High Performance Cluster Computing
Architectures and Systems

Book Editor: Rajkumar Buyya
Slides Prepared by: Hai Jin

Internet and Cluster Computing Center

Cluster Computing at a Glance
Chapter 1: by M. Baker and R. Buyya

- Introduction
- Scalable Parallel Computer Architecture
- Towards Low Cost Parallel Computing and Motivations
- Windows of Opportunity
- A Cluster Computer and its Architecture
- Clusters Classifications
- Commodity Components for Clusters
- Network Service/Communications SW
- Cluster Middleware and Single System Image
- Resource Management and Scheduling (RMS)
- Programming Environments and Tools
- Cluster Applications
- Representative Cluster Systems
- Cluster of SMPs (CLUMPS)
- Summary and Conclusions

Introduction

- Need more computing power
  - Improve the operating speed of processors & other components
    - constrained by the speed of light, thermodynamic laws, & the high financial costs for processor fabrication
  - Connect multiple processors together & coordinate their computational efforts
    - parallel computers
    - allow the sharing of a computational task among multiple processors

How to Run Applications Faster?

- There are 3 ways to improve performance:
  - Work Harder
  - Work Smarter
  - Get Help

  Computer Analogy
  - Using faster hardware
  - Optimized algorithms and techniques used to solve computational tasks
  - Multiple computers to solve a particular task

Era of Computing

- Rapid technical advances
  - the recent advances in VLSI technology
  - software technology
    - OS, PL, development methodologies, & tools
    - grand challenge applications have become the main driving force
  - Parallel computing
    - one of the best ways to overcome the speed bottleneck of a single processor
    - good price/performance ratio of a small cluster-based parallel computer

Two Eras of Computing

- Era of Computing
- Two Eras of Computing
Scalable Parallel Computer Architectures

- **Taxonomy**
  - based on how processors, memory & interconnect are laid out
- **Massively Parallel Processors (MPP)**
- **Symmetric Multiprocessors (SMP)**
- **Cache-Coherent Nonuniform Memory Access (CC-NUMA)**
- **Distributed Systems**
- **Clusters**

MPP
- A large parallel processing system with a shared-nothing architecture
- Consist of several hundred nodes with a high-speed interconnection network/switch
- Each node consists of a main memory & one or more processors
- Runs a separate copy of the OS

SMP
- 2-64 processors today
- Shared-everything architecture
- All processors share all the global resources available
- Single copy of the OS runs on these systems

CC-NUMA
- a scalable multiprocessor system having a cache-coherent nonuniform memory access architecture
- every processor has a global view of all of the memory

Distributed systems
- considered conventional networks of independent computers
- have multiple system images as each node runs its own OS
- the individual machines could be combinations of MPPs, SMPs, clusters, & individual computers

Clusters
- a collection of workstations of PCs that are interconnected by a high-speed network
- work as an integrated collection of resources
- have a single system image spanning all its nodes

Towards Low Cost Parallel Computing

- Parallel processing
  - linking together 2 or more computers to jointly solve some computational problem
  - since the early 1990s, an increasing trend to move away from expensive and specialized proprietary parallel supercomputers towards networks of workstations
  - the rapid improvement in the availability of commodity high performance components for workstations and networks

- Low-cost commodity supercomputing
  - from specialized traditional supercomputing platforms to cheaper, general purpose systems consisting of loosely coupled components built up from single or multiprocessor PCs or workstations
  - need to standardization of many of the tools and utilities used by parallel applications (ex) MPI, HPF

Key Characteristics of Scalable Parallel Computers

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>MPP</th>
<th>SMP</th>
<th>Cluster</th>
<th>Distributed</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Number of Nodes</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Node Commodity</td>
<td>High</td>
<td>Mid</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td>Node Memory</td>
<td>High</td>
<td>Low</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>Inter-node Communication</td>
<td>Message passing</td>
<td>Shared memory</td>
<td>Model passing</td>
<td>Multiple queues</td>
</tr>
<tr>
<td>Job Scheduling</td>
<td>Dynamic</td>
<td>Static</td>
<td>Dynamic</td>
<td>Static</td>
</tr>
<tr>
<td>Job Support</td>
<td>Local</td>
<td>Local</td>
<td>Local</td>
<td>Local</td>
</tr>
<tr>
<td>Node OS Types</td>
<td>Single</td>
<td>Multiple</td>
<td>Single</td>
<td>Multiple</td>
</tr>
<tr>
<td>Inter-node Security</td>
<td>Necessary</td>
<td>Necessary</td>
<td>Necessary</td>
<td>Necessary</td>
</tr>
<tr>
<td>Ownership</td>
<td>Single</td>
<td>Multi</td>
<td>Single</td>
<td>Multi</td>
</tr>
</tbody>
</table>

Motivations of using NOW over Specialized Parallel Computers

- Individual workstations are becoming increasing powerful
- Communication bandwidth between workstations is increasing and latency is decreasing
- Workstation clusters are easier to integrate into existing networks
- Typical low user utilization of personal workstations
- Development tools for workstations are more mature
- Workstation clusters are a cheap and readily available
- Clusters can be easily grown
### Trend
- Workstations with UNIX for science & industry vs PC-based machines for administrative work & work processing
- A rapid convergence in processor performance and kernel-level functionality of UNIX workstations and PC-based machines

### Windows of Opportunities
- **Parallel Processing**
  - Use multiple processors to build MPP/DSM-like systems for parallel computing
- **Network RAM**
  - Use memory associated with each workstation as aggregate DRAM cache
- **Software RAID**
  - Redundant array of inexpensive disks
  - Use the arrays of workstation disks to provide cheap, highly available, & scalable file storage
  - Possible to provide parallel I/O support to applications
- **Multipath Communication**
  - Use multiple networks for parallel data transfer between nodes

### Cluster Computer and its Architecture
- A cluster is a type of parallel or distributed processing system, which consists of a collection of interconnected stand-alone computers cooperatively working together as a single, integrated computing resource
- A node
  - a single or multiprocessor system with memory, I/O facilities, & OS
  - generally 2 or more computers (nodes) connected together
  - in a single cabinet, or physically separated & connected via a LAN
  - appear as a single system to users and applications
  - provide a cost-effective way to gain features and benefits

### Cluster Computer Architecture

### Prominent Components of Cluster Computers (I)
- **Multiple High Performance Computers**
  - PCs
  - Workstations
  - SMPs (CLUMPS)
  - Distributed HPC Systems leading to Metacomputing
### Prominent Components of Cluster Computers (II)

- **State of the art Operating Systems**
  - Linux (Beowulf)
  - Microsoft NT (Illinois HPVM)
  - SUN Solaris (Berkeley NOW)
  - IBM AIX (IBM SP2)
  - HP UX (Illinois - PANDA)
  - Mach (Microkernel based OS) (CMU)
  - Cluster Operating Systems (Solaris MC, SCO Unixware, MOSIX (academic project))
  - OS gluing layers (Berkeley Glunix)

### Prominent Components of Cluster Computers (III)

- **High Performance Networks/Switches**
  - Ethernet (10Mbps), Fast Ethernet (100Mbps)
  - InfiniteBand (1-8 Gbps)
  - Gigabit Ethernet (16bps)
  - SCI (Dolphin - MPI- 12micro-sec latency)
  - ATM
  - Myrinet (1.2Gbps)
  - Digital Memory Channel
  - FDDI

### Prominent Components of Cluster Computers (IV)

- **Network Interface Card**
  - Myrinet has NIC
  - InfiniteBand (HBA)
  - User-level access support

### Prominent Components of Cluster Computers (V)

- **Fast Communication Protocols and Services**
  - Active Messages (Berkeley)
  - Fast Messages (Illinois)
  - U-net (Cornell)
  - XTP (Virginia)

### Prominent Components of Cluster Computers (VI)

- **Cluster Middleware**
  - Single System Image (SSI)
  - System Availability (SA) Infrastructure

- **Hardware**
  - DEC Memory Channel, DSM (Alewife, DASH), SMP Techniques

- **Operating System Kernel/Gluing Layers**
  - Solaris MC, Unixware, GLUnix

- **Applications and Subsystems**
  - Applications (system management and electronic forms)
  - Runtime systems (software DSM, PFS etc.)
  - Resource management and scheduling software (RMS)
  - CODINE, LSF, PBS, NQS, etc.

### Prominent Components of Cluster Computers (VII)

- **Parallel Programming Environments and Tools**
  - Threads (PCs, SMPs, NOW.)
  - POSIX Threads
  - Java Threads
  - MPI
  - Linux, NT, on many Supercomputers
  - PVM
  - Software DSMs (Shmem)
  - Compilers
    - C/C++/Java
    - Parallel programming with C++ (MIT Press book)
  - RAD (rapid application development tools)
  - GUI based tools for PP modeling
  - Debuggers
  - Performance Analysis Tools
  - Visualization Tools
Prominent Components of Cluster Computers (VIII)

- Applications
  - Sequential
  - Parallel / Distributed (Cluster-aware app.)
    - Grand Challenging applications
      - Weather Forecasting
      - Quantum Chemistry
      - Molecular Biology Modeling
      - Engineering Analysis (CAD/CAM)
    - …………..
  - PDBs, web servers, data-mining

Key Operational Benefits of Clustering

- High Performance
- Expandability and Scalability
- High Throughput
- High Availability

Clusters Classification (I)

- Application Target
  - High Performance (HP) Clusters
    - Grand Challenging Applications
  - High Availability (HA) Clusters
    - Mission Critical applications

Clusters Classification (II)

- Node Ownership
  - Dedicated Clusters
  - Non-dedicated clusters
    - Adaptive parallel computing
    - Communal multiprocessing

Clusters Classification (III)

- Node Hardware
  - Clusters of PCs (CoPs)
  - Piles of PCs (PoPs)
  - Clusters of Workstations (COWs)
  - Clusters of SMPs (CLUMPs)

Clusters Classification (IV)

- Node Operating System
  - Linux Clusters (e.g., Beowulf)
  - Solaris Clusters (e.g., Berkeley NOW)
  - NT Clusters (e.g., HPVM)
  - AIX Clusters (e.g., IBM SP2)
  - SCO/Compaq Clusters (Unixware)
  - Digital VMS Clusters
  - HP-UX clusters
  - Microsoft Wolfpack clusters
Clusters Classification (V)

- **Node Configuration**
  - **Homogeneous Clusters**
    - All nodes will have similar architectures and run the same OSs
  - **Heterogeneous Clusters**
    - All nodes will have different architectures and run different OSs

Clusters Classification (VI)

- **Levels of Clustering**
  - Group Clusters (#nodes: 2-99)
    - Nodes are connected by SAN like Myrinet
  - Departmental Clusters (#nodes: 10s to 100s)
  - Organizational Clusters (#nodes: many 100s)
  - National Metacomputers (WAN/Internet-based)
    - International Metacomputers (Internet-based, #nodes: 1000s to many millions)
      - Metacomputing
      - Web-based Computing
      - Agent-Based Computing
        - Java plays a major role in web and agent based computing

Commodity Components for Clusters (I)

- **Processors**
  - Intel x86 Processors
    - Pentium Pro and Pentium Xeon
  - AMD x86, Cyrix x86, etc.
  - Digital Alpha
    - Alpha 21364 processor integrates processing, memory controller, network interface into a single chip
  - IBM PowerPC
  - Sun SPARC
  - SGI MIPS
  - HP PA
  - Berkeley Intelligent RAM (IRAM) integrates processor and DRAM onto a single chip

Commodity Components for Clusters (II)

- **Memory and Cache**
  - Standard Industry Memory Module (SIMM)
  - Extended Data Out (EDO)
    - Allow next access to begin while the previous data is still being read
  - Fast page
    - Allow multiple adjacent accesses to be made more efficiently
  - Access to DRAM is extremely slow compared to the speed of the processor
    - the very fast memory used for cache is expensive & cache control circuitry becomes more complex as the size of the cache grows
  - Within Pentium-based machines, uncommon to have a 64-bit wide memory bus as well as a chip set that support 2Mbytes of external cache

Commodity Components for Clusters (III)

- **Disk and I/O**
  - Overall improvement in disk access time has been less than 10% per year
  - Amdahl’s law
    - Speed-up obtained by from faster processors is limited by the slowest system component
  - Parallel I/O
    - Carry out I/O operations in parallel, supported by parallel file system based on hardware or software RAID

Commodity Components for Clusters (IV)

- **System Bus**
  - ISA bus (AT bus)
    - Clocked at 5MHz and 8 bits wide
    - Clocked at 13MHz and 16 bits wide
  - VESA bus
    - 32 bits bus matched system’s clock speed
  - PCI bus
    - 133Mbytes/s transfer rate
    - Adopted both in Pentium-based PC and non-Intel platform (e.g., Digital Alpha Server)
Commodity Components for Clusters (V)

- **Cluster Interconnects**
  - Communicate over high-speed networks using a standard networking protocol such as TCP/IP or a low-level protocol such as AM
  - **Standard Ethernet**
    - 10 Mbps
    - cheap, easy way to provide file and printer sharing
    - bandwidth & latency are not balanced with the computational power
  - **Ethernet, Fast Ethernet, and Gigabit Ethernet**
    - Fast Ethernet – 100 Mbps
    - Gigabit Ethernet
    - preserve Ethernet’s simplicity
    - deliver a very high bandwidth to aggregate multiple Fast Ethernet segments

Commodity Components for Clusters (VI)

- **Cluster Interconnects**
  - Asynchronous Transfer Mode (ATM)
    - Switched virtual-circuit technology
    - Cell (small fixed-size data packet)
  - use optical fiber - expensive upgrade
  - **Ethernet, Fast Ethernet, and Gigabit Ethernet**
    - **Fast Ethernet**
      - 100 Mbps
    - **Gigabit Ethernet**
      - preserve Ethernet’s simplicity
      - deliver a very high bandwidth to aggregate multiple Fast Ethernet segments
  - Scalable Coherent Interfaces (SCI)
    - IEEE 1596-1992 standard aimed at providing a low-latency distributed shared memory across a cluster
    - Point-to-point architecture with directory-based cache coherence
    - reduce the delay interprocessor communication
    - eliminates the need for runtime layers of software protocol-paradigm translation
    - less than 12 usec zero message-length latency on Sun SPARC
    - Designed to support distributed multiprocessor
    - with high bandwidth and low latency
    - SCI cards for SPARC’s SBus and PCI-based SCI cards from Dolphin
    - Scalability constrained by the current generation of switches & relatively expensive components

Commodity Components for Clusters (VII)

- **Cluster Interconnects**
  - Myrinet
    - 1.28 Gbps full duplex interconnection network
    - Use low latency cut-through routing switches, which is able to offer fault tolerance by automatic mapping of the network configuration
    - Support both Linux & NT
    - Advantages
      - Very low latency (5 usec, one-way point-to-point)
      - Very high throughput
      - Programmable on-board processor for greater flexibility
    - Disadvantages
      - Expensive: $1500 per host
      - Complicated scaling: switches with more than 16 ports are unavailable

Commodity Components for Clusters (VIII)

- **Operating Systems**
  - 2 fundamental services for users
    - make the computer hardware easier to use
    - create a virtual machine that differs markedly from the real machine
    - share hardware resources among users
    - Processor - multitasking
    - The new concept in OS services
    - support multiple threads of control in a process itself
      - parallelism within a process
      - multitasking
    - POSIX thread interface is a standard programming environment
  - Trend
    - Modularity – MS Windows, IBM OS/2
    - Microkernel – provide only essential OS services
    - high level abstraction of OS portability

Commodity Components for Clusters (IX)

- **Operating Systems**
  - Linux
    - UNIX-like OS
    - Runs on cheap x86 platform, yet offers the power and flexibility of UNIX
    - Readily available on the Internet and can be downloaded without cost
    - Easy to fix bugs and improve system performance
    - Users can develop or fine-tune hardware drivers which can easily be made available to other users
    - Features such as preemptive multitasking, demand-page virtual memory, multiuser, multiprocessor support

Commodity Components for Clusters (X)

- **Operating Systems**
  - Solaris
    - UNIX-based multitasking and multiuser OS
    - support Intel x86 & SPARC-based platforms
    - Real-time scheduling feature critical for multimedia applications
    - Support two kinds of threads
      - Light Weight Processes (LWP)
      - User-level thread
    - Support both BSD and several non-BSD file system
      - CacheFS
      - AutoClient
      - TmpFS: uses main memory to contain a file system
      - Proc file system
      - Volume file system
    - Support distributed computing & is able to store & retrieve distributed information
    - OpenWindows allows application to be run on remote systems
Commodity Components for Clusters (XI)

- Operating Systems
  - Microsoft Windows NT (New Technology)
    - Preemptive, multitasking, multiuser, 32-bits OS
    - Object-based security model and special file system (NTFS) that allows permissions to be set on a file and directory basis
    - Support multiple CPUs and provide multitasking using symmetrical multiprocessing
    - Support different CPUs and multiprocessor machines with threads
    - Have the network protocols & services integrated with the base OS
      - several built-in networking protocols (IPX/SPX, TCP/IP, NetBEUI), & APIs (NetBIOS, DCE RPC, Window Sockets (Winsock))

Windows NT 4.0 Architecture

- Network Services/ Communication SW
  - Communication infrastructure support protocol for
    - Bulk-data transport
    - Streaming data
    - Group communications
  - Communication service provide cluster with important QoS parameters
    - Latency
    - Bandwidth
    - Reliability
    - Fault-tolerance
    - Jitter control
  - Network service are designed as hierarchical stack of protocols with relatively low-level communication API, provide means to implement wide range of communication methodologies
    - RPC
    - DSM
    - Stream-based and message passing interface (e.g., MPI, PVM)

Cluster Middleware & SSI

- SSI
  - Supported by a middleware layer that resides between the OS and user-level environment
  - Middleware consists of essentially 2 sublayers of SW infrastructure
    - SSI infrastructure
      - Glue together OSs on all nodes to offer unified access to system resources
    - System availability infrastructure
      - Enable cluster services such as checkpointing, automatic failover, recovery from failure, & fault-tolerant support among all nodes of the cluster

What is Single System Image (SSI)?

- A single system image is the illusion, created by software or hardware, that presents a collection of resources as one, more powerful resource.
- SSI makes the cluster appear like a single machine to the user, to applications, and to the network.
- A cluster without a SSI is not a cluster

Single System Image Boundaries

- Every SSI has a boundary
- SSI support can exist at different levels within a system, one able to be build on another
SSI Boundaries -- an applications
SSI boundary

SSI Levels/Layers

SSI at Hardware Layer

SSI at Operating System Kernel
(Underware) or Gluing Layer

SSI at Application and Subsystem
Layer (Middleware)

Single System Image Benefits

- Provide a simple, straightforward view of all system resources and activities, from any node of the cluster
- Free the end user from having to know where an application will run
- Free the operator from having to know where a resource is located
- Let the user work with familiar interface and commands and allows the administrators to manage the entire clusters as a single entity
- Reduce the risk of operator errors, with the result that end users see improved reliability and higher availability of the system
Single System Image Benefits (Cont’d)

- Allowing centralize/decentralize system management and control to avoid the need of skilled administrators from system administration
- Present multiple, cooperating components of an application to the administrator as a single application
- Greatly simplify system management
- Provide location-independent message communication
- Help track the locations of all resource so that there is no longer any need for system operators to be concerned with their physical location
- Provide transparent process migration and load balancing across nodes.
- Improved system response time and performance

Middleware Design Goals

- Complete Transparency in Resource Management
  - Allow user to use a cluster easily without the knowledge of the underlying system architecture
  - The user is provided with the view of a globalized file system, processes, and network
- Scalable Performance
  - Can easily be expanded, their performance should scale as well
  - To extract the max performance, the SSI service must support load balancing & parallelism by distributing workload evenly among nodes
- Enhanced Availability
  - Middleware service must be highly available at all times
  - At any time, a point of failure should be recoverable without affecting a user’s application
  - Employ checkpointing & fault tolerant technologies
  - Handle consistency of data when replicated

SSI Support Services

- Single Entry Point
  - telnet cluster.myinstitute.edu
  - telnet node1.cluster.myinstitute.edu
- Single File Hierarchy: xFS, AFS, Solaris MC Proxy
- Single Management and Control Point: Management from single GUI
- Single Virtual Networking
- Single Memory Space - Network RAM / DSM
- Single Job Management: GLUnix, Codine, LSF
- Single User Interface: Like workstation/PC windowing environment (CDE in Solaris/NT), may it can use Web technology

Availability Support Functions

- Single I/O Space (SIOS)
  - any node can access any peripheral or disk devices without the knowledge of physical location.
- Single Process Space (SPS)
  - Any process on any node create process with cluster wide process wide and they communicate through signal, pipes, etc, as if they are one a single node.
- Checkpointing and Process Migration.
  - Saves the process state and intermediate results in memory to disk to support rollback recovery when node fails
  - PM for dynamic load balancing among the cluster nodes

Resource Management and Scheduling (RMS)

- RMS is the act of distributing applications among computers to maximize their throughput
- Enable the effective and efficient utilization of the resources available
- Software components
  - Resource manager
    - Locating and allocating computational resource, authentication, process creation and migration
  - Resource scheduler
  - Queuing applications, resource location and assignment
- Reasons using RMS
  - Provide an increased, and reliable, throughput of user applications on the systems
  - Load balancing
  - Utilizing spare CPU cycles
  - Providing fault tolerant systems
  - Manage access to powerful system, etc
- Basic architecture of RMS: client-server system

Services provided by RMS

- Process Migration
  - Computational resources have become too heavily loaded
  - Fault tolerant concern
- Checkpointing
- Scavenging Idle Cycles
  - 70% to 90% of the time most workstations are idle
- Fault Tolerance
- Minimization of Impact on Users
- Load Balancing
- Multiple Application Queues
Some Popular Resource Management Systems

<table>
<thead>
<tr>
<th>Project</th>
<th>Commercial Systems - URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSF</td>
<td><a href="http://www.platform.com/">http://www.platform.com/</a></td>
</tr>
<tr>
<td>CODINE</td>
<td><a href="http://www.genias.de/products/codine/tech_desc.html">http://www.genias.de/products/codine/tech_desc.html</a></td>
</tr>
<tr>
<td>Easy-LL</td>
<td><a href="http://www.tc.cornell.edu/UserDoc/SP/LL12/Leasy/">http://www.tc.cornell.edu/UserDoc/SP/LL12/Leasy/</a></td>
</tr>
<tr>
<td>NQE</td>
<td><a href="http://www.cray.com/products/software/nqe/">http://www.cray.com/products/software/nqe/</a></td>
</tr>
<tr>
<td>CONDOR</td>
<td><a href="http://www.cs.wisc.edu/condor/">http://www.cs.wisc.edu/condor/</a></td>
</tr>
<tr>
<td>GNQS</td>
<td><a href="http://www.gnqs.org/">http://www.gnqs.org/</a></td>
</tr>
<tr>
<td>DQS</td>
<td><a href="http://www.scri.fsu.edu/~pasko/dqs.html">http://www.scri.fsu.edu/~pasko/dqs.html</a></td>
</tr>
<tr>
<td>PRM</td>
<td><a href="http://gost.isi.edu/gost-group/products/prm/">http://gost.isi.edu/gost-group/products/prm/</a></td>
</tr>
<tr>
<td>PBS</td>
<td><a href="http://pbs.mrj.com/">http://pbs.mrj.com/</a></td>
</tr>
</tbody>
</table>

Programming Environments and Tools (I)

- **Threads (PCs, SMPs, NOW..)**
  - In multiprocessor systems
    - Used to simultaneously utilize all the available processors
  - In uniprocessor systems
    - Used to utilize the system resources effectively
  - Multithreaded applications offer quicker response to user input and run faster
  - Potentially portable, as there exists an IEEE standard for POSIX threads interface (pthreads)
  - Extensively used in developing both application and system software

Programming Environments and Tools (II)

- **Message Passing Systems (MPI and PVM)**
  - Allow efficient parallel programs to be written for distributed memory systems
  - 2 most popular high-level message-passing systems – PVM & MPI
  - PVM
    - both an environment & a message-passing library
  - MPI
    - a message passing specification, designed to be standard for distributed memory parallel computing using explicit message passing
    - attempt to establish a practical, portable, efficient, & flexible standard for message passing
  - generally, application developers prefer MPI, as it is fast becoming the de facto standard for message passing

Programming Environments and Tools (III)

- **Distributed Shared Memory (DSM) Systems**
  - Message-passing
    - the most efficient, widely used, programming paradigm on distributed memory systems
    - complex & difficult to program
  - Shared memory systems
    - offer a simple and general programming model
    - but suffer from scalability
  - DSM on distributed memory systems
    - alternative cost-effective solution
  - Software DSM
    - Usually built as a separate layer on top of the commun interface
    - Take full advantage of the application characteristics: virtual pages, objects, & language types are units of sharing
  - Hardware DSM
    - Better performance, no burden on user & SW layers, fine granularity of sharing, extensions of the cache coherence scheme, & increased HW complexity

Programming Environments and Tools (IV)

- **Parallel Debuggers and Profilers**
  - Debuggers
    - Very limited
    - HPDF (High Performance Debugging Forum) as Parallel Tools Consortium project in 1996
    - Developed a HPD version specification, which defines the functionality, semantics, & syntax for a commercial-line parallel debugger
  - TotalView
    - A commercial product from Dolphin Interconnect Solutions
    - The only widely available GUI-based parallel debugger that supports multiple HPC platforms
    - Only used in homogeneous environments, where each process of the parallel application being debugged must be running under the same version of the OS

Functionality of Parallel Debugger

- Managing multiple processes and multiple threads within a process
- Displaying each process in its own window
- Displaying source code, stack trace, and stack frame for one or more processes
- Diving into objects, subroutines, and functions
- Setting both source-level and machine-level breakpoints
- Sharing breakpoints between groups of processes
- Defining watch and evaluation points
- Displaying arrays and its slices
- Manipulating code variable and constants
Programming Environments and Tools (V)

- **Performance Analysis Tools**
  - Help a programmer to understand the performance characteristics of an application
  - Analyze & locate parts of an application that exhibit poor performance and create program bottlenecks
  - Major components
    - A means of inserting instrumentation calls to the performance monitoring routines into the user's applications
    - A run-time performance library that consists of a set of monitoring routines
    - A set of tools for processing and displaying the performance data
  - Issue with performance monitoring tools
    - Intrusiveness of the tracing calls and their impact on the application performance
    - Instrumentation affects the performance characteristics of the parallel application and thus provides a false view of its performance behavior

Performance Analysis and Visualization Tools

<table>
<thead>
<tr>
<th>Tool</th>
<th>Supports</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIMS</td>
<td>Instrumentation, monitoring library, analysis</td>
<td><a href="http://science.nasa.gov/Software/AIMS">http://science.nasa.gov/Software/AIMS</a></td>
</tr>
<tr>
<td>MPE</td>
<td>Monitoring library and analysis</td>
<td><a href="http://www.mcs.anl.gov/mpi/mpech">http://www.mcs.anl.gov/mpi/mpech</a></td>
</tr>
<tr>
<td>Pablo</td>
<td>Monitoring library and analysis</td>
<td><a href="http://www-pablo.cs.wisc.edu/Projects/Pablo/">http://www-pablo.cs.wisc.edu/Projects/Pablo/</a></td>
</tr>
<tr>
<td>Parodyn</td>
<td>Dynamic instrumentation monitoring analysis</td>
<td><a href="http://www.cs.wisc.edu/parodyn">http://www.cs.wisc.edu/parodyn</a></td>
</tr>
<tr>
<td>SvPablo</td>
<td>Integrated instrumentation, monitoring library and analysis</td>
<td><a href="http://www-pablo.cs.wisc.edu/Projects/Pablo/">http://www-pablo.cs.wisc.edu/Projects/Pablo/</a></td>
</tr>
<tr>
<td>Vampir</td>
<td>Monitoring library and analysis</td>
<td><a href="http://www.pallas.de/pages/vampir.htm">http://www.pallas.de/pages/vampir.htm</a></td>
</tr>
<tr>
<td>Dimemas</td>
<td>Performance prediction and message passing programs</td>
<td><a href="http://www.pallas.com/pages/dimemas.htm">http://www.pallas.com/pages/dimemas.htm</a></td>
</tr>
<tr>
<td>Paraver</td>
<td>Program visualization and analysis</td>
<td><a href="http://www.cepba.upc.es/paraver">http://www.cepba.upc.es/paraver</a></td>
</tr>
</tbody>
</table>

Programming Environments and Tools (VI)

- **Cluster Administration Tools**
  - Berkeley NOW
    - Gather & store data in a relational DB
    - Use Java applet to allow users to monitor a system
  - SMILE (Scalable Multicomputer Implementation using Low-cost Equipment)
    - Called K-CAP
    - Consist of compute nodes, a management node, & a client that can control and monitor the cluster
    - K-CAP uses a Java applet to connect to the management node through a predefined URL address in the cluster
  - PARMON
    - A comprehensive environment for monitoring large clusters
    - Use client-server techniques to provide transparent access to all nodes to be monitored
    - parmsvr-server & parmon-client

Need of more Computing Power: Grand Challenge Applications

Solving technology problems using computer modeling, simulation and analysis

- Geographic Information Systems
- Life Sciences
- Aerospace
- CAD/CAM
- Digital Biology
- Military Applications

Representative Cluster Systems (I)

- **The Berkeley Network of Workstations (NOW) Project**
  - Demonstrate building of a large-scale parallel computer system using mass produced commercial workstations & the latest commodity switch-based network components
  - Interprocess communication
    - Active Messages (AM)
      - basic communication primitives in Berkeley NOW
      - A simplified remote procedure call that can be implemented efficiently on a wide range of hardware
  - Global Layer Unix (GLUnix)
    - An OS layer designed to provide transparent remote execution, support for interactive parallel & sequential jobs, load balancing, & backward compatibility for existing application binaries
    - Aim to provide a cluster-wide namespace and use Network PIDs (NPIDs), and Virtual Node Numbers (VNNs)

Architecture of NOW System

- Sequential Applications
- Paralal Applications
- Global Layer Unix
- Network PIDs, NXNs, Virtual Node Numbers
Representative Cluster Systems (II)

- The Berkeley Network of Workstations (NOW) Project
  - Network RAM
    - Allow to utilize free resources on idle machines as a paging device for busy machines
    - Serverless
      - Any machine can be a server when it is idle, or a client when it needs more memory than physically available
  - xFS: Serverless Network File System
    - A serverless, distributed file system, which attempts to have low latency, high bandwidth access to file system data by distributing the functionality of the server among the clients
    - The function of locating data in xFS is distributed by having each client responsible for servicing requests on a subset of the files
    - File data is striped across multiple clients to provide high bandwidth

Representative Cluster Systems (III)

- The High Performance Virtual Machine (HPVM) Project
  - Deliver supercomputer performance on a low cost COTS system
  - Hide the complexities of a distributed system behind a clean interface
  - Challenges addressed by HPVM
    - Delivering high performance communication to standard, high-level APIs
    - Coordinating scheduling and resource management
    - Managing heterogeneity

HPVM Layered Architecture

<table>
<thead>
<tr>
<th>Applications</th>
<th>Fast Messages</th>
<th>MPI</th>
<th>SBMEM</th>
<th>Global Arrays</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fast Messages</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SBMEM</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Global Arrays</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fast Messages</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SBMEM</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Global Arrays</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Representative Cluster Systems (IV)

- The High Performance Virtual Machine (HPVM) Project
  - Fast Messages (FM)
    - A high bandwidth & low-latency comm protocol, based on Berkeley AM
    - Contains functions for sending long and short messages & for extracting messages from the network
    - Guarantees and controls the memory hierarchy
    - Guarantees reliable and ordered packet delivery as well as control over the scheduling of communication work
    - Originally developed on a Cray T3D & a cluster of SPARCstations connected by Myrinet hardware
    - Low-level software interface that delivers hardware communication performance
    - High-level layers interface offer greater functionality, application portability, and ease of use

Representative Cluster Systems (V)

- The Beowulf Project
  - Investigate the potential of PC clusters for performing computational tasks
  - Refer to a Pile-of-PCs (PoPC) to describe a loose ensemble or cluster of PCs
  - Emphasize the use of mass-market commodity components, dedicated processors, and the use of a private communication network
  - Achieve the best overall system cost/performance ratio for the cluster

Representative Cluster Systems (VI)

- The Beowulf Project
  - System Software
    - Grendel
      - The collection of software tools, resource management & support distributed applications
    - Communication
      - Through TCP/IP over Ethernet internal to cluster
      - Employ multiple Ethernet networks in parallel to satisfy the internal data transfer bandwidth required
      - Achieved by ‘channel binding’ techniques
    - Extend the Linux kernel to allow a loose ensemble of nodes to participate in a number of global namespaces
    - Two Global Process ID (GPID) schemes
      - Independent of external libraries
      - GPID-PVM compatible with PVM Task ID format & uses PVM as its signal transport
Representative Cluster Systems (VII)

- Solaris MC: A High Performance Operating System for Clusters
  - A distributed OS for a multicomputer, a cluster of computing nodes connected by a high-speed interconnect
  - Provide a single system image, making the cluster appear like a single machine to the user, to applications, and the network
  - Built as a globalization layer on top of the existing Solaris kernel
  - Interesting features
    - Extends existing Solaris OS
    - Preserves the existing Solaris ABI/API compliance
    - Provides support for high availability
    - Uses C++, IDL, CORBA in the kernel
    - Leverages spring technology

Solaris MC Architecture

Representative Cluster Systems (VIII)

- Solaris MC: A High Performance Operating System for Clusters
  - Uses an object-oriented framework for communication between nodes
    - Based on CORBA
    - Provide remote object method invocations
    - Provide object reference counting
    - Support multiple object handlers
  - Single system image features
    - Global file system
      - Distributed file system, called Proxy File System (PXFS), provides a globalized file system without need for modifying the existing file system
    - Globalized process management
    - Globalized network and I/O

Cluster System Comparison Matrix

<table>
<thead>
<tr>
<th>Project</th>
<th>Platform</th>
<th>Communications</th>
<th>OS</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>Beowulf</td>
<td>PCs</td>
<td>Multiple Ethernet with TCP/IP</td>
<td>Linux and OpenBSD</td>
<td>MPI2/PVM, PVM, MPI, Sockets and Pthreads</td>
</tr>
<tr>
<td>Berkeley Now</td>
<td>Solaris-based PCs and workstations</td>
<td>Myrinet and Active Messages</td>
<td>Solaris + GLinux + XFS</td>
<td>AM, PVM, MPI, HPP, Split-C</td>
</tr>
<tr>
<td>HPVM</td>
<td>PCs</td>
<td>Myrinet with Fast Messages</td>
<td>NT or Linux connection and global resource manager + LSF</td>
<td>Java-fronted, FM, Sockets, Global Arrays, SHE/MEM and MPI</td>
</tr>
<tr>
<td>Solaris MC</td>
<td>Solaris-based PCs and workstations</td>
<td>Solaris-supported</td>
<td>Solaris + globalization layer</td>
<td>C++ and CORBA</td>
</tr>
</tbody>
</table>

Cluster of SMPs (CLUMPS)

- Clusters of multiprocessors (CLUMPS)
  - To be the supercomputers of the future
  - Multiple SMPs with several network interfaces can be connected using high performance networks
  - 2 advantages
    - Benefit from the high performance, easy-to-use-and program SMP systems with a small number of CPUs
    - Clusters can be set up with moderate effort, resulting in easier administration and better support for data locality inside a node

Hardware and Software Trends

- Network performance increase of tenfold using 100BaseT Ethernet with full duplex support
- The availability of switched network circuits, including full crossbar switches for proprietary network technologies such as Myrinet
- Workstation performance has improved significantly
- Improvement of microprocessor performance has led to the availability of desktop PCs with performance of low-end workstations at significant low cost
- Performance gap between supercomputer and commodity-based clusters is closing rapidly
- Parallel supercomputers are now equipped with COTS components, especially microprocessors
- Increasing usage of SMP nodes with two to four processors
- The average number of transistors on a chip is growing by about 40% per annum
- The clock frequency growth rate is about 30% per annum
Advantages of using COTS-based Cluster Systems

- Price/performance when compared with a dedicated parallel supercomputer
- Incremental growth that often matches yearly funding patterns
- The provision of a multipurpose system

What are key attributes in HPC?

- Computation Time
- Communication Time