-based metacomputing environment

25

Globus

- A computational grid
  - A hardware and software infrastructure to provide dependable, consistent, and pervasive access to high-end computational capabilities, despite the geographical distribution of both resources and users
- A layered architecture
  - high-level global services are built upon essential low-level core local services
- Globus Metacomputing Toolkit (GMT)
  - a central element of the Globus system
  - defines the basic services and capabilities required to construct a computational grid
  - consists of a set of components that implement basic services
  - provides a bag of services
  - only possible when the services are distinct and have well-defined interfaces (API)

26

Globus

- The GMT consists of the followings
  - Resources allocation and process management (GRAM)
  - Unicast and multicast communications services (Nexus)
  - Authentication and related security services (GSI)
  - Distributed access to structure and state information (MDS)
  - Monitoring of health and status of system components (HBM)
  - Remote access to data via sequential and parallel interfaces (GASS)
  - Construction, caching, and location of executables (GEM)

27

Globus

- Administrative hierarchy
  - no obvious administrative hierarchy
- Communication services
  - Nexus
    - Support for multimethod communication
    - Provide an application a single relatively low-level communication API to support a wide range of high-level communication protocol characteristics
- Directory/registration services
  - Globus Metacomputing Directory Service (MDS)
    - Provide static and dynamic information about the status of Globus system components
    - Use Lightweight Directory Access Protocol (LDAP) server to store metacomputing-specific objects
    - House information pertaining to the potential computing resource, their specifications, and their current availability

28

Globus

- Processes, threads, and concurrency control
  - Work at process level
  - Nexus API can be used to construct communication primitive between threads
  - No concurrency control
- Time and clocks
  - No particular time service
- Naming services
  - Use of LDAP, DNS and X.500
- Security and authorization
  - Generic Security Service API (GSI) using an implementation of Secure Sockets Layer and X509 certificate as authentication system
  - Use RSA encryption algorithm and associated public and private keys

29

Globus

- Distributed file systems and caching
  - Global Access to Secondary Storage (GASS)
    - Define a global name space via URLs
    - Provide basic access to remote files via standard I/O interface
    - File cache is used to address bandwidth management issues associated with repeated access to remote files
    - A simple locking protocol for local concurrency control, not implement a wide-area cache coherency mechanism
  - Remote I/O (RIO)
    - A distributed implementation of MPI-IO, parallel I/O API
  - Globus Executable Management (GEM)
    - Enable loading and executing a remote file through GRAM using GASS caching calls

30

Globus

- System status and fault tolerance
  - Detection of a fault is a necessary prerequisite to fault recovery and fault tolerance
  - Heartbeat monitor (HBM) as fault detection service
  - Nexus communication library support for fault detection
- Resource management and scheduling
  - Globus Resource Allocation Manager (GRAM)
    - Allow jobs to run remotely and provide an API for submitting, monitoring, and terminating jobs
    - Provide local component for resource management
    - GRAM is responsible for
      - Parsing and processing the Resource Specific Language (RSL) specifications that outline job requests
      - Enabling remote monitoring and managing of jobs already created
      - Updating MDS with information regarding the availability of the resource it manages

31

Globus

- Programming tools and paradigms
  - Support MPI, Java, Compositional C++, Simple RPC, Perl
- User and administrative GUI
  - Use Web and command line interface
- Availability
  - Available on most version of UNIX

32

Legion

- An object-based metasystem
- Organized by classes and metaclasses
  - Every thing is an object
  - Classes manage their instances
  - Users can define their own classes
  - Core objects
- Core objects
  - Classes and Metaclasses, Host objects, Vault objects, Binding Agents
  - Implementation Objects and Caches, Context objects and Context spaces

33

Legion

- Interface Definition Language (IDL)
  - The set of methods of an object describes its interface
- Two state of Legion object
  - active: run as a process that is accept function invocations.
  - inert: the object which resides on some stable storage (OPR)
- Three-tiered naming system
  - Users refer to objects using human-readable strings, called context names
  - Context objects map context names to LOIDs which are location-independent identifier that include an RSA public key
  - A LOID is mapped to an LOA

34

WebFlow

- A computational extension of the Web model
  - can act as a framework for the wide-area distributed computing and metacomputing
- The main goal
  - build a seamless framework for publishing and reusing computational modules on the Web
- Three-tier Java-based architecture
- The high performance backend tier is implemented using the Globus toolkit .
  - MDS, GRAM, GASS
- A high level, visual user interface and job broker for Globus

35

WebFlow

- The Management Infrastructure - by three servelets
  - Session Manager
  - Module Manager
  - Connection Manager
- use URL addresses
- offer dynamic information about their services and current state
- communicate with each other via sockets

36

Metacomputing Functionality Matrix

Design Objective	Globus	Legion	Webflow
Admin. Hierarchy	Peer	Peer	Peer
Comm. Service	Nexus – Low-Level	MMPS – Socket-based	Hierarchical – Sockets+MPI
Dir/Reg. Services	MDS - LDAP	via Binding agents	MDS - LDAP
Process	Process-based	Object/process-based	Process-based
Clock	Not specified	Not specified	Not specified
Naming Services	LDAP +DNS/x.500	Context Manger+DNS	LDAP +DNS
Filesystems & Caching	GASS+ROMIO	Custom Legion filesystem	GASS
Security	GSI (RSA+X.509 certs)	Object-based with RSA	SSL
Fault Tolerance	Heart-beat monitor	Not available yet	None
Resource Management	GRAM+RSL+Local	Host object+Local	GRAM-based
Prog. Paradigms	Many and varied	MPL, BFS+wrappers	MPI
User Interface	GUI+command-line	GUI+command-line	Applet-based GUI
Availability	Most UNIX	Most UNIX	Most UNIX and NT

37

Emerging Metacomputing Environments

- Java and the Web
  - as the communications infrastructure
  - Java has revolutionized the shape and characteristics of the software environments for heterogeneous distributed systems
  - So the developers no longer have to focus on aspects such as portability and heterogeneity

38

Metacomputing Trends

- Java
- CORBA
- From a relatively slow start, the development of metacomputers is accelerating fast with the advent of these new and emerging technologies
- The framework of a metacomputing environment must be adaptable, malleable, and extensible, whatever technology and fashions become influential

39

The Impact of Metacomputing

- Metacomputing is an infrastructure that can bond and unify globally remote and diverse resources
- At some stage in the future, our computing needs will be satisfied in same pervasive and ubiquitous manner that we use the electricity power grid

40