High Performance Computing

Course Notes 2008-2009

Cluster Technologies
Design Space of Competing Computer Architecture

This is a diagram drawn ten years ago!
Original Food Chain Picture
1980s Computer Food Chain

- Mainframe
- Vector Supercomputer
- Mini Computer
- Workstation
- PC
Early 1990s Computer Food Chain

Mainframe

Vector Supercomputer

Workstation

MPP

Mini Computer

Obsolete

PC

(hitting wall soon)

(future is bleak)
Computer Food Chain (since late 1990s)
Computational Power Improvement

Multiprocessor

C.P.I.

Uniprocessor

No. of Processors

1 2 . . .
Human Physical Growth Analogy: Computational Power Improvement

Vertical 

Horizontal

Growth

Age

5 10 15 20 25 30 35 40 45
Why Clusters now?

Clustering gained new wave of interests when 3 technologies converged:

- 1. Very high performance Microprocessors
  - workstation performance = yesterday supercomputers
- 2. High speed communication
- 3. Standard tools for parallel/ distributed programming
Cluster Components...1
Nodes

Multiple High Performance Components:

- PCs
- Workstations
- SMPs (CLUMPS)

They can be based on different architectures and running different OS

But usually, the nodes in a cluster are homogenous - they have same architecture and performance and are installed with the same OS
Cluster Components...2
OS

OS used in various cluster 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 Components...3
High Performance Networks

Ethernet (10Mbps),
Fast Ethernet (100Mbps),
Gigabit Ethernet (1Gbps)
Myrinet
Infiniband
FDDI
Cluster Components...4
Network Interfaces

Network Interface Card

- Myrinet NIC
- Ethernet NIC
- ...
Cluster Components...5 Communication Software

Traditional OS supported facilities (heavy weight due to protocol processing)...

- Pipes, Sockets (TCP/IP), etc.

Light weight protocols (User Level)

- Active Messages (Berkeley): used in NOW system
- Fast Messages (Illinois): used in HPVM
- U-net (Cornell)
- XTP (Virginia)

System can be built on top of the above protocols
Cluster Components...6
Cluster Middleware

Resides Between OS and Applications and support:

- Single System Image (SSI)
- System Availability (SA)

SSI makes collection appear as single machine (globalised view of system resources).
ssh cluster.myinstitute.edu

SA - Check pointing and process migration..
Cluster Components... 7 Programming environments

Threads (PCs, SMPs, NOW..)
- POSIX Threads
- Java Threads

MPI
- Linux, Windows

PVM

DSMs
Cluster Components...8
Development Tools

Compilers

- C/C++/Java;
- MPICC

Debuggers: e.g. Buster (based on sequential debuggers like gdb and dbx)

Performance Analysis Tools and Visualization Tools: e.g. Vampire Trace
Cluster Components

Applications

Sequential

Parallel / Distributed

- Grand Challenging applications
  - Weather Forecasting
  - Computational Fluid Dynamics
  - Molecular Biology Modeling
  - Engineering Analysis (CAD/CAM)
  - ..................

- Web servers
Cluster Architectures

Cluster system architecture
Clusters Classification

Based on Focus (in Market)

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

Based on Workstation/PC Ownership

- Dedicated Clusters
- Non-dedicated clusters
Clusters Classification..3

Based on Node Architecture..

- Clusters of PCs (CoPs)
- Clusters of Workstations (COWs)
- Clusters of SMPs (CLUMPs)
Clusters Classification

Based on Node OS Type

- Linux Clusters (Beowulf)
- Solaris Clusters (Berkeley NOW)
- NT Clusters (HPVM)
- AIX Clusters (IBM SP2)
Clusters Classification..5

Based on node components architecture & configuration (Processor Arch, Node Type: PC/Workstation.. & OS: Linux/NT..):

- **Homogeneous Clusters**
  - All nodes will have similar configuration

- **Heterogeneous Clusters**
  - Nodes based on different processors and running different OSes.
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
A typical Cluster Computing Environment

Application

PVM / MPI / SSH

Hardware/OS

???
The missing link is provided by cluster middleware/underware.

Application

PVM / MPI / SSH

Middleware

Hardware / OS

Single Entry Point

ssh cluster.my_institute.edu

ssh node1.cluster.institute.edu
Benefits of Single System Image

- Usage of system resources transparently
- Transparent process migration and load balancing across nodes.
- Improved reliability and higher availability
- Improved system response time and performance
- 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
Resource Management System : Condor

- Condor was designed to make use of the spare capabilities in a network of pc/workstations.

- Dual modes
  - Non-dedicated mode (cycle stealing).
  - Dedicated mode that can manage clusters

- Condor uses resource ClassAd to advertise resources

- Jobs’ requirements are specified in job submission script

- Condor matchmaker matches the job requirements with resource requirements

- Condor Software is available for free on the web.

- Hundreds of Condor installations used worldwide in both academia and industry.
Resource Management System : Condor

Universes:

- Condor have the notion of an “execution universe”.
- Each universe is an environment that supports a particular set of functionalities.
- Condor has a number of these:
  - Vanilla - very basic execution functionality designed to support plain serial applications.
  - Standard - more advanced environment that supports checkpointing and process migration; use condor_compile to relink the executable.
Resource Management System: Condor

ClassAds:

- ClassAds allow Condor to match job resource requests with the most appropriate resource for running the job.

- Users (jobs) have constraints - Job ClassAds:
  - “must run on Linux... have 1 GB of memory available... 50 GB of disk space”

- Owners (machines) wish to advertise their resources but also have constraints - Machine ClassAds:
  - “this cluster consists of 32 Nodes... 2GHz Opterons... reject jobs from Ligang”

- Condor matchmaker reviews the Job ClassAds and the Machine ClassAds, then identifies a match that satisfies the requirements of both.
Submitting (from Condor manual):

- Write a condor job submission script to inform the system about the job:
  - includes the executable, universe, input, output and error files to use, command-line arguments, requirements.

- Can describe many jobs at once, each with different parameters.

```plaintext
# Simple condor_submit input file
Universe   = vanilla
Executable = my_job
Queue
```
Resource Management System : Condor

Condor Submission Process:

- After you have submitted the job submission script to Condor, it generates a ClassAd that describes your job(s).

- It is then up to Condor to identify the appropriate match between the generated job ClassAd and the Machine ClassAds.
# Example condor_submit input file

Universe = vanilla
Executable = /home/dps/condor/dan-job.condor
Input = myjob.stdin
Output = myjob.stdout
Error = myjob.stderr
Arguments = -X 10 -Y 20 -Z 30
Requirements = Memory >= 1024 && Disk > 50000
Queue
Condor daemons

→ condor_master
- It is responsible for keeping the rest of the Condor daemons running on each machine in a pool
- The master spawns the other daemon
- Runs on every machine in your Condor pool

→ condor_startd
- Any machine that wants to execute jobs needs to have this daemon running
- It advertises a machine ClassAd
- Responsible for enforcing the policy under which the jobs will be started, suspended, resumed, vacated, or killed

→ condor_starter
- Spawned by condor_startd
- Sets up the execution environment, creates a process to run the user job, and monitors the job running
- Upon job completion, sends back status information to the submitting machine, and exits
condor_schedd

- Any machine that allows users to submit jobs needs to have the daemon running
- Users submit jobs to the condor_schedd, where they are stored in the job queue.
- condor_submit, condor_q, or condor_rm connect to the condor_schedd to view and manipulate the job queue

condor_shadow

- Created by condor_schedd
- runs on the machine where a job was submitted
- Any system call performed on the remote execute machine is sent over the network to this daemon, and the shadow performs the system call (such as file I/O) on the submit machine and the result is sent back over the network to the remote job
→ **condor_collector**

- collecting all the information about the status of a Condor pool
- All other daemons periodically updates information to the collector
- These information contain all the information about the state of the daemons, the resources they represent, or resource requests of the submitted jobs
- **condor_status** connects to this daemon for information

→ **condor_negotiator**

- responsible for all the matchmaking within the Condor system
- responsible for enforcing user priorities in the system
Resource Management System : Condor

Most jobs are not independent:

- Dependencies exists between jobs.
- Second stage cannot start until first stage has completed.

Condor uses DAGMan - Directed Acyclic Graph Manager

- DAGMan allows you to specify dependencies between your Condor jobs, so it can manage them automatically for you.
- DAGs are the data structures used by DAGMan to represent these dependencies.
- Each job is a “node” in the DAG.
- Each node can have any number of “parent” or “children” nodes – as long as there are no loops.

*(example from Condor tutorial).*
A DAG is defined by another text file, listing each of nodes and their dependencies:

```
# diamond.dag
Job A a.sub
Job B b.sub
Job C c.sub
Job D d.sub
Parent A Child B C
Parent B C Child D
```

Each node will run the Condor job specified by an accompanying Condor submit file.