How to run a Linux Cluster

Thomas Neff

t.neff@gsi.de

GSI Darmstadt

Linux Clusters

- GSI Theory Cluster, 28 PCs
  (300MHz PII-1.8GHz P4, 256MB-1GB RAM)
- GSI Batch Farm, 70 nodes
  (Dual Xeon 2.0GHz, 2GB)
- Livermore, 1152 nodes
  (Dual Xeon 2.4GHz, 4GB)
- (Sandia ’Red Storm’, 16000 nodes, AMD Opteron)

Cluster of Workstations

Make a Cluster of Workstations (Desktop PCs)

Goals

- Simplify administration
- Common environment for users
- Use idle time for number crunching

COW - Architecture

- Server machine
  (NIS, NFS shares, Batch, Samba, WWW, mail, ...)
- Client machines
  (provide uniform environment for users)

COW - Installation

Identical software on all clients

- Debian installed locally
- Cloning for new machines
- `/usr/local` is exported from server (Mathematica, StarOffice, IDL, ...)

COW - Cloning

replicator package allows automatic installation of new machine in $\approx$ 15 minutes

- boot floppy for new clients
- naked new box boots with floppy, installation system is NFS mounted
- automatic or manual partitioning of harddisk
- clone system from other client machine (rsync)
- machine specific configuration (hardware, network, ...) by cfengine scripts

COW - Account information

Use NIS to distribute account information

- passwd, shadow, aliases, automounter maps
- accounts have to be created only once
- account data is always in sync (passwords ...)
- still possible to explicitly allow/deny access to certain machines

COW - Home directories

Share home directories

- `/home` is exported for all machines in the Cluster
- automounter allows transparent access on all machines
- permanent staff has home on their machines
- homes of guests are concentrated on a single machine

COW - Login

Recommend **ssh**

- make ssh hostkeys known on all machines
- ssh makes X forwarding painless and secure
- use ssh-agent with public key authentification

COW - Administration

- user administration on server machine (database with account information, location of home directories, email aliases ...)
- keep repository of configuration files
- use **cfengine** to distribute configuration files, start and stop services
- alternatively use **ssh/scp**

COW - cfengine

cfengine is a make like tool for system administration

- define classes for machines, processes, ...
- copy files
- start and stop services
- edit files
- kill processes
- ...

Batch System

Batch System distributes jobs

- users submit a job
- job is started on fastest empty machine
- provides fair scheduling
- can provide different queues (short, long, large, ...)

needed

- common environment (uids, home directories, software)

COW - Batch System

on a COW batch jobs should use idle cpu time, should not affect ordinary user

- run batch jobs with low priority
- jobs should not consume too much memory and io
- runtime of jobs not deterministic

possible problems

- long running jobs can block the queue
- scheduling is difficult for parallel jobs

Batch Systems

Several packages on the market

- **DQS**, Debian package available, no frills, development is dead
- **OpenPBS**, widely used, gui
- **Condor**, active development, job migration, COWs considered in design, gui (included in **globus** package)
- **LSF**, commercial, very expensive, gui

Dedicated Compute Clusters

Dedicated clusters provide CPU time for a larger group of users

- Fast CPUs, lots of RAM, fast network
- Parallel jobs (PVM, MPI)
- No interactive use, use **Batch System** for serial and parallel jobs

Setup for Compute Clusters

Master node

- users access cluster via master node
- has public network address

Client nodes

- no interactive login
- boot from net (PXE, dhcp, nfs)
- system files NFS mounted
- local disk used for `/tmp`

Ready to use (?) Packages

Packages that bundle necessary software to setup and administer a (homogenous) cluster

- OSCAR
- ROCKS
- Scyld Beowulf (commercial)

Mosix

Integrate clustering into the operating system

- Automatic work distribution - for parallel processing or to migrate processes from slower to faster nodes.
- Load balancing - for even work distribution.
- Memory ushering - migrate processes from a node that run out of main memory, to avoid swapping or thrashing.
- High I/O performance - by migrating an intensive I/O process to a file server