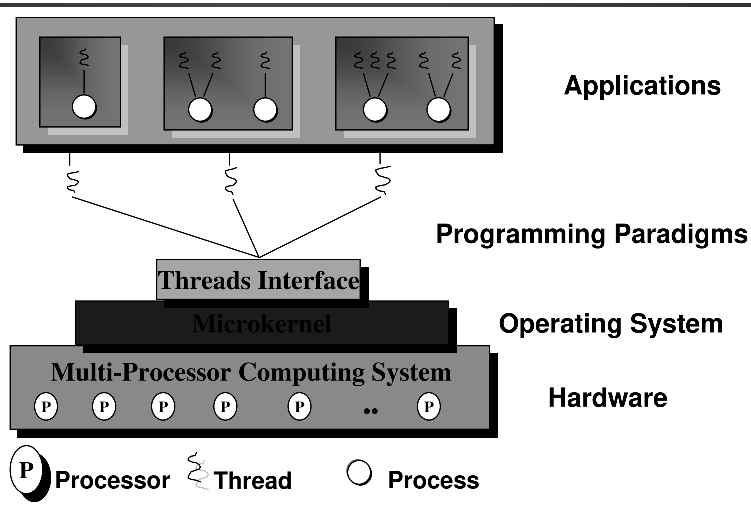
CSE 791: High-Performance Distributed Computing

Cluster Computing
At a Glance
(Some slides from R. Buyya or D. Culler)

Fundamental Terms
and Concepts

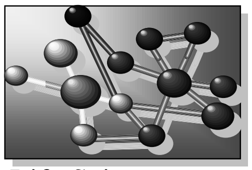
Computing Elements

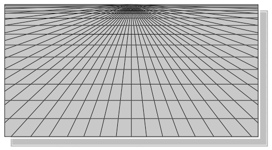


Computing Power and Computer Architectures

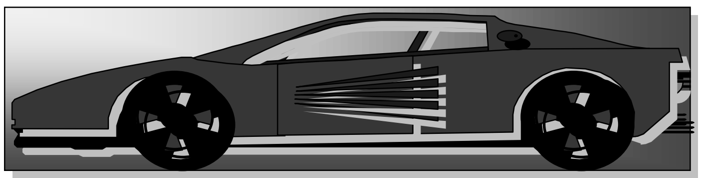
Need of more Computing Power: Grand Challenge Applications

Solving technology problems using computer *modeling*, *simulation* and *analysis*


Life Sciences


Aerospace


Geographic Information Systems


Mechanical Design & Analysis (CAD/CAM)

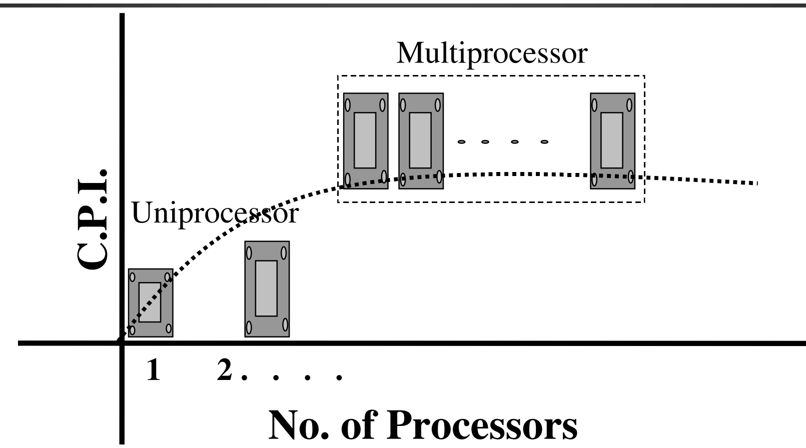
How to Run App. Faster ?

- There are 3 ways to improve performance:
  - **1. Work Harder**
  - **2. Work Smarter**
  - **3. Get Help**
- Computer Analogy
  - **1. Use faster hardware: e.g. reduce the time per instruction (clock cycle).**
  - **2. Optimized algorithms and techniques**
  - **3. Multiple computers to solve problem: That is, increase no. of instructions executed per clock cycle.**

Sequential Architecture Limitations

- Sequential architectures reaching physical limitation (speed of light, thermodynamics)
- Hardware improvements like pipelining, Superscalar, etc., are non-scalable and requires sophisticated Compiler Technology.
- Vector Processing works well for certain kind of problems.

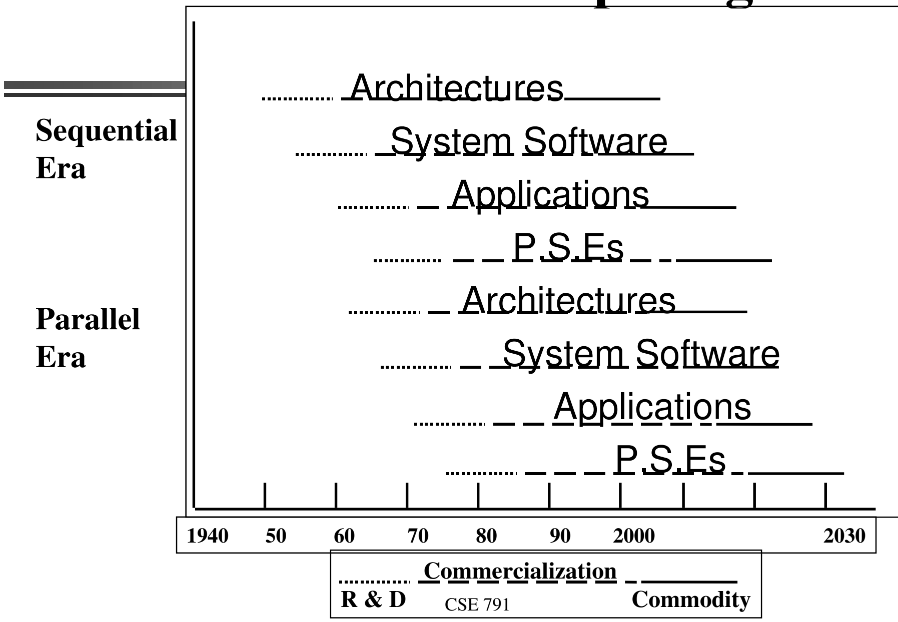
Computational Power Improvement



Why Parallel Processing NOW?

- The Tech. of PP is mature and can be exploited commercially; significant R & D work on development of tools & environment.
- Significant development in Networking technology is paving a way for heterogeneous computing.

Two Eras of Computing



Networking Infrastructure

- System-Area Networks (SAN), e.g. Myrinet
- Local-Area Networks (Gigabit Ethernet)
- Wide-Area Networks
  - gigabit-class WAN
  - QoS control in WAN

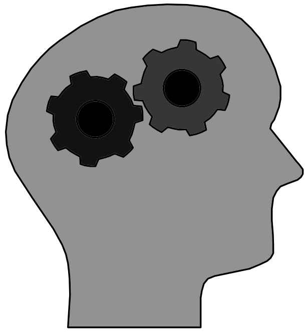
=> Networks at all levels can now support HPDC.

Cluster Interconnects

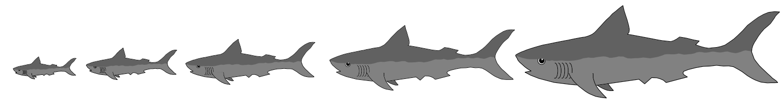
- Cost (Cheap/Expensive) - Latency (High/Medium/Low) - Bandwidth (H/M/L)
- C-M-L: Ethernet
- C-M-M: Fast Ethernet (100Mbps)
- E-M-H: Gigabit Ethernet
- E-L-H: Myrinet, ATM, FDDI, Fibre Channel, SCI, ...

Motivating Factors

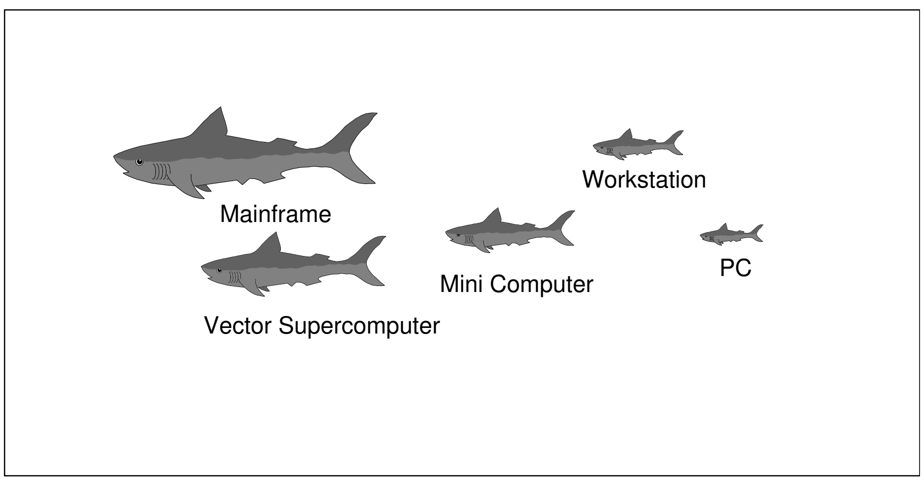
- Aggregated speed with which complex calculations can be carried out by millions of neurons in the human brain is amazing, although individual neurons response is slow (milliseconds). This is parallel processing in action!



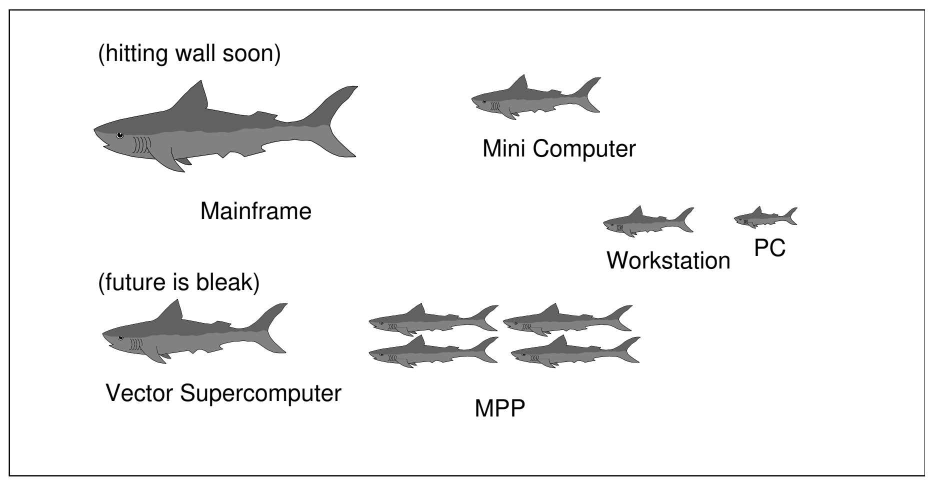
Original Food Chain Picture



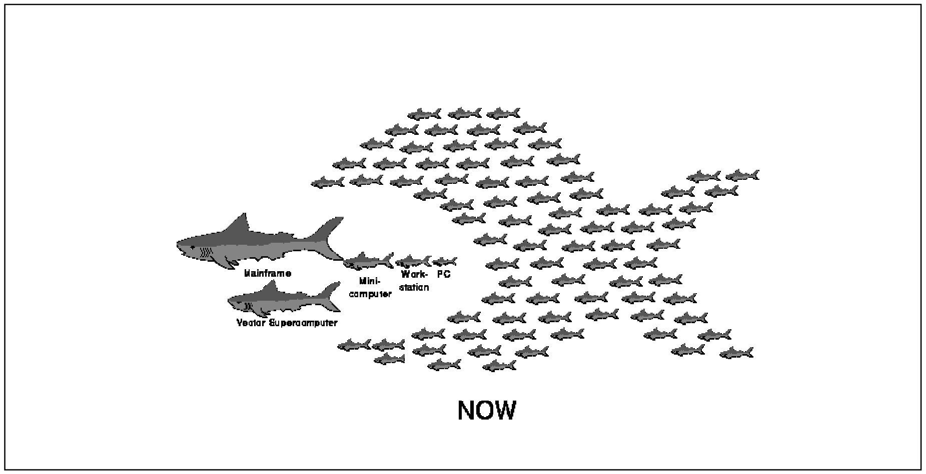
1984 Computer Food Chain



1994 Computer Food Chain



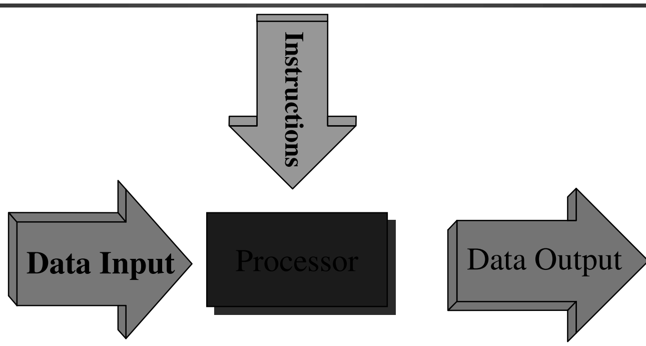
Computer Food Chain (Now and Future)



Taxonomy of Architectures

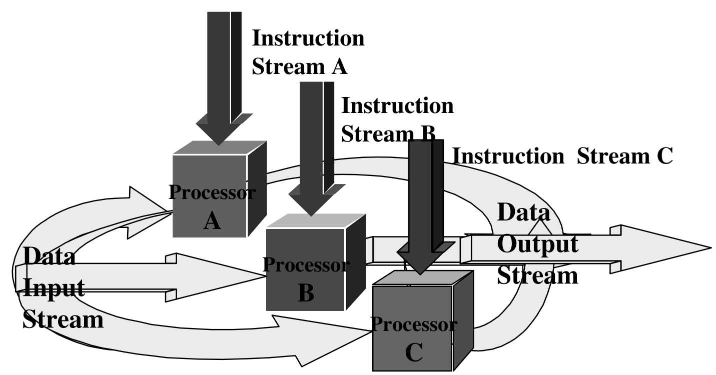
- Simple classification by Flynn:
  (No. of instruction and data streams)
  - **SISD - conventional**
  - **SIMD - data parallel, vector computing**
  - **MISD - systolic arrays**
  - **MIMD - very general, multiple approaches**
- Current focus is on MIMD model, using general purpose processors or multicomputers.

SISD : A Conventional Computer



→ Speed is limited by the rate at which computer can transfer information internally.

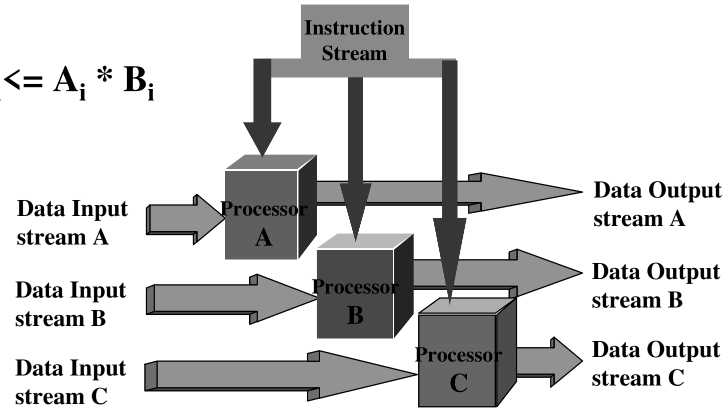
The MISD Architecture



→ **More of an intellectual exercise than a practical configuration.**

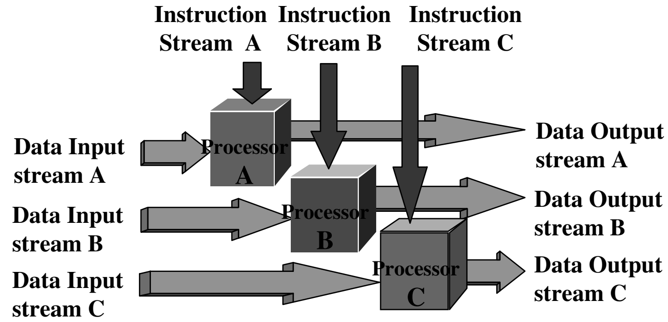
SIMD Architecture

$C_i <= A_i * B_i$



Ex: CRAY machine vector processing, Thinking machine cm*

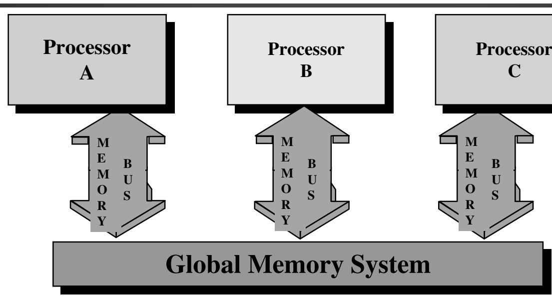
MIMD Architecture



Unlike SISD, MISD, MIMD computer works asynchronously.

- **Shared memory (tightly coupled) MIMD**
- **Distributed memory (loosely coupled) MIMD**

Shared Memory MIMD machine

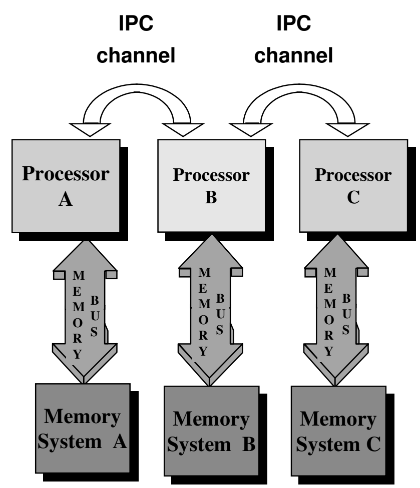


Comm: Source PE writes data to GM & destination retrieves it

- ➔ **Easy to build, conventional OS of SISD can be "easily" be ported**
- ➔ **Limitation : reliability & expandability. A memory component or any processor failure affects the whole system.**
- ➔ **Increase of processors leads to memory contention.**

Ex. : Silicon graphics supercomputers....

Distributed Memory MIMD



- **Communication : IPC on High Speed Network (ideally, a SAN).**
- **Network can be configured to ... Tree, Mesh, Cube, etc.**
- **Unlike Shared MIMD**
  - ➔ **easily/ readily expandable**
  - ➔ **Highly reliable (any CPU failure does not affect the whole system)**

Main HPC Architectures

- **SISD - mainframes, workstations, PCs.**
- **SIMD Shared Memory - Vector machines, Cray...**
- **MIMD Shared Memory - Sequent, KSR, Tera, SGI, SUN.**
- **SIMD Distributed Memory - DAP, TMC CM-2...**
- **MIMD Distributed Memory - Cray T3D, Intel, Transputers, TMC CM-5, plus recent workstation clusters (IBM SP2, DEC, Sun, HP).**

Main HPC Architectures II

- NOTE: Modern sequential machines are not purely SISD - advanced processors use many concepts from
  - vector and parallel architectures (pipelining, parallel execution of instructions, prefetching of data, etc) in order to achieve one or more arithmetic operations per clock cycle.
  - MMX instructions

Parallel Processing Paradox

- Time required to develop a parallel application for solving a Grand Challenge Application is equal to:
  - Half Life of Parallel Supercomputers.
  - Clearly, this is too long...

The Case for Alternative Supercomputing Resources

- Vast numbers of underutilized workstations available to use (cf. Condor).
- Huge numbers of unused processor cycles and resources that could be put to good use in a wide variety of applications areas.
- Reluctance to buy Supercomputer due to their cost and short life span.
- Distributed compute resources “fit” better into today's funding model.

Scalable Parallel Computers

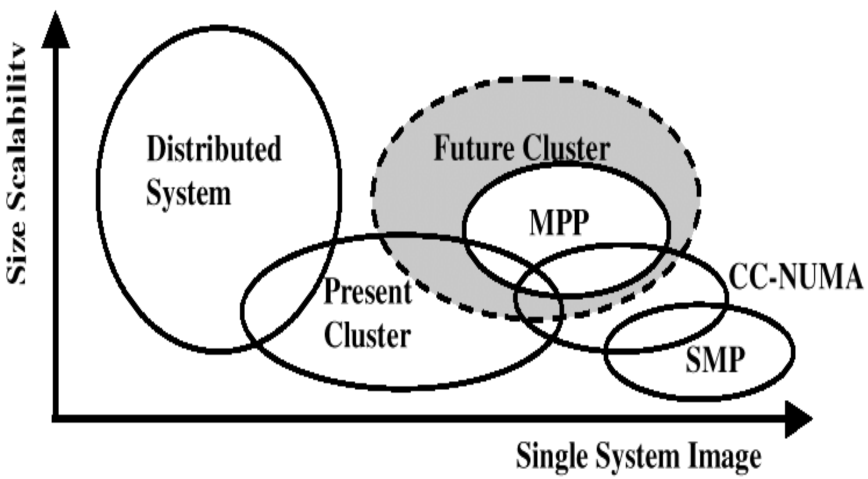
Characteristic	MPP	SMP, CC-NUMA	Cluster	Distributed System
Number of Nodes	O(100) - O(1000)	O(10) - O(100)	O(100) or less	O(10) - O(1000)
Node Complexity	Fine or medium grain	Medium or coarse grain	Medium grain	Wide range
Internode Communication	Message passing or shared variables for DSM	Centralized and distributed shared memory	Message Passing	Shared files, RPC, message passing, IPC protocol
Job Scheduling	Single run queue at host	Single run queue mostly	Multiple queues but coordinated	Independent multiple Queues
SSI Support	Partially	Always in SMP and some NUMA	Desired	No
Node OS Copies and Type	N microkernels and 1 monolithic OS at host	One monolithic for SMP and multiple for NUMA	N OS platforms (Homogeneous or microkernel)	N OS platforms (Heterogeneous)
Address Space	Multiple (Single for DSM)	Single	Multiple or single	Multiple
Internode Security	Unnecessary	Unnecessary	Required if exposed	Required
Ownership	One organization	One organization	One or More organizations	Many organizations
Network Protocol	Nonstandard	Nonstandard	Standard & Lightweight	Standard

Towards Inexpensive Supercomputing

Cluster Computing..

The Commodity Supercomputing!

Design Space of Competing Computer Architecture



Other Names

- Network of Workstations (NOW)
- Collection of Workstations (COW)
- Beowulf (named after the “cheapness first!” approach at JPL)
- Pile-of-PCs (another name for Beowulf)

Motivation for using Clusters (and especially cycle scavenging)

- Surveys show utilisation of CPU cycles of desktop workstations is typically <10%.
- Performance of workstations and PCs is rapidly improving
- As performance grows, percent utilisation will decrease even further!
- Organisations are reluctant to buy large supercomputers, due to the large expense and short useful life span.

Motivation for using Clusters

- The communications bandwidth between workstations is increasing as new networking technologies and protocols are implemented in LANs and WANs.
- Workstation clusters are easier to integrate into existing networks than special parallel computers.

Motivation for using Clusters

- The development tools for workstations are more mature than the contrasting proprietary solutions for parallel computers - mainly due to the non-standard nature of many parallel systems.
- Workstation clusters are a cheap and readily available alternative to specialised High Performance Computing (HPC) platforms.
- Use of clusters of workstations as a distributed compute resource is very cost effective - incremental growth of system!!!

Cycle Scavenging

- Usually a workstation will be *owned* by an individual, group, department, or organisation - they are dedicated to the exclusive use by the *owners*.
- This brings problems when attempting to form a cluster of workstations for running distributed applications.
- Remember what I said about autonomy.

Cycle Scavenging

- Typically, there are three types of owners, who use their workstations mostly for:
  1. Sending and receiving email and preparing documents.
  2. Software development - edit, compile, debug and test cycle.
  3. Running compute-intensive applications.

Cycle Scavenging

- Cluster computing aims to steal spare cycles from (1) and (2) to provide resources for (3).
- However, this requires overcoming the *ownership hurdle* - people are very protective of *their* workstations (autonomy, again!).
- Usually requires organizational mandate that computers are to be used this way.

Cycle Scavenging

- Stealing cycles outside standard work hours (e.g. overnight) is easy, stealing idle cycles during work hours without impacting interactive use (both CPU and memory) is much harder.

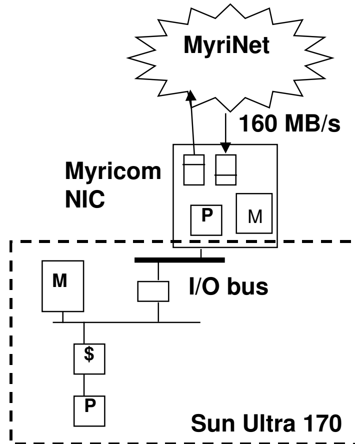
Cycle Scavenging

- Condor is the foremost cycle-scavenging project.
- Works *extremely* well for a class of applications and organizations
  - quick turnaround not required
  - single administrative domain
  - no coordination between processes
- But doesn't solve all problems

Example Clusters: Berkeley NOW

- 100 Sun UltraSparcs
  - 200 disks
- Myrinet SAN
  - 160 MB/s
- Fast comm.
  - AM, MPI, ...
- Ether/ATM switched external net
- Global OS
- Self Config

Basic Components



Massive Cheap Storage Cluster

- Basic unit:

  2 PCs double-ending four SCSI chains of 8 disks each

Currently serving Fine Art at http://www.thinker.org/imagebase/

Cluster of SMPs (CLUMPS)

- Four Sun E5000s
  - 8 processors
  - 4 Myricom NICs each
- Multiprocessor, Multi-NIC, Multi-Protocol
- NPACI => Sun 450s

Millennium PC Clumps

- Inexpensive, easy to manage Cluster
- Replicated in many departments
- Prototype for very large PC cluster

Centurion (and Orange Grove)

- So far, these have all been homogeneous
- But that ignores a strength of clusters
- Check out Centurion:
  - 128 2x Pentium, 128 Alpha
  - Myrinet, Fast Ethernet
- http://legion.virginia.edu/centurion

Adoption of the Approach

OPPORTUNITIES & CHALLENGES

Research Opportunities

- MPP/DSM:
  - ◆Compute across multiple systems: parallel.
- Network RAM:
  - ◆Idle memory in other nodes. Page across other nodes idle memory
- Software RAID:
  - ◆file system supporting parallel I/O and reliablity, mass-storage.

More Research Opportunities

- Multi-path Communication:
  - ◆Communicate across multiple networks: Ethernet, ATM, Myrinet
- Efficient thread support
  - ◆support for pthreads, OpenMP
- Scheduling
  - ◆coscheduling, job placement

Parallel Processing

- Scalable Parallel Applications require
  - ◆good floating-point performance
  - ◆low overhead communication scalable network bandwidth
  - ◆parallel file system

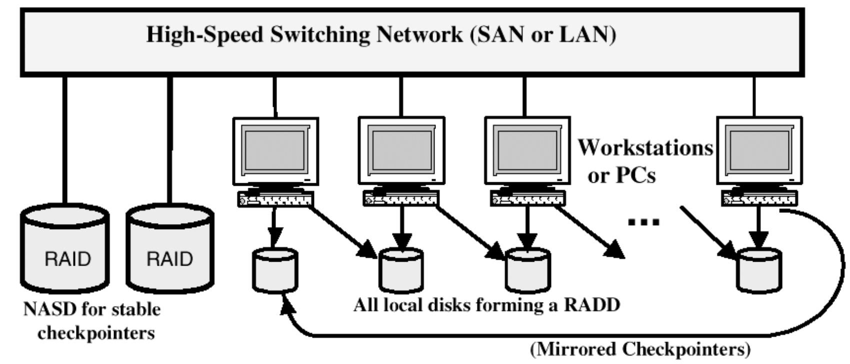
Network RAM

- Performance gap between processor and disk has widened.
- Paging to disk degrades performance significantly
- Paging across networks can be effective with high performance networks and OS that recognizes idle machines
- Typically paging to network RAM can be 5 to 10 times faster than paging to disk

Software RAID: Redundant Array of Workstation Disks

- I/O Bottleneck:
  - Microprocessor performance is improving more than 50% per year, disk access < 10%.
  - Application often perform I/O
- RAID cost per byte is high cf. single disks
- RAIDs are connected to host computers which are often a performance and availability bottleneck
- RAID in software, writing data across an array of workstation disks provides performance and some degree of redundancy provides availability.

Software RAID, Parallel File Systems, and Parallel I/O

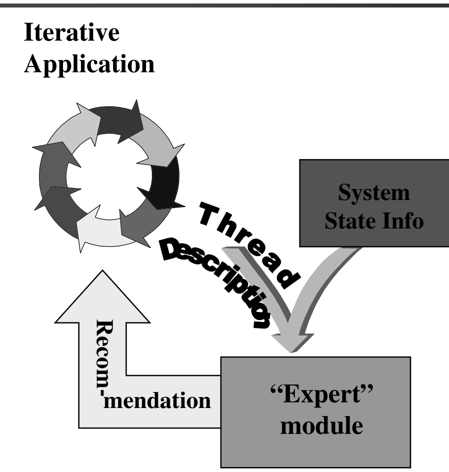


Multi-path Communications

- Many clusters have multiple networks, with differing latency/bandwidth
  - Fast Ethernet, Gigabit Ethernet
  - Myrinet
- Match traffic profile to network characteristics (control/data channels)
- Lilja, et al.

Dynamic Thread Support

- Of particular utility in heterogeneous clusters (mix of SMP nodes)
- When should an application start up or tear down threads?



Scheduling

- Coscheduling
  - Getting all the processes in a single job to run at the same time
  - non-obvious without OS coordination
  - Berkeley: implicit coscheduling
- Placement
  - Which node to use?
  - Instruction set, # proc, network, etc.

Enabling Technologies

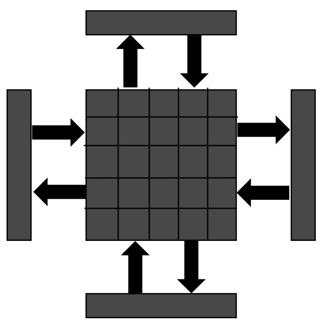
- Efficient communication hardware and software
- Global co-ordination of multiple workstation Operating Systems (some aspects of single system image)

Efficient Communication: *The key Enabling Technology*

- Communication overhead components
  - bandwidth
  - network latency
  - processor overhead
- Switched LANs allow bandwidth to scale
- Network latency can be overlapped with computation in some cases
- Processor overhead is the real problem - it consumes CPU cycles

Latency Hiding

- Grid-based computation exchanging boundary values
- Normal cycle: compute all, then exchange edges
- Optimization: compute edges first, then send, then compute rest, then read.



Processor Overhead

- SS10 connected by Ethernet
  - 456 μs processor overhead
- With ATM
  - 626 μs processor overhead
- Target :
  - MPP communication performance: low latency and scalable bandwidth
  - CM5 user-level network overhead 5.7 μs

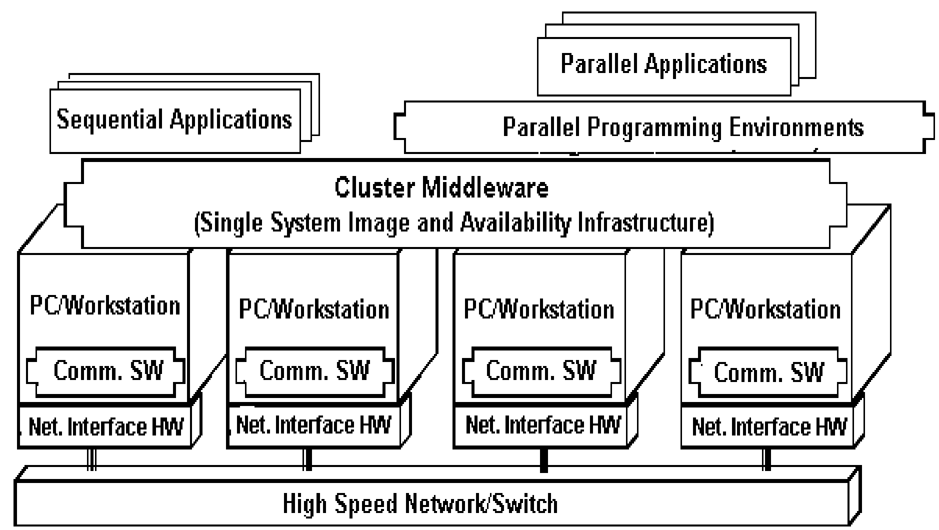
Cluster Communication Constraints

- Greater routing delay and less than complete reliability
- Constraints on where the network connects into the node (standard bus)
- UNIX has a rigid device and scheduling interface (remember coscheduling)
- WAN protocol stacks are too cumbersome

Efficient Communication Approaches

- Efficient Network Interface Hardware
- Minimal Interface into the Operating System (FM, Portals, etc.)
  - user must transmit directly into and receive from the network without OS intervention
  - communication protection domains to be established by interface card and OS
  - treat message loss as an infrequent case

Cluster Computer Architecture



Cluster Software Layers

- Development Tools
- Applications
- Programming Environment /Middleware
- Operating System and Communication Software

Communication Software

- Traditional OS supported facilities (heavy weight due to protocol processing)..
  - Sockets (TCP/IP), Pipes, etc.
- Light weight protocols (User Level)
  - Active Messages (Berkeley)
  - Fast Messages (Illinois)
  - U-net (Cornell)
  - XTP (Virginia)

Cluster Middleware

- Resides Between OS and Applications and offers in infrastructure for supporting:
  - Single System Image (SSI)
  - System Availability (SA)
- SSI makes collection appear as single machine (more later)
- SA - Check pointing and process migration

Programming Environments

- Threads (PCs, SMPs, NOW..)
  - POSIX Threads
  - OpenMP
  - Java Threads
- MPI, PVM
  - Linux, NT, on many Supercomputers
- Software DSMs (Treadmarks, etc.)
- Object systems (Legion, CORBA, DCOM)

Development Tools

- 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

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

Cluster Middleware and Single System Images

Contents

- What is Middleware ?
- What is a Single System Image ?
- Benefits of a Single System Image
- SSI Boundaries
- SSI Levels
- Relationship between Middleware Modules.
- Strategy for SSI via OS
- Solaris MC: An example OS supporting SSI
- Cluster Monitoring Software

Middleware Recap

- An interface between between user applications and cluster hardware and OS platform.
- Middleware Layers:
  - ◆SSI Layer
  - ◆Availability Layer: It enables the cluster services of checkpointing, failure recovery

Middleware Design Goals

- Complete Transparency
  - ◆Lets the user see a single cluster system.
    - – Single entry point, ftp, telnet, software loading.
- Scalable Performance
  - ◆Easy growth of cluster
    - – no change of API & automatic load distribution.
- Enhanced Availability
  - ◆Automatic Recovery from failures
  - ◆Handle consistency of data when replicated.

What is a Single System Image (SSI) ?

- A single system image is the illusion, created by software or hardware, that a collection of computing elements appear as a single computing resource.
- SSI makes the cluster appear like a single machine to the user, to applications, and to the network.
- R. Buyya: A cluster without an SSI is not a cluster

Benefits of Single System Image

- Usage of system resources transparently
- Improved reliability and higher availability
- Simplified system management
- Reduction in the risk of operator errors
- User need not be aware of the underlying system architecture to use these machines effectively

Buyya’s Desired SSI Services

- Single Entry Point
  - telnet cluster.my_institute.edu
  - telnet node1.cluster. institute.edu
- Single File Hierarchy: xFS, AFS, Solaris MC Proxy
- Single Control Point: Management from single GUI
- Single Virtual Network: any node can access any network in the cluster

Buyya’s Desired SSI Services II

- Single memory space - DSM
- Single Job Management System: Glunix, Codine, LSF, etc.
- Single User Interface: Like workstation/PC windowing environment (CDE in Solaris/NT), perhaps using web technology

Chapin’s Response

- The power of clusters is in their rich feature set => heterogeneity
- Stop trying to do everything as “one size fits all.”
- DSM on the whole cluster? You *must* be kidding! Kiss scalability goodbye.

Availability Support Functions

- Single I/O Space (SIO):
  - any node can access any peripheral or disk device without the knowledge of physical location.
- Single Process Space (SPS)
  - Any process on any node can create processes cluster wide, and they communicate through signal, pipes, etc., as if they are on a single node.

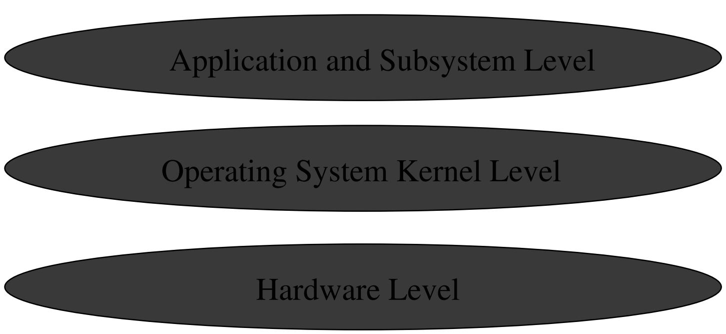
Availability Support Functions II

- Checkpointing and Process Migration.
  - ◆Saves the process state and intermediate results in memory to disk to support rollback recovery when node fails.
  - ◆Can be used for Load balancing.
- Reduction in the risk of operator errors
- User need not be aware of the underlying system architecture to use these machines effectively

Chapin's Response

- The scalability of the SSI is directly proportional to its level in the layering scheme
  - ◆low-level (e.g., I/O mapping): low scalability
  - ◆high-level (e.g. job management): high scalability
- Most of these ideas only scale to a few dozen nodes

SSI Levels



SSI at Application and Subsystem Level

Level	Examples	Boundary	Importance
application	cluster batch system, system management	an application	what a user wants
subsystem	distributed DB, OSF DME, Lotus Notes, MPI, PVM	a subsystem	SSI for all applications of the subsystem
file system	Sun NFS, OSF, DFS, NetWare, and so on	shared portion of the file system	implicitly supports many applications and subsystems
toolkit	OSF DCE, Sun ONC+, Apollo Domain	explicit toolkit facilities: user, service name,time	best level of support for heter-ogeneous system

(c) In search of clusters

SSI at Operating System Kernel Level

Level	Examples	Boundary	Importance
Kernel/ OS Layer	Solaris MC, Unixware MOSIX, Sprite,Amoeba / GLunix	each name space: files, processes, pipes, devices, etc.	kernel support for applications, adm subsystems
kernel interfaces	UNIX (Sun) vnode, Locus (IBM) vproc	type of kernel objects: files, processes, etc.	modularizes SSI code within kernel
virtual memory	none supporting operating system kernel	each distributed virtual memory space	may simplify implementation of kernel objects
microkernel	Mach, PARAS, Chorus, OSF/1AD, Amoeba	each service outside the microkernel	implicit SSI for all system services

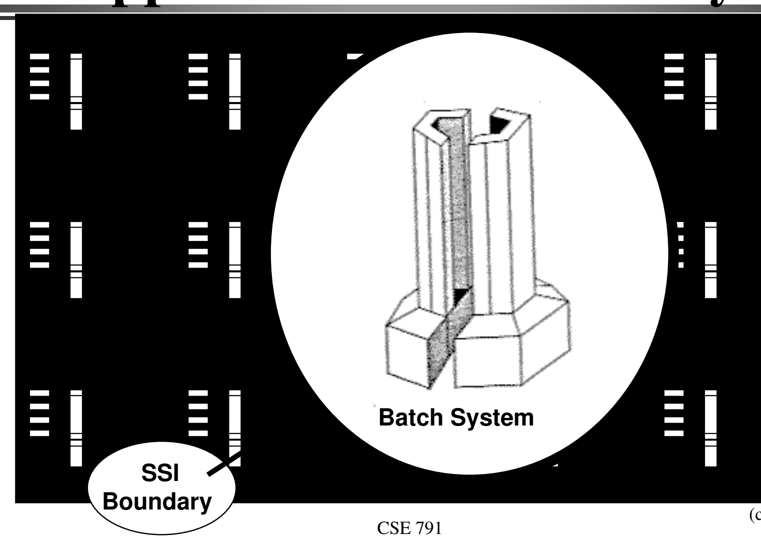
SSI at Harware Level

Level	Examples	Boundary	Importance
Application and Subsystem Level			
Operating System Kernel Level			
memory	SCI, DASH	memory space	better communica-tion and synchro-nization
memory and I/O	SCI, SMP techniques	memory and I/O device space	lower overhead cluster I/O

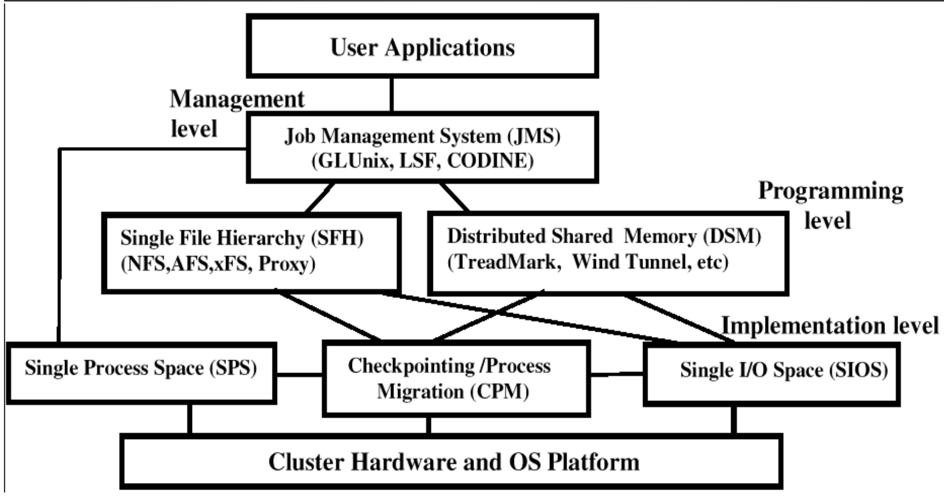
SSI Characteristics

- Every SSI has a boundary
- Single system support can exist at different levels within a system, one able to be built on another

SSI Boundaries -- an applications SSI boundary



Relationship Among Middleware Modules



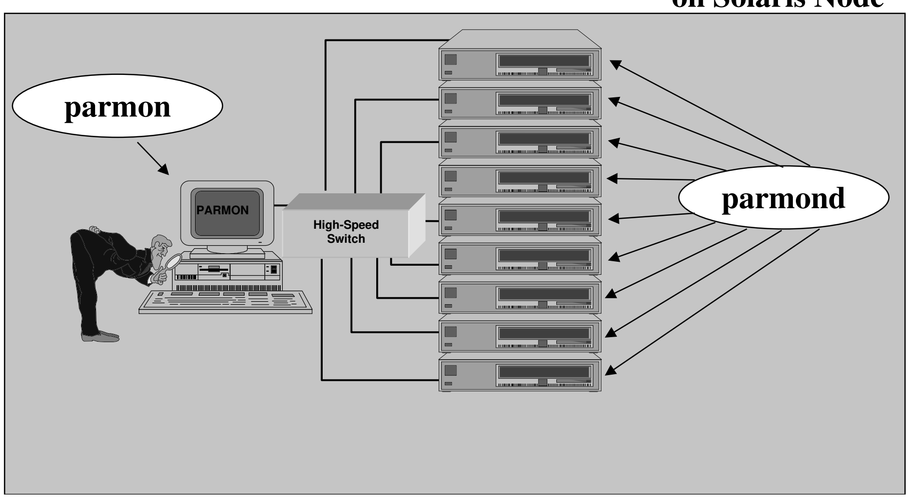
Cluster Monitoring

- Need to keep track of state of large application and the cluster

PARMON: A Cluster Monitoring Tool

PARMON Client on JVM

PARMON Server on Solaris Node



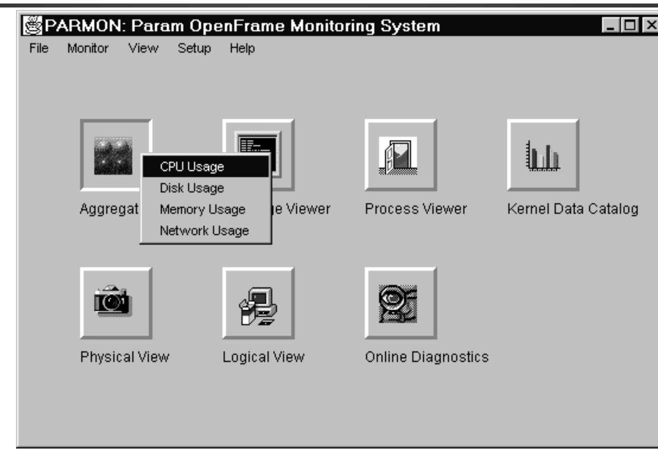
Motivations

- Monitoring such huge systems is a tedious and challenging task since typical workstations are designed to work as a standalone system, rather than a part of workstation clusters.
- System administrators require tools to effectively monitor such huge systems. PARMON provides the solution to this challenging problem.

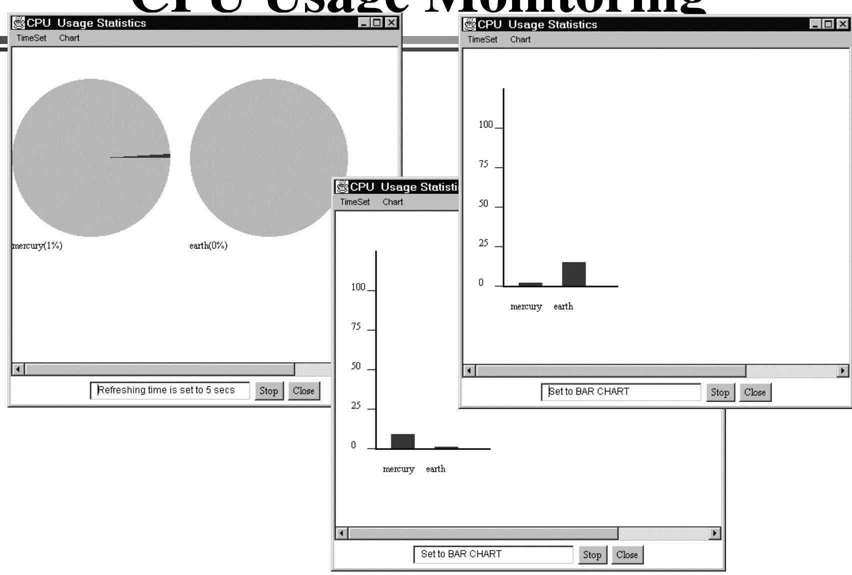
PARMON - Salient Features

- Allows to monitor system activities at Component, Node, Group, or entire Cluster level monitoring
- Monitoring of System Components :
  - ◆CPU, Memory, Disk and Network
- Allows to monitor multiple instances of the same component.
- PARMON provides GUI interface for initiating activities/request and presents

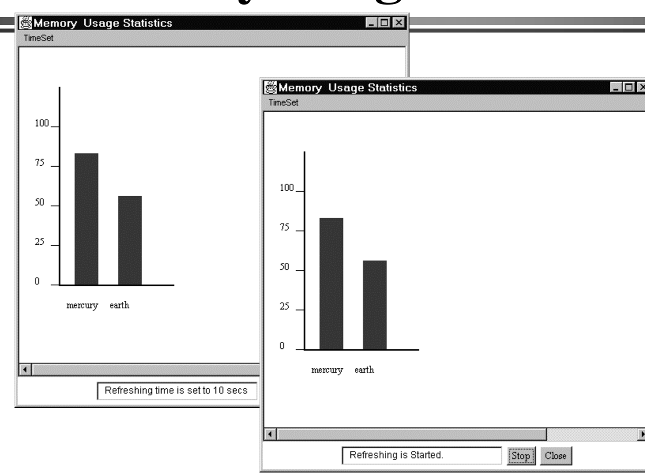
Resource Utilization at a Glance



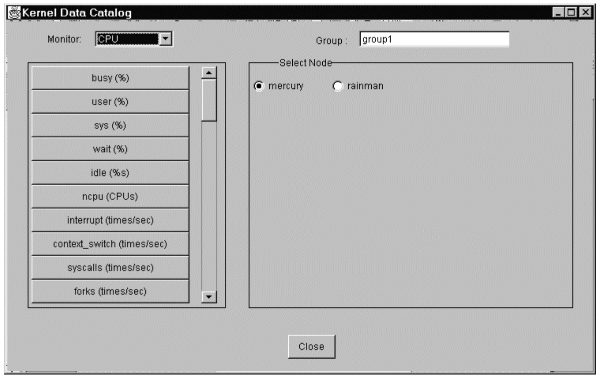
CPU Usage Monitoring



Memory Usage monitoring



Kernel Data Catalog - CPU



Strategy for SSI via OS

- Build as a layer on top of the existing OS. (eg. Glunix)
  - ◆Benefits: makes the system quickly portable, tracks vendor software upgrades, and reduces development time.
  - ◆i.e. new systems can be built quickly by mapping new services onto the functionality provided by the layer beneath. Eg: Glunix/Solaris-MC
  - ◆This is really low-level middleware...

Strategy for SSI in OS

- Build SSI at kernel level, True Cluster OS
  - ◆Good, but requires an open system
  - ◆**Can’t leverage of OS improvements by vendor**
  - ◆E.g. Unixware and Mosix (built using BSD Unix)
  - ◆Scalability will be an issue

Representative Cluster Systems

- Solaris -MC
- Berkeley NOW
- Beowulf
- HPVM

Solaris MC

Why new software?

- Without software, a cluster is:
  - **Just a network of machines**
  - **Requires specialized applications**
  - **Hard to administer**
- With a cluster operating system:
  - **Cluster becomes a scalable, modular computer**
  - **Users and administrators see a single large machine**
  - **Runs existing applications**
  - **Easy to administer**

Cluster computing and Solaris MC

- Goal: use computer clusters for general-purpose computing
- Support existing customers and applications
- Solution: Solaris MC (Multi Computer) operating system

A distributed operating system (OS) for multi-computers

What is the Solaris MC OS ?

- Solaris MC extends standard Solaris
- Predates current cluster vogue (1995)
- Solaris MC makes the cluster look like a single machine
  - **Global file system**
  - **Global process management**
  - **Global networking**
- Solaris MC runs existing applications unchanged (Solaris API)

Applications

- Ideal for:
  - **Web and interactive servers**
  - **Databases**
  - **File servers**
  - **Timesharing**
- Benefits for vendors and customers
  - **Preserves investment in existing applications**
  - **Modular servers with low entry-point price and low cost of ownership**
  - **Easier system administraion**
  - **Solaris could become a preferred platform for clustered systems**

Solaris MC is a running research system

- Designed, built and demonstrated Solaris MC prototype
  - **CLuster of SPARCstations connected with Myrinet network**
  - **Runs unmodified commercial parallel database, scalable Web server, parallel make**
- Next: Solaris MC Phase II
  - **High availability**
  - **New I/O work to take advantage of clusters**
  - **Performance evaluation**

Advantages of Solaris MC

- Leverages continuing investment in Solaris
  - Same applications: binary-compatible
  - Same kernel, device drivers, etc.
  - As portable as base Solaris - will run on SPARC, x86, PowerPC

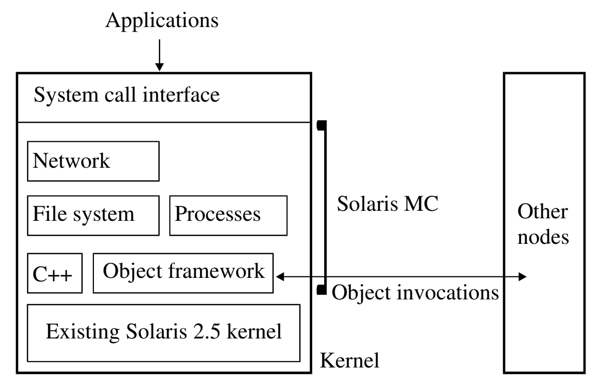
Solaris MC Advantages II

- State of the art distributed systems techniques
  - High availability designed into the sytem
  - Powerful distributed object-oriented framework
- Ease of administration and use
  - Looks like a familiar multiprocessor server to users, sytem administrators, and applications

Solaris MC details

- Solaris MC is a set of C++ loadable modules on top of Solaris
  - ◆Very few changes to existing kernel
- A private Solaris kernel per node: provides reliability
- Object-oriented system with well-defined interfaces

Key components of Solaris-MC proving SSI



Solaris MC Architecture

- global file system
- globalized process mgmt
- globalized networking and I/O

Solaris MC components

- Object and communication support
- High availability support
- PXFS global distributed file system
- Process mangement
- Networking

Object Orientation

- Better software maintenance, change, and evolution
  - ◆Well-defined interfaces
  - ◆Separate implementation from interface
  - ◆Interface inheritance
- Solaris MC uses:
  - ◆IDL: a better way to define interfaces
  - ◆CORBA object model: a better RPC
  - ◆C++: a better C

Object and Communication Framework

- Mechanism for nodes and modules to communicate
  - **Inter-node and intra-node interprocess communication**
- Optimized protocols for trusted computing base
- Efficient, low-latency communication primitives
- Object communication independent of interconnect
  - **Ethernet, fast Ethernet, FibreChannel, Myrinet**
  - **Allows interconnect hardware to be upgraded**

High Availability Support

- Node failure doesn't crash entire system
  - **Unaffected nodes continue running**
  - **Better than a SMP**
  - **A *requirement* for mission critical market**
- Well-defined failure boundaries
  - **Separate kernel per node - OS does not use shared memory**
- Object framework provides support
  - **Delivers failure notifications to servers and clients**
  - **Group membership protocol detects node failures**
- Each subsystem resposible for its recovery

PXFS: Global Filesystem

- Single-system image of file sytem
- Backbone of Solaris MC
- Coherent access and caching of files and directories
  - **Caching provides high performance**
- Access to I/O devices

PXFS: An object-oriented VFS

- PXFS builds on existing Solaris file sytems
  - **Uses the vnode/virtual file system interface (VFS) externally**
  - **Uses object communication internally**

Process management

- Provide global view of processes on any node
  - **Users, administrators, and applications see global view**
  - **Supports existing applications**
- Uniform support for local and remote processes
  - **Process creation/waiting/exiting (including remote execution)**
  - **Global process identifiers, groups, sessions**
  - **Signal handling**
  - **procfs (/proc)**

Process management benefits

- Global process management helps users and administrators
- Users see familiar single machine process model
- Can run programs on any node
- Location of process in the cluster doesn't matter
- Use existing commands and tools: unmodified ps, kill, etc.

Networking goals

- Cluster appears externally as a single SMP server
  - **Familiar to customers**
  - **Access cluster through single network address**
  - **Multiple network interfaces supported but not required**
- Scalable design
  - **protocol and network application processing on any mode**
  - **Parallelism provides high server performance**

Networking: Implementation

- A programmable "packet filter"
  - **Packets routed between network device and the correct node**
  - **Efficient, scalable, and supports parallelism**
  - **Supports multiple protocols with existing protocol stacks**
- Parallelism of protocol processing and applications
  - **Incoming connections are load-balanced across the cluster**

Status

- 4 node, 8 CPU prototype with Myrinet demonstrated
- Object and communication infrastructure
- Global file system (PXFS) with coherency and caching
- Networking TCP/IP with load balancing
- Global process management (ps, kill, exec, wait, rfork, /proc)

Status II

- Monitoring tools
- Cluster membership protocols
- Demonstrated applications
  - Commercial parallel database
  - Scalable Web server
  - Parallel make
  - Timesharing

Summary of Solaris MC

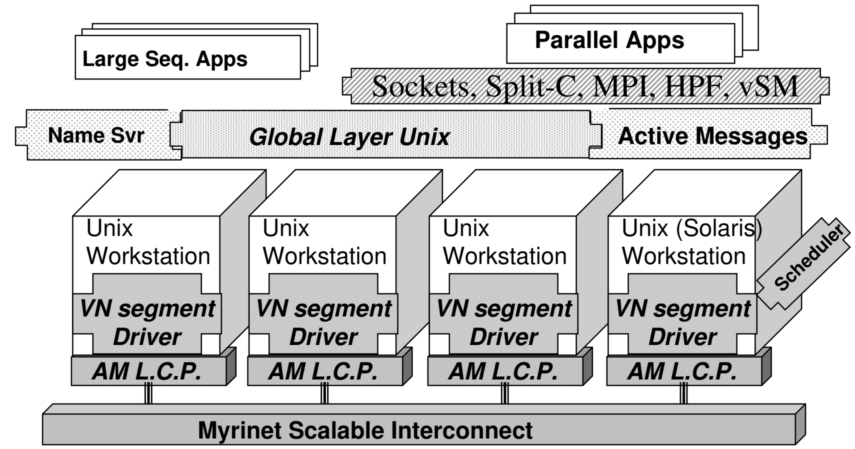
- **Clusters likely to be an important market**
- **Solaris MC preserves customer investment in Solaris**
  - **Uses existing Solaris applications**
- **Familiar to customers**
  - **Looks like a multiprocessor, not a special cluster architecture**
- **Ease of administration and use**
- **Clusters are ideal for important applications**
  - **Web server, file server, databases, interactive services**
- **State-of-the-art object-oriented distributed implementation**
  - **Designed for future growth**

Berkeley NOW Project

NOW @ Berkeley

- Design & Implementation of higher-level system
  - **Global OS (Glunix)**
  - **Parallel File Systems (xFS)**
  - **Fast Communication (HW for Active Messages)**
  - **Application Support**
- Overcoming technology shortcomings
  - **Fault tolerance**
  - **System Management**
- NOW Goal: Faster for Parallel AND Sequential

NOW Software Components



Active Messages: Lightweight Communication Protocol

- Key Idea: Network Process ID attached to every message that HW checks upon receipt
  - Net PID match, as fast as before
  - Net PID mismatch, interrupt and invoke OS
- Can mix LAN messages and MPP messages; invoke OS & TCP/IP only when not cooperating (if everyone uses same physical layer format)

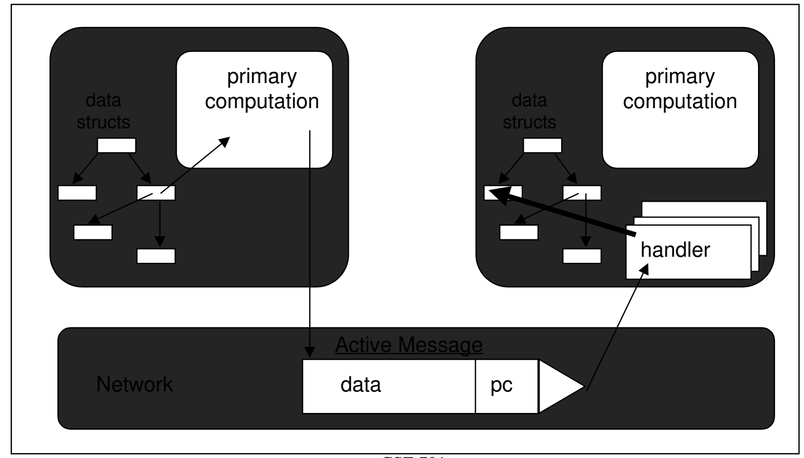
MPP Active Messages

- Key Idea: associate a small user-level handler directly with each message
  - **Sender injects the message directly into the network**
  - **Handler executes immediately upon arrival**
  - **Pulls the message out of the network and integrates it into the ongoing computation, or replies**
  - **No buffering (beyond transport), no parsing, no allocation, primitive scheduling**

Active Message Model

- **Every message contains at its header the address of a user level handler which gets executed immediately in user level**
- **No receive side buffering of messages**
- **Provides naming and protection**
- **Active message operations, communication events and threads are integrated in a simple and cohesive model**
- **Supports protected multiprogramming of a large number of users onto finite physical network resource**

Active Message Model (Contd..)



xFS: File System for NOW

- Serverless File System: All data with clients
  - **Uses MP cache coherency to reduce traffic**
- Files striped for parallel transfer
- Large file cache ("cooperative caching- Network RAM")

	Miss Rate	Response Time
Client/Server	10%	1.8 ms
xFS	4%	1.0 ms

Glunix: Gluing Unix

- It glues together Solaris running on Cluster nodes.
- Support transparent remote execution, load balancing, allows to run existing applications.
- Provides globalized view of system resources like SolarisMC
- Gang schedule parallel jobs to be as good as dedicated MPP for parallel jobs

3 Paths for Applications on NOW

- Revolutionary (MPP Style): write new programs from scratch using MPP languages, compilers, libraries,…
- Porting: port programs from mainframes, supercomputers, MPPs, …
- Evolutionary: take sequential program & use the following:

Evolutionary Approach

- **Network RAM: first use memory of many computers to reduce disk accesses; if not fast enough, then:**
- **Parallel I/O: use many disks in parallel for accesses not in file cache; if not fast enough, then:**
- **Parallel program: change program until it sees enough processors that is fast**
- **This yields large speedup without fine grain parallel program**

Comparison of 4 Cluster Systems

Project	Platform	Communications	OS	Other
Beowulf	PCs	Multiple Ethernet with TCP/IP with TCP/IP	Linux and Grendel	MPI/PVM, Sockets and HPF
Bereley NOW	Solaris-based PCs and workstations	Myrinet and Active Messages	Solaris + GLUunix + XFs	AM, PVM, MPI, HPF, Split-C
HPVM	PCs	Myrinet with Fast Messages	NT or Linux connection and global resource manager + LSF	Java-frontend, FM, Sockets, Global Arrays, SHMEM and MPI
Solaris MC	Solaris-based PCs and workstations	Solaris-supported	Solaris + Globalization layer	C++ and CORBA