# UNIT-II

# Cluster Computing At a Glance

by

**Dr. K. Shahu Chatrapati**
**Professor & Addl. Controller of Examinations**
**Life Member of CSI, ISTE**

# Contents

# 1. Introduction

**Computing Power Limitations and Parallel Computing:**

• Parallel computers are systems that connect multiple processors to coordinate their computational efforts.

• These systems allow for the sharing of computational tasks among multiple processors.

Pfister suggests three ways to improve performance:

1. work harder 2. work smarter, and 3. Get help.

• Working harder involves using faster hardware, working smarter involves efficient algorithms and techniques, and getting help involves using multiple computers to solve a task.

# 2. Eras of Computing

The Computing Industry: Rapid Technological Developments

• Rapid technological advancements in computer hardware and software have driven the computing industry.

• Hardware advancements include chip development and fabrication technologies, fast and cheap microprocessors, and high bandwidth and low latency interconnection networks.

• VLSI technology has significantly contributed to the development of powerful sequential and parallel computers.

• Software technology is also rapidly developing, with mature software like Operating Systems, programming languages, development methodologies, and tools now available.

• Grand challenge applications like weather forecasting and earthquake analysis have driven the development of powerful parallel computers.

• Computing eras are viewed as Sequential Computing Era and Parallel Computing Era.

• Parallel computing technology needs to advance as it is not mature enough to be exploited as commodity technology.

• Parallel computers overcome the speed bottleneck of a single processor and offer a smaller price performance ratio.

• The chapter covers architecture alternatives for constructing parallel computers, motivations for transition to low cost parallel computing, a generic model of a cluster computer, commodity components used in building clusters, cluster middleware, resource management and scheduling, programming environments and tools, and representative cluster systems.

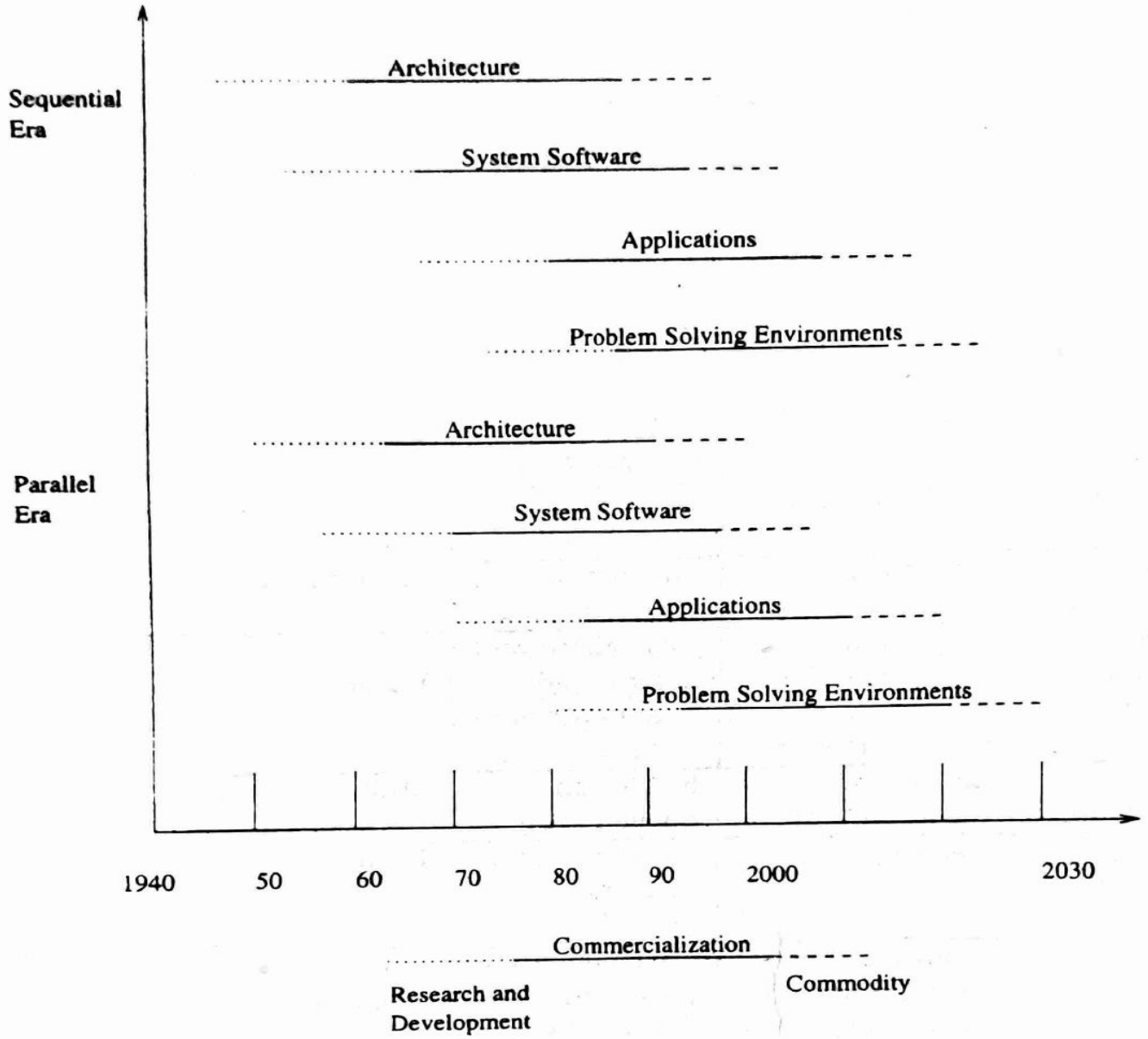


**Figure** Two eras of computing.

# 3. A Cluster Computer and its Architecture

**Cluster Computing Overview**

• A cluster is a parallel or distributed processing system consisting of interconnected stand-alone computers working together as a single, integrated computing resource.

• Computer nodes can be single or multiprocessor systems with memory, I/O facilities, and an operating system.

• Clusters can exist in a single cabinet or be physically separated and connected via a LAN.

• Cluster computers can provide cost-effective features and benefits found only on more expensive proprietary shared memory systems.

Key components of cluster computers include multiple high-performance computers, state-of-the-art operating systems, high-performance networks/switches, Network Interface Cards (NICS), fast communication protocols and services, cluster middleware, hardware, and operating system kernel or gluing layer.

• Applications and subsystems include system management tools, runtime systems, resource management and scheduling software, parallel programming environments and tools, and applications.

• Network interface hardware acts as a communication processor, transmitting and receiving data packets between cluster nodes via a network/switch.

• Cluster nodes can work collectively or operate as individual computers.

• Programming environments offer portable, efficient, and easy-to-use tools for application development, including message passing libraries, debuggers, and profilers.

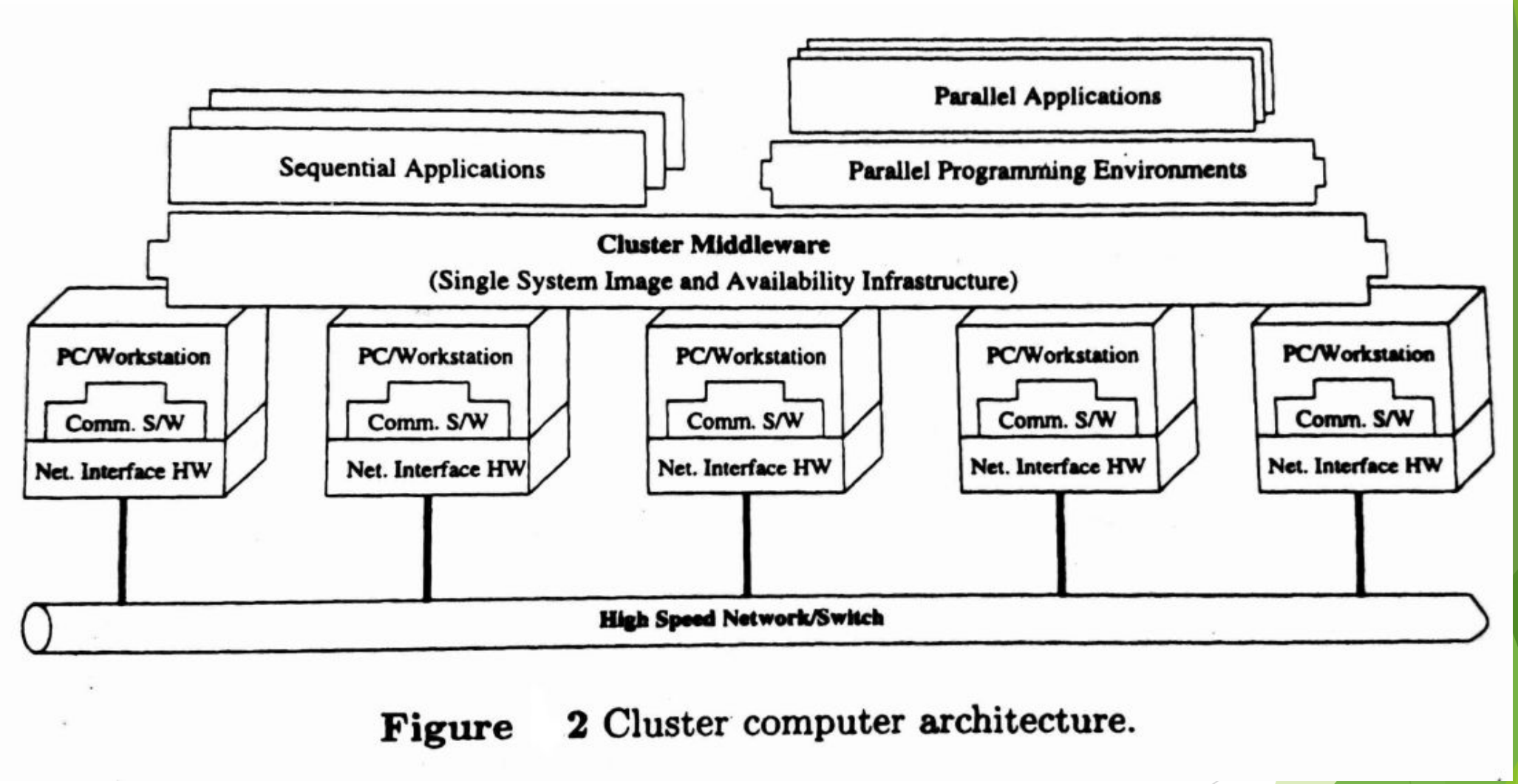


**Figure 2** Cluster computer architecture.

# 4. Clusters Classifications

**4.1 Cluster Technology Overview**

- Offers High Performance, Expandability and Scalability
- Provides high throughput and availability
- Allows organizations to increase their processing power using standard technology
- Enhances application performance with a scalable software environment
- Provides failover capability for a failed computer

**4.2 Clusters Classification**

- Application Target: Computational science or mission-critical applications
- High Performance (HP) Clusters
- High Availability (HA) Clusters

**4.3 Node Ownership**

• Dedicated Clusters: Shared resources for parallel computing across the entire cluster

• Nondedicated Clusters: Individuals own workstations and applications are executed by stealing idle CPU cycles

4.4 **Node Hardware PC, Workstation, or SMP**

• Clusters of PCs (CoPs) or Piles of PCs (POPs)

• Clusters of Workstations (COWS)

• Clusters of SMPS (CLUMPS)

4.5 **Node Operating System - Linux, NT, Solaris, AIX, etc.**

• Linux Clusters (e.g., Beowulf)

• Solaris Clusters (e.g., Berkeley NOW)

• NT Clusters (e.g., HPVM)

• AIX Clusters (e.g., IBM SP2)

• Digital VMS Clusters

• HP-UX clusters

• Microsoft Wolfpack clusters

## 4.6 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

## 4.7 Levels of Clustering

- Group Clusters (#nodes: 2-99)
- Departmental Clusters (#nodes: 10s to 100s)
- Organizational Clusters (#nodes: many 100s)
- National Metacomputers (WAN/Internet-based)
- International Metacomputers (Internet-based)

# 5. Commodity Components for Clusters

**Cluster-Based Parallel Systems: Progress and Challenges**

## 5.1 Processors

• Advances in microprocessor architecture have made single-chip CPUs nearly as powerful as supercomputer processors.

• Researchers are exploring the integration of processor and memory or network interface into a single chip.

• The Berkeley Intelligent RAM (IRAM) project is exploring the entire spectrum of issues involved in designing general purpose computer systems that integrate a processor and DRAM onto a single chip.

• Intel processors are most commonly used in PC-based computers, with the Pentium Pro and II offering strong integer performance.

• Other popular processors include x86 variants, Digital Alpha, IBM PowerPC, Sun SPARC, SGI MIPS, and HP PA.

## 5.2 Memory and Cache

• The memory present within a PC was originally 640 KBytes, usually 'hardwired' onto the motherboard.

• Computer systems can use various types of memory, including Extended Data Out (EDO) and fast page.

• The amount of memory needed for the cluster is likely to be determined by the cluster target applications.

• Access to DRAM is extremely slow compared to the speed of the processor, taking up to orders of magnitude more time than a CPU clock cycle.

• Caches are used to keep recently used blocks of memory for very fast access if the CPU references a word from that block again.

• Within Pentium-based machines, it is not uncommon to have a 64-bit wide memory bus and a chip set that supports 2 MBytes of external cache.

**5.3 Disk and I/0:**

• Improvements in disk access time have not kept pace with microprocessor performance, which has been improving by 50 percent or more per year.

• Grand challenge applications often need to process large amounts of data and data sets.

• To improve I/O performance, parallel file systems based on hardware or software RAID can be constructed by using disks associated with each workstation in the cluster.

**5.4 System Bus:**

• The initial PC bus (AT, now ISA bus) was clocked at 5 MHz and 8 bits wide.

• The performance of PCs has increased since the ISA bus was first used, but it has become a bottleneck.

• The ISA bus was extended to be 16 bits wide and clocked in excess of 13 MHz, but it still doesn't meet the demands of the latest CPUs, disk interfaces, and other peripherals.

• The VESA local bus, a 32-bit bus, was introduced by PC manufacturers.

• The Intel-created PCI bus, which allows 133 Mbytes/s transfers, has largely replaced the VESA bus.

## 5.5 Cluster Interconnects Overview

• Cluster interconnects use standard networking protocols like TCP/IP or Active Messages.

• Standard Ethernet is used for file and printer sharing, but its performance is showing its age.

• Gigabit Ethernet2 is the state-of-the-art Ethernet, offering high bandwidth and support for high-speed server connections, interswitch links, and workgroup networks.

## 5.6 **Asynchronous Transfer Mode (ATM)**

• ATM is a switched virtual-circuit technology developed for the telecommunications industry.

• It is intended for both LAN and WAN, presenting a unified approach to both.

• ATM uses small fixed-size data packets termed cells, which can be transferred using various media.

• ATM initially used optical fiber as the link technology, which is undesirable in desktop environments.

## 5.7 Scalable Coherent Interface (SCI)

• SCI is an IEEE 1596-1992 standard aimed at providing low-latency distributed shared memory across a cluster.
• It is the modern equivalent of a Processor-Memory-I/O bus and LAN combined.
• SCI is a point-to-point architecture with directory-based cache coherence.
• SCI has been favored for fast distributed shared memory support, but its scalability is constrained by the current generation of switches and its components are relatively expensive.

## 5.8 Myrinet

• Myrinet is a 1.28 Gbps full duplex interconnection network supplied by Myricom.
• It uses low latency cut-through routing switches, offering fault tolerance and simplifying network setup.
• Myrinet is relatively expensive compared to Fast Ethernet but has advantages such as very low-latency, high throughput, and a programmable on-board processor.
• However, its price compared to Fast Ethernet is in the range of $1,500 per host.

# 6. Network Services/Communication SW Overview

• Distributed applications require diverse communication needs, ranging from reliable point-to-point to unreliable multicast communications.

• The communications infrastructure supports protocols for bulk data transport, streaming data, group communications, and distributed objects.

• Communication services provide basic mechanisms for transporting administrative and user data, and provide quality of service parameters like latency, bandwidth, reliability, fault-tolerance, and jitter control.

Network services are typically designed as a hierarchical stack of protocols, with each protocol layer exploiting the services provided by the protocols below it.

• Traditionally, operating system services were used for communication between processes in message passing systems, involving expensive operations.

• Clusters with special networks/switch like Myrinet use lightweight communication protocols for fast communication among nodes, bypassing the operating system and providing direct, user-level access to the network interface.

• Network services are often built from a low-level communication API, supporting a wide range of high-level communication libraries and protocols.

## 7. Cluster Middleware and Single System Image (SSI)

### 7.1 Single System Image (SSI) and Middleware Layers

• SSI is a unified resource that is supported by a middleware layer between the operating system and user-level environment.

• SSI Infrastructure and System Availability Infrastructure are two sublayers of software infrastructure.

• SSI infrastructure binds operating systems on all nodes to provide unified access to system resources.

• System Availability Infrastructure enables cluster services like checkpointing, automatic failover, recovery from failure, and fault-tolerant support.

### 7.2 Single System Image Levels/Layers

• SSI can be applied to applications, specific subsystems, or the entire cluster.

• SSI layers support both cluster-aware (parallel applications developed using MPI) and non-aware applications (typically sequential programs).

• Clusters can function as an SMP or MPP system with a high degree of SSI, or as a distributed system with multiple system images.

### 7.3 Hardware Layer

• Systems like Digital's Memory Channel and hardware DSM offer SSI at hardware level and allow the user to view the cluster as a shared memory system.

• Operating System Kernel (Underware) or Gluing Layer supports efficient execution of parallel applications in an environment shared with sequential applications.

• Full cluster-wide SSI allows all physical resources and kernel resources to be visible and accessible from all nodes within the system.

• Most operating systems that support SSI are built as a layer on top of existing operating systems and perform global resource allocation.

## 7.4 Single System Image (SSI) Layer in Cluster Management

### Applications and Subsystems Layer (Middleware)

• SSI can be supported by applications and subsystems, presenting multiple components of an application as a single application.

• Application level SSI is crucial as it is what the end user sees.

• Cluster administration tools offer a single point of management and control SSI services.

• Subsystems provide software for creating an efficient cluster system.

• Run time systems, like cluster file systems, make disks attached to cluster nodes appear as a single large storage system.

• Global job scheduling systems manage resources and enable the scheduling of system activities and execution of applications.

## 7.5 Single System Image Boundaries

- Every SSI has a boundary.
- SSI support can exist at different levels within a system.
- A subsystem can make a collection of interconnected machines appear as one big machine.
- Another subsystem/application can make the same set of machines appear as a large database/storage system.

**7.6 Benefits of SSI**

- Provides a simple view of all system resources and activities.
- Frees the end user from knowing where an application will run.
- Allows the administrator to manage the entire cluster as a single entity.
- Reduces the risk of operator errors.
- Centralizes/decentralizes system management and control.
- Presents multiple, cooperating components of an application as a single application

## 7.7 Middleware Design Goals for Cluster-Based Systems

### 7.7.1 Complete Transparency

• The SSI layer should allow users to use a cluster easily and effectively without knowledge of the underlying system architecture.

• The operating environment should appear familiar and convenient to use, providing a view of a globalized file system, processes, and network.

• Resource management and control activities such as resource allocation, de-allocation, and replication are invisible to user processes.

### 7.7.2 Scalable Performance

• Clusters should be scalable without the need for new protocols and APIs.

• The SSI service should support load balancing and parallelism by distributing workload evenly among nodes.

• The time required to execute the same operation on a cluster should not be larger than on a single workstation.

### 7.7.3 Enhanced Availability

• Middleware services should be highly available at all times.

• A point of failure should be recoverable without affecting a user's application.

• Check pointing and fault tolerant technologies (hot standby, mirroring, failover, and failback services) should enable rollback recovery.

## 8. Key Services of SSI and Availability Infrastructure

### 8.1 SSI Support Services:

- Single Point of Entry: A user can connect to the cluster as a single system.
- Single File Hierarchy: A file system is seen as a single hierarchy of files and directories under the same root directory.
- Single Point of Management and Control: The entire cluster can be monitored or controlled from a single window using a single GUI tool.
- Single Virtual Networking: Any node can access any network connection throughout the cluster domain.
- Single Memory Space: The illusion of shared memory over memories associated with nodes of the cluster.
- Single Job Management System: A user can submit a job from any node using a transparent job submission mechanism.
- Single User Interface: The user should be able to use the cluster through a single GUI.

**8.2 Availability Support Functions:**

- Single I/O Space (SIOS): Allows any node to perform I/O operation on local or remotely located peripheral or disk devices.
- Single Process Space: Processes have a unique cluster-wide process id.
- Check pointing and Process Migration: Allows process state and intermediate computing results to be saved periodically.

**9. Resource Management and Scheduling (RMS) Overview**

• RMS is the process of distributing applications among computers to maximize their throughput and efficiently utilize available resources.

• The software consists of a resource manager and a resource scheduler, each responsible for tasks like locating and allocating computational resources, authentication, process creation, and migration.

• RMS has various benefits, including load balancing, utilizing spare CPU cycles, providing fault tolerant systems, and managed access to powerful systems.

• The basic RMS architecture is a client-server system, with each computer sharing computational resources running a server daemon.

- Applications can be run in interactive or batch mode, with batch mode being the most commonly used.
- RMS environments provide middleware services to enable heterogeneous environments of workstations, SMPs, and dedicated parallel platforms to be easily and efficiently utilized.
- Services provided by RMS include Process Migration, Checkpointing, Scavenging Idle Cycles, Fault Tolerance, Minimization of Impact on Users, Load Balancing, and Multiple Application Queues.
- There are many commercial and research packages available for RMS, with several in-depth reviews of the available RMS systems.

| Project | Commercial Systems URL |
| --- | --- |
| LSF | http://www.platform.com/ |
| CODINE | http://www.genias.de/products/codine/tech_desc.html |
| Easy-LL | http://www.tc.cornell.edu/User Doc/SP/LL12/Easy/ |
| NQE | http://www.cray.com/products/software/nge/ |
| | Public Domain Systems --URL |
| CONDOR | http://www.cs.wisc.edu/condor/ |
| GNQS | http://www.gnqs.org/ |
| DQS | http://www.scri.fsu.edu/~pasko/dqs.html |
| PRM | http://gost.isi.edu/gost-group/products/prm/ |
| PBS | http://pbs.mrj.com/ |

# 10. Programming Environments and Tools

**10.1 Threads**

• Threads are a popular paradigm for concurrent programming on both uniprocessor and multiprocessor machines.

• They exploit the asynchronous behavior of an application for overlapping computation and communication.

• Threads are potentially portable, with an IEEE standard for POSIX threads interface, pthreads.

• Programming languages like Java have built-in multithreading support, enabling easy development of multithreaded applications.

**10.2 Message Passing Systems (MPI and PVM)**

• Message passing libraries allow efficient parallel programs to be written for distributed memory systems.

• PVM is both an environment and a message passing library, used to run parallel applications on systems ranging from high-end supercomputers to clusters of workstations.

• MPI is a message passing specification designed to be standard for distributed memory parallel computing using explicit message passing.

• MPI is available on most of the HPC systems, including SMP machines.

## 10.3 Distributed Shared Memory (DSM) Systems

• Message passing is the most efficient programming paradigm on distributed memory systems.

• DSM enables shared-variable programming and can be implemented using software or hardware solutions.

• Software DSM systems are usually built as a separate layer on top of the communications interface, while hardware DSM systems have better performance, no burden on user and software layers, fine granularity of sharing, extensions of the cache coherence schemes, and increased hardware complexity.

## 10.4 Parallel Debuggers and Profilers in High Performance Applications

• Parallel debuggers and profilers are essential for efficient development of high performance applications.

• Most vendors of HPC systems provide debuggers and performance analyzers for their platforms.

• These tools should work in a heterogeneous environment, enabling parallel application development on a NOW and production runs on a dedicated HPC platform.

## 10.5 Debuggers

• The High Performance Debugging Forum (HPDF) was formed in 1996 to define a cross-platform parallel debugging standard.

• The forum developed a HPD Version specification defining the functionality, semantics, and syntax for a command-line parallel debugger.

• A parallel debugger should manage multiple processes and threads, display source code, stack trace, and stack frame, and manage code variables and constants.

**10.6 TotalView**

• Total View is a commercial product from Dolphin Interconnect Solutions that supports multiple HPC platforms.

• It supports most commonly used scientific languages, message passing libraries, and operating systems.

• Total View can only be used in homogeneous environments where each process of the parallel application must run under the same OS version.

**10.7 Performance Analysis Tools**

• Performance analysis tools help programmers understand the performance characteristics of an application.

• They include a means of inserting instrumentation calls to the performance monitoring routines, a run-time performance library, and tools for processing and displaying performance data.

| Tool | Supports | URL |
|---|---|---|
| **AIMS** | instrumentation, monitoring library, analysis | http://science.nas.nasa.gov/Software/AIMS |
| **MPE** | logging library and snapshot performance visualization | http://www.mcs.anl.gov/mpi/mpich |
| **Pablo** | monitoring library and analysis | http://www-pablo.cs.uiuc.edu/Projects/Pablo/ |
| **Paradyn** | dynamic instrumentation runtime analysis | http://www.cs.wisc.edu/paradyn |
| **SvPablo** | integrated instrumentor, monitoring library and analysis | http://www-pablo.cs.uiuc.edu/Projects/Pablo/ |
| **Vampir** | monitoring library performance visualization | http://www.pallas.de/pages/vampir.htm |
| **Dimemas** | performance prediction for message passing programs | http://www.pallas.com/pages/dimemas.htm |
| **Paraver** | program visualization and analysis | http://www.cepba.upc.es/paraver |

# 11. Cluster Administration Tools

**Tools for monitoring clusters using a GUI:**

- Essential for exploiting clusters as high-performance computing platforms.
- Berkeley NOW, SMILE, and PARMON are key projects.
- Berkeley NOW: Data storage and monitoring in a relational database.
- SMILE: K-CAP: Compute nodes, management node, and client for cluster control and monitoring.
- Node Status Reporter (NSR): Standard mechanism for measuring and accessing cluster status information.
- PARMON: Comprehensive environment for monitoring large clusters using client-server techniques.
- PARMON-server and parmon-client: System resource activities and utilization information provider and real-time cluster information visualization.

# 12. Cluster Applications

- Clusters can handle grand challenge or super computing applications.
- GCAs are fundamental problems in science and engineering with broad economic and scientific impact.
- GCAs are intractable without state-of-the-art parallel computers
- Resource requirements include processing time, memory, and communication needs.
- Examples include massive crystallographic and micro tomographic structural problems, protein dynamics, bio catalysis, relativistic quantum chemistry, virtual materials design, global climate modeling, and discrete event simulation.