Scalable Performance Clustering
State of the Art
and the future

Donald Becker
Scyld Software
A Penguin Computing Company

becker@scyld.com

Presented with MagicPoint

Definitions

Cluster:

A widely used term meaning

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

Cluster Types:

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

What is Beowulf?

Beowulf is

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

What is Beowulf?

Scalable Performance Clusters

- Improving performance proportionally with added machines

Commodity hardware

- Mass-market, stand-alone compute nodes

Private system network

- Nodes dedicated to computation
- Predictable, efficient and a simple security model

Open source software (Linux) infrastructure

- Core software is verifiable

Why clusters?

Price for Performance

- Obvious initial reason
- 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

Commodity platforms

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

Advantages of Commodity Systems

Commodity CPUs

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

Common environment software

- HPC software traditionally "utilitarian"
- Software differences a barrier to understanding and use

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
- Innovative new generation cluster software
- "Scyld" was the father of Beowulf

Cluster Software

What is important in a cluster software system?

Well, what are the problems?

- Complexity
  - System management model
- Installation
  - First use learning curve
- Applications to use the system
  - Tool availability
- Maintenance
  - Maturity and continuity

What is the goal?

Understanding the goal helps show the path

Uniform virtual environment
Single System Image

Single System Illusion
Creating the illusion of a single standard machine
- No performance impact allowed
- Changed+simpler is not necessarily simpler

Previous Solutions

How have these problems been addressed in the past?

Classic Beowulf

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

New-generation Solution

How the world gotten better?

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
- Eliminates software version skew
- 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

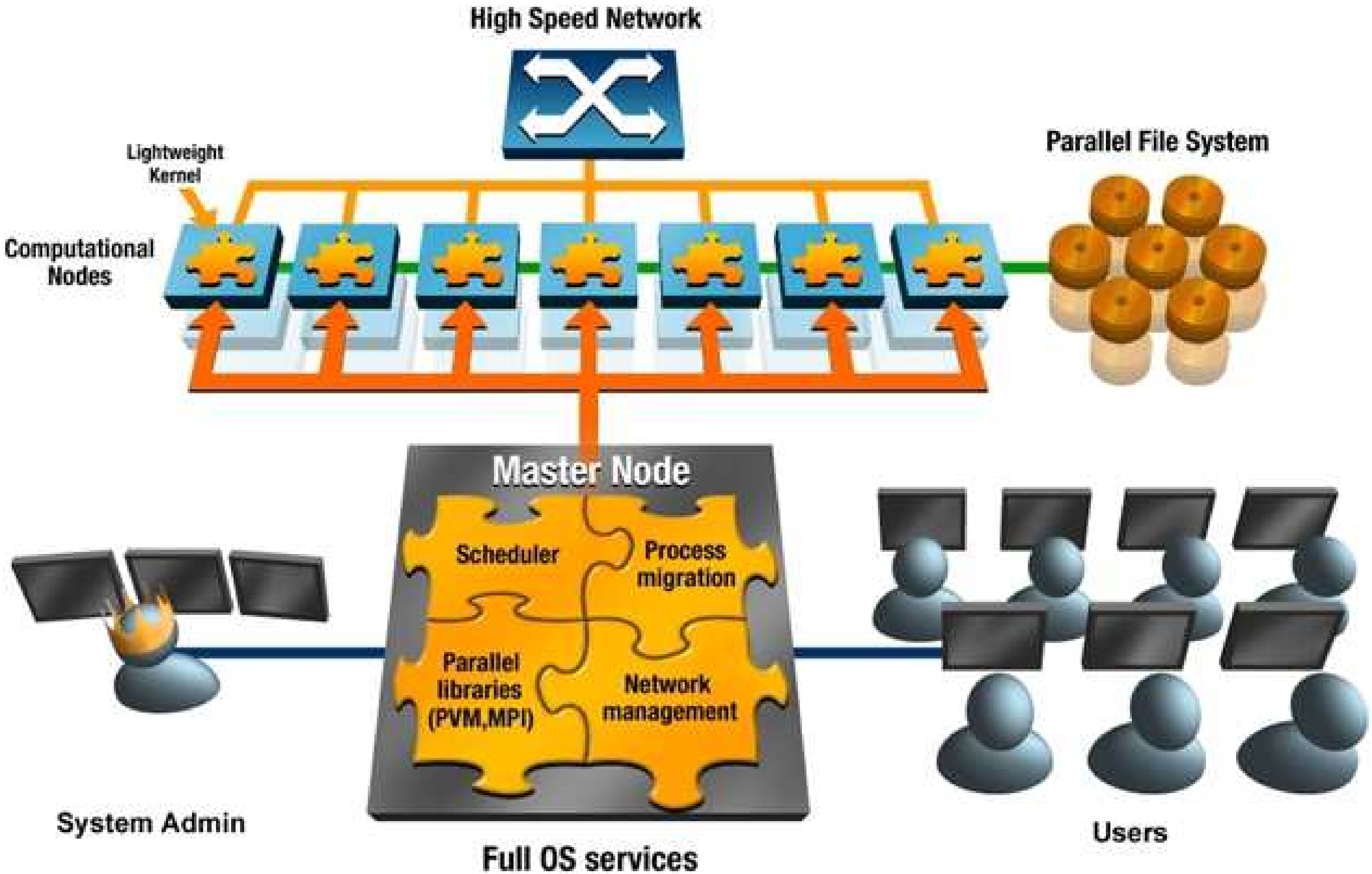
System Model

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

Master
- Full operating system installation
- All standard tools and utilities available unchanged
- Supports user login
- Provides OS, drivers, libraries and applications

Slaves
- Tuned kernel
- No required file system
- No user logins or system services
- No required executables!

Master-Controlled Cluster System Model



Why this System Model?

Combines the advantages of

- A standard user environment
- Ability to run unchanged applications
- Specialized compute systems

Scyld Beowulf Single System Image

Single Installation

Single point upgrade

- Kernel, drivers, system libraries
- User applications, user libraries

No version skew

Zero-installation scaling

- New nodes take seconds to provision
- Full performance on compute nodes

File system semantics selected

- At system integration, or
- By administrator

Unified process space

Operational Details

Nodes are added dynamically

- PXE or Scyld Beoboot booting
- Provisioning takes as little as one second
- Next started job may use new node

A heartbeat is used to detect missing and failed systems

- Immediately removed from scheduling
- Eventually running processes reported as crashed

Detection of lost system connection

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

Subsystems Overview

Booting and Provisioning Clusters

Unified Process Space

Beowulf Name Services

Scheduling

Booting and Provisioning

Functionality not needed with a grid
Tightly tied with the architecture and management
Some systems mistakenly assume it "just happens"
An opportunity and challenge

Booting Clusters

Booting has long been a hot topic

- Various boot media
- Disk-based and Disk-less models
- Safe system software updates problematic
- Multiple reporting points for boot problems

BeoBoot: Scyld Booting and Provisioning

Boot requirements

- Reliable network boot
- Dynamic new node addition
- All run-time components from a controlling system
- Configuration from a central point

Solution

- Unchanging boot code
  - Scyld developed BeoBoot Stage 1
  - Now ubiquitous PXE network boot
- Flow controlled boot server
- Kernel and minimal system from a boot server
- Configuration and provisioning from a master

Beoboot Stage 2: Initial Provisioning

Beoboot has a minimal initial system

- Identifies network devices
- Loads network device driver
- Contacts server for identity information
- Connects to master for configuration

The magic is...

This Space Intentionally Blank

Beoboot Final Stage

Concept: Configure for specific need

For a compute node:

- 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 administrative"

Only 10-50MB of required cached data

Default environment supports most applications

- Visible /lib/* libraries
- /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 /bin or other

Recommended but optional local disks

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

Unified Process Space

Problems:

- Starting jobs on a dynamic cluster
- Monitoring and controlling running processes
- Allowing interactive and scheduler-based usage

Opportunity:

- Clusters jobs are issued from designated masters
- That master has exactly the required environment
- We already have a POSIX job control model
- Tight communication

Solution: A Beowulf Cluster Process Space

Create a cluster-wide Unified Process Space

- Control processes with a local process table entry
- Forward signals and exit status

Precise process creation through migration

- Remote fork or execute to create processes
- Implement using checkpoint/restart migration
- Magic trick: make it fast and efficient

Result

- 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

Performance Characteristics

Start-up

- Under 10 msec. to complete a remote job!
- 10X faster than rsh, 20-30X ssh

No run time performance impact

- System calls and paging are local
- Process status update to master is compact and low-rate
- Only fork(), signals and exit() require a round-trip interaction
- Compare to transparent 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

Improvements

- Better dynamic caching of objects
- Caching selection of RAM (default) or local disk
- Pluggable transport selection e.g. TCP or Myrinet
- Detach process and re-master node

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

Benefits of Cluster Nameservices

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

Cluster Name Service Opportunities

Why can we do a better job?

Clusters have a single set of users
User credentials available at job initiation point
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.

User Name lookups

Names are reported as password table entry ’pwent’

Processes are moved with their user information

BeoNSS reports only the current user and root

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

Other name services

Netgroups to automate file server export security

Services and Protocols databases

- All common, fixed values
- Frequency of use analysis to select and sort entries

Scheduling on Grids vs. Clusters

Similar words and concepts are used
Opportunities and thus architecture differ
Clusters support interactive and administrative use

Definitions

Scheduling: A combination of concepts about running jobs

Queuing: Delaying jobs until resources are available
Backfill: Reordering queue for better utilization
Mapping: Assigning processes of a job to nodes
Environment Configuration: Making files, etc. available
Job Initiation: Creating processes on specified nodes
Job Control: Stopping, resuming and killing processes
Reporting: Tracking resource usage and exit status
Environment Clean-up: Undoing configuration

Scyld Beowulf Scheduling Support

Queuing: BBQ, or external scheduler
Backfill: Only with external scheduler
Mapping: Beomap, NPR, or external scheduler
Environment Configuration: Ad hoc, responsibility of job
Job Initiation: BProc
Job Control: BProc
Reporting: BProc
Environment Clean-up: Ad hoc

Scyld Scheduling

Scyld supports external schedulers,

Differences between Scyld and External Schedulers

Scyld programs call library functions for a map
Extensible by dynamic loading libraries into the application

External Schedulers provide a daemon that schedules jobs
Extensible by loading dynamic libraries into the daemon

Scyld Scheduler Interface

Scyld provides centralized scheduler support

Use Beostat library

- node capability: processor count, speed, memory
- status: load average, free memory

Use BProc library for node state

- Node state is up
- Permission for user execution

Option to force scheduler-only job submissions

- Set node group ownership to scheduler
- Set execute permission only for group

BeoMap, the Scyld Mapping System

Beomap is a layered job mapping system

Programs call beomap functions

Scripts call ’beomap’ program

- Thin wrapper for mapping function
- NODE=beomap --no-local --np 1

Mapping interacts with node status

- Node state -- only use ’up’ nodes
- Node information -- need free memory

BeoMap Implementation Layer

The BeoMap system allows "pluggable" schedulers

- Looks for system- or user-provided dynamic library
- Library function is passed a key (program name)

Default scheduler is good for most uses

- Looks for least-loaded nodes
- Prefers grouping processes on SMP nodes
- Sorts node list by node number

Extended schedulers

- Have access to program name, BeoStat library and BProc state

BeoMap

BeoMap: a better approach

- Other schedulers are daemon-based
- Loading dynamic libraries is more efficient and flexible
- Users and administrators may install customized rules
- Complex network topologies may be handled

Why is this possible in Scyld?

- Kernel-enforced node ownership mechanism
- Invalid mappings simply fail.

Daemon-based schedulers must be closed systems

- May not execute arbitrary user code
- Must use only their internal statistics and job monitoring

Using Scyld Beowulf Cluster

Application Server Cluster

Compute nodes used as server nodes

Inverts traditional cluster network

- Server nodes connect to master and Internet
- Master is firewalled by server nodes

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 *NIX process concepts

```
while true; do
  NODE=`beomap --no-local --np 1`
  bpsh $NODE appserver
  logger -t appserver Exited with status $?
done
```

Using Beowulf Process Space (BProc) Calls

Cluster applications can be very simple.

Basic call is bproc_move(), bproc_rfork()

- Remote move or fork semantics
- Takes a numeric destination node ID.
- Available node ID may be found from the NPR or beomap library
- Resulting processes controlled with *NIX interface

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

Pragmatic Issues

Scyld is a complete supported OS distribution

- Ships as installation/upgrade CDs
- Provides isolation from unexpected "upgrade" changes
- Allows delivering a real cluster OS

Automated installation part of consistent deploys

- Avoid system changes with re-install
- Don’t confuse installation time with learning!

Implications of Single System Image

Single System Image and ad hoc installations are fundamentally at odds

- One kernel over cluster
  - Consider a Filesystem or NIC driver update
- One set of utilities over the cluster

Node specialization not a conflict with this principle

Selecting an optimized kernel is not automatic

Per-machine library and kernel optimizations problematic

- Breaks singles point, single file updates
- Breaks application portability and repeatability

Node Specialization

Specialization allowed by

- Specific machine (MAC address)
- Position in cluster (node number)
- Hardware resources

Heterogeneity support

- Range of support hardware is a common question
- Instruction set must be the same
  - Cannot run the application otherwise
- Installed hardware detection is automatic
  - Drivers installed based on e.g. PCI ID

Operation Issues

Dynamically, automatically scalable

New nodes assigned a permanent node number

- Based on MAC address
- Manual intervention to renumber

New nodes take only seconds to provision

- 750 msec for base system
- Disk detection and file system mounts extra

Hardening System Tools

All system tools should be "hard" and single layered

Don’t rely on interpreters: "Perl v5.6.1.33 only"

- Reserve interpreters for end site use
- Provide language bindings for system interfaces

Single layer implementation

- Human-oriented text configuration files
- Trace problems back to the original configuration
- Generated configuration files are a potential disaster

Libraries, shell, command line and GUI interfaces

- GUIs

Provide an efficient monitoring interface

- Example: Keep vital state mappable shared memory

Future

Even better boot and failure analysis system

- PXE-based CPU and NIC detection
- Complete boot state (failure) reporting
- Environment and kernel fault reporting

Multiple master architectures

- Different structures are possible
- Automatic detection and configuration needed

Integrated process mirroring

- Extension to existing migration
- Opportunity with InfiniBand and other RDMA
- Client pull may increases scalability
- Reliability trade-off

Complete virtual environment creation and mirroring

- Too inefficient today
- Extend process migration to environment migration

Deployment and Support

And now the commercial message

Training available on-site or scheduled
Northrop Grumman hosts training in D.C. area

Scyld Beowulf is available on GSA and SEWP
Integrated clusters, integration services, professional services

Penguin Computing provides standard clusters
Most common commercial cluster deployments:

- AMD Operton w/ gigabit Ethernet or Infiniband
- Intel Xeon on racks or blades