The Scyld Beowulf Cluster System:

Donald Becker
Scyld Computing Corporation

becker@scyld.com

Presented with MagicPoint

What does "Cluster" mean to you?

Cluster:

A widely used term meaning

- Independent computers
- Combined into a unified system
- Through software and networking

Cluster Types

Cluster Types:

- Scalable Performance Cluster
- High Availability (Fail-over) Cluster
- Resource Access Cluster

Linux Cluster Software

Linux has software for all clustering types

- Scalable Performance Cluster
  - Beowulf
  - Linux Virtual Server
- High Availability (Fail-over) Cluster
  - Piranha
- Resource Access Systems
  - GFS Global File System
  - Mosix
  - Grid software

What is Beowulf?

Beowulf is

- Scalable Performance Clusters based on
- Commodity hardware
- Private system network
- Open source software (Linux) infrastructure

Why clusters?

Much better price-performance than traditional supercomputers

As-needed scalability

Commodity platforms

- Performance growth rate
- Better continuity and availability
- Long-term viability

Cluster Advantages

Price for Performance

- 3X to 10X better
- Business market pays for engineering
- Efficient distribution and service channels

As-need Scalability

- New machines can be automatically added
- New, faster machines can replace older machines
- Architecture and software remains the same
- Investment preserved

Advantages of Commodity Systems

Commodity CPUs

- Always available
- Many vendors
- Multiple CPU development teams
- Rapid improvements

New technology

- Now arrives first on the PC

Industry Trends

CC-NUMA and SMP machines remain very expensive

Clusters are 65% Linux based, 30% other Unix

IDC reports 30%/year cluster market growth

What can Beowulf Systems be used for?

Supercomputer replacement

- Running specially written cluster applications
- Running multiple standard applications

Managing compute and server farms

- Load balancing network servers
- Constructing highly secure servers
- Controlling multiple highly available servers

Controlling other machines

- Single-purpose "kiosk" devices
- Audio servers

What are Beowulf Systems be used for?

- Traditional technical applications
  - Simulations
  - Biotechnology
  - Petro-cluster
- Financial market modeling
- Internet servers
  - Audio
  - Game servers

Very Brief Beowulf History

Beowulf Project

- The Beowulf Project was started at NASA in 1994
- Beowulf was intended to supplement supercomputers
- "Beowulf" was an apt project name
- Linux continues to be the dominant cluster OS

Scyld Beowulf

- Scyld was started in 1998
- Redesigned for ease of use and deployment
- Scyld Beowulf is the Scyld product
- "Scyld" was the father of Beowulf

Cluster Software

What to look for in cluster software system?

Well, what are the problems?

- Complexity
- Installation
- Applications to use the system
- Maintainence

Cluster Software Infrastructure

What to look for in cluster software system

- System management model
- Complexity minimization
- Application and tool availability
- Maturity and continuity

Previous Solutions

How have these cluster problems been address in the past?

Classic Beowulf

- Full OS installation on all nodes
- Supports user login on any node
- Administration by collective operations
- Consistency and synchronization tools
- Cluster monitoring GUI

New-generation Solution

How have we improved the world?

New-generation Beowulf

- ☐ Full OS installation only on "master"
- ☐ Compute nodes designed as a computational resource
- ☐ Single point administration
- ☐ Single point updates
- ☐ Single process space view
- ☐ Centralized monitoring and job control

Scyld Beowulf

A standard, supported Beowulf cluster operating system
Simplifies integration and administration
Targeting deployment of complex applications

What it is not:

- automatic parallelization
- a new language, or
- an integrated development environment.

Scyld Beowulf Features

- "Install once, execute everywhere"
- Administration and use is very similar to a single machine
- Dynamically adding compute nodes is fast and automatic
- Scalable to over a thousand compute nodes
- Software version skew has been eliminated.
- Based on Linux
- Open Source software infrastructure

Design Philosophy and Goals

Administrators

- Simplicity
- Minimal new cluster-specific tools

Users

- Application users should not need to know they are on a cluster
- Administration should require little new knowledge

Developers

- Need to be sophisticated only in application area
- Compile-run development cycle, not compile-copy-run
- Deployment with a single executable

System Model

- "Master" front-end
- Multiple "Slave node" compute machines
- Booting and configuration controlled from a master

- Master
  - Full operating system installation
  - Provides OS, drivers, libraries and applications
  - Supports user login
- Slaves
  - Have a full kernel
  - Only a minimal file system
  - No user logins
  - No required executables!

- Processes are started with a remote execution system

Scyld Beowulf Single System Image

- Single Installation
- Single point upgrade
  - Kernel, drivers, system libraries
  - User applications, user libraries
- No version skew
- Zero-installation scaling
  - Full performance on compute nodes
  - File system semantics selected
    - At system integration, or
    - By administrator
- Unified process space

Operational Details

Nodes are added dynamically

A heartbeat is used to detect lost systems

Detection of lost system connection

- Compute node default is rebooting after 30 seconds
- Configurable behavior

Advanced Features

- Cluster security model
  - User / group node ownership
- Limits on process migration
  - "Jailed" processes on slave
  - "No Hijack" external server slaves
- Multiple masters
- Scheduler interface
- Checkpoint / restart

In-depth Subsystems Description

Unified Process Space

Beowulf Name Services

Booting Scyld Clusters

Unified Process Space

Problem: Starting and Monitoring jobs on a cluster

Opportunity: Clusters jobs are issued from designated masters

Unified Process Space

- All jobs appear to exist on the front-end "master".
- Job control and process monitoring work as expected!
- Control-Z suspends all jobs, "bg" starts all running
- The ’ps’ and ’top’ programs work unchanged

Running top

The output from an unmodified 'top' run

```
 11:13pm  up  4:19, 10 users,  load average: 0.00, 0.00, 0.00
73 processes: 64 sleeping, 9 running, 0 zombie, 0 stopped
CPU states:  0.2% user,  0.5% system,  0.0% nice, 99.1% idle
Mem:   128156K av,  122384K used,    5772K free,   30500K shrd,   23608K buff
Swap:  265064K av,     212K used,  264852K free                   74292K cached

  PID USER     PRI  NI  SIZE  RSS SHARE STAT  LIB %CPU %MEM   TIME COMMAND
 1948 becker     0   0     0    0     0 RW      0 99.9  0.0   2:44 cpumunch
 1949 becker     0   0     0    0     0 RW      0 99.9  0.0   2:44 cpumunch
 1950 becker     0   0     0    0     0 RW      0 99.9  0.0   2:44 cpumunch
 1951 becker     0   0     0    0     0 RW      0 99.9  0.0   2:44 cpumunch
 1952 becker     0   0     0    0     0 RW      0 99.9  0.0   2:44 cpumunch
 1953 becker     0   0     0    0     0 RW      0 99.9  0.0   2:44 cpumunch
 1954 becker     0   0     0    0     0 RW      0 99.9  0.0   2:44 cpumunch
 1955 becker     0   0     0    0     0 RW      0 99.9  0.0   2:44 cpumunch
```

BProc: Beowulf Distributed Process Space

BProc is the kernel mechanism

- Remote Fork model
- A process is initialized on the master
- "VMA-dump" in the kernel writes the process to a stream
- On the slave node the stream is loaded as a new binary type
- The master retains only a process table entry

Performance Characteristics

Start-up

- Under 10 msec. to execute a remote job!
- bpsh uptime takes 0.6 seconds on 64 slow nodes.
- This is about 10X faster than rsh, 20X ssh

No run time performance impact

- System calls and paging are local
- Process status update to master is compact and low-rate
- Compare to perfect process migration of Mosix

How BProc works

BProc is a "Directed Process Migration" Mechanism

BProc has architectural elements of

- Remote Fork
- Process migration
- Checkpoint / restart

Design details

- VMA dump and restart -- essentially "checkpoint" to a socket/stream
- In general, files and sockets are closed
  - stdin, stdout, and stderr may remain connected
- Process environment info (process ID) appears unchanged
- Preserves Posix process family semantics
- Signals (SIG*) are forwarded both ways.
- Slave updates state to master.
- Resource usage on exit

How can this be Fast?

- Cached libraries ("VMA regions")
- Copy on changed pages in known VMA regions
- Copy unknown VMA regions

Developing improvements

- Dynamic caching of objects
- Caching on swap space disks
- Automatic selection of caching or network file system

Name Service / Directory Service

"Name Service" and "Directory Service" mean the same thing.

A directory service

- Maps a name to a value, or
- Provides a list of names.

Specific Examples

- User names
  - Password and user information
- Host names
  - IP addresses and Ethernet MAC addresses
- Network groups
  - A list of similar hosts

Cluster Name Services

Why are cluster nameservices important?

Simplicity

- ☐ Eliminates per-node configuration files
- ☐ Automates scaling and updates

Performance

- ☐ Avoid the serialization of network name lookups.
- ☐ Avoid communicating with a busy server
- ☐ Avoid failures from server overload
- ☐ Avoid the latency of consulting large databases

Opportunities

Clusters have a single set of users
Nodes are similar
New nodes will have predictable names
Cluster nodes are granted similar access permissions

Solution: BeoNSS, Beowulf Name Services

BeoNSS is a mechanism that

- Caches,
- Computes or
- Avoids name lookup

Hostnames

Cluster hostnames have the form .<N>

Syntax does not conflict

- Compare with DNS and local hostnames

Special names for "self" and "master"

- Current machine is ".-2" or "self".
- Master is known as ".-1"
  - Aliases of "master" and "master0".

Cluster nodes start at ".0"

- Zero based for flexibility
- Do not assign ".0" for 1-based naming
- Extend to maximum node e.g. ".31"
- Maximum resolvable number defined.

Mapping Hostnames to IP Addresses

Required information
- IP address of master
- IP address of first compute node
- Count of compute nodes

Additional configuration information
- Netmask
- Node station address

Simple computation of IP address
- First Node IP + Node number
- Little endian addition
- Netmask must support full cluster

User Name lookups

Names are reported as password table entry ’pwent’

BeoNSS reports only the current user

- Cluster jobs do not need to know other users
- Much faster than scanning large lists

Use BProc name (full passwd entry) if available

Otherwise compute ’pwent’ from environment variable

- USER
- HOME
- SHELL

No security issues for correctly written programs

- Programs should check for UID = 0, not username == "root"

Netgroups

Netgroups are used primarily for file server permissions

- Netgroups are used in /etc/exports for NFS
- Other file systems have similar security
- Other uses, such as rexec and rsh, are supported

The supported netgroup is "cluster"

- Alias for "cluster0"

Group members are all compute nodes

- Hostnames are reported as ".0", ".1" ... ".31"
- Maximum cluster node count is important

Use ’getnetgrent()’ to access this name service.

Booting Scyld Beowulf

Booting has long been a hot topic

- Various boot media
- Disk-based and Disk-less models
- Kernel and driver updates problematic

We solve the problems with a two phase boot

- Similar to the model used for Linux 2.5

Beowulf in Two Phases

Booting Scyld Beowulf Compute Nodes

- Magic Boot
- Two Kernel Monte, and an Intermediate FS
- The final running system

Phase 1 Boot: Magic Boot

Concept: Convert to all boot methods to network boot

Details:

- Minimal Kernel
  - Only IP networking support
  - No boot options possible
- Simple "RAMdisk"
  - Boot program with "insmod" module support
  - All known network driver modules
- Mounts /proc
- Loads drivers using /proc/bus/pci
- RARP requests on all interfaces
- Loads boot image from the responding server.
- Syscall to Two Kernel Monte

Two Kernel Monte

Concept: Switch Kernel

Details:

- Scyld-developed mechanism to switch kernels
- Substitutes a new kernel in place of the old
- Similar to a reboot, but without
  - going through the BIOS, or
  - using persistent media

Two Kernel Monte Implementation

Implemented as a kernel module
No kernel patches (even new exported symbols) required
The replacement kernel is loaded with the module.
Much hidden work is involved in setting up legacy BIOS tables

Beoboot Second Stage

Phase 2 Boot: Our Operational Kernel

- The new kernel starts up on the compute node
- The second stage RAMdisk is loaded
- The node repeats the network interface detection and RARP
- The "slave daemon" /usr/sbin/bpslave is started
- BPslave contacts the master
- The slave begins accepting commands from the master

Beoboot Final Stage

Concept: Configure for specific cluster

Details:

- Master sets time of day
- Master mounts file systems
- Master starts any application or services

Compute nodes with Scyld Beowulf

Base system model is diskless

Only 10-50MB of file system data

Minimal file system

- Most space taken by /lib/* libraries
- Most directory entries in /dev/*
- /etc/ is mostly empty
  - /etc/passwd and /etc/group are not needed!
  - /etc/mtab exists only so that 'df' works.
  - Name services (hostname, password) are usually bypassed.
- No executables are required, not even /bin.

Recommended but optional local disks

- Used for databases and additional caching
- Optionally mounted and checked on startup

Various network/cluster file systems are available.

Developing for Beowulf

Three levels

- Explicit job creation
- Writing your own MPI or PVM applications
- Using BProc calls

Simple Cluster Use

Explicit Job Creation

- Easiest Approach to using a cluster
- Just run jobs on a remote node with bpsh

Parametric Execution

- Run the same job on multiple data sets
- Easier with a simple queue system
- BBQ: Beowulf Batch Queue

Compute Farm

- Accept jobs from multiple sources
- Usually used at large sites
- PBS, SGE, LSF and Condor are common systems

Application Server Cluster

Compute nodes used as server nodes
One network connection to master
Other network connections to Internet

Application Server Security

Highly Secure Server Nodes

- No network services to exploit
- No OS password information
- No local executables

Applications "locked" to not migrate from node

Example Script

Script Run on master at start

Uses standard Linux commands and concepts

while true; do

- bpsh $NODE appserver
- logger -t appserver Exited with status $?

done

Using BProc Calls

Enhance existing applications with BProc moves

See ’modprobe’ for a great example

- Reads dependency file from the master
- Reads kernel symbols from the slave
- Reads driver module from the master
- Loads module into slave kernel

Basic call is bproc_move()

- Remote fork semantics
- Takes a numeric destination node ID.
- Available node ID may be found from the NPR or beomap library

Development Environment

- We use community-standard programming interfaces
  - MPI
  - PVM
- BeoMPI allows a binary to dynamically link
  - Ethernet MPICH
  - Myrinet GMPI
  - Dolphin SCI
- An extensible node scheduler, NPR, is integrated
- The development cycle is compile-run, not compile-copy-run.

Deployment and Support

Integrated systems available from many vendors

Training available from Scyld and HP

Commercial add-ons

- Sistina GFS file system
- Veridan PBSPro scheduler
- Platform LSF load sharing
- SteelEye Lifekeeper fail-over

Commercial development tools

- Etnus TotalView debugger (May 2002 and later)
- Veridan PBSPro
- MPI/Pro
- Intel Fortran and C++

Library support for commercial compilers

Deployment and Support

Integrated systems from HP and other vendors

- HP has world-wide availability and support

Training available

- Scyld (on-site)
- HP (world-wide)
- Northrup Grumman for U.S. (GSA and SEWP)