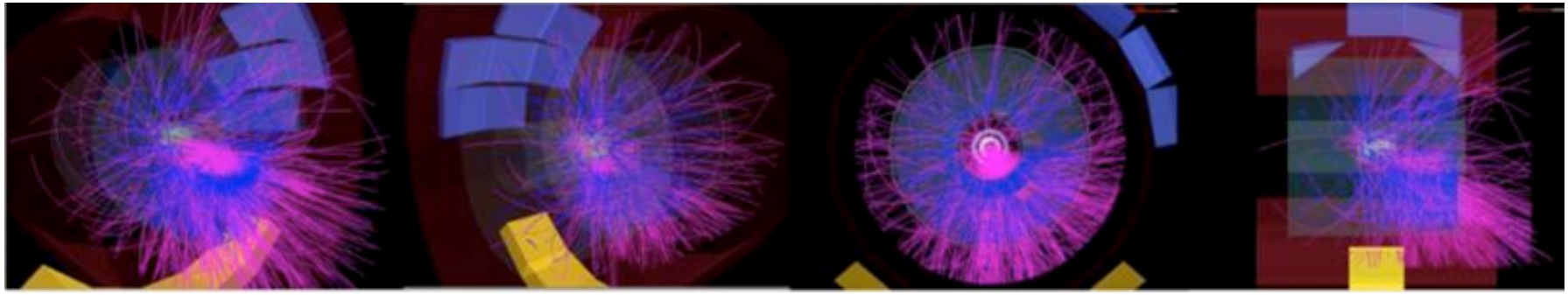


CHOS in Production

Multiple Linux Environments on PDSF at NERSC

Larry Pezzaglia

National Energy Research Scientific Computing Center
Lawrence Berkeley National Laboratory

April 2012

U.S. DEPARTMENT OF ENERGY | Office of Science

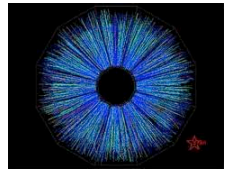
NERSC National Energy Research Scientific Computing Center

BERKELEY LAB Lawrence Berkeley National Laboratory

PDSF at NERSC

- A commodity Linux cluster at NERSC serving HEP and NS projects
- 1GbE and 10GbE interconnect
- In continuous operation since 1996
- ~1500 compute cores on ~200 nodes
- Over 750 TB shared GPFS storage in 17 filesystems
- Over 650 TB of XRootD storage
- Supports SL5 and SL6 environments
- Projects “buy in” to PDSF and the UGE share tree is adjusted accordingly

PDSF Workloads

- PDSF has a broad user base (including non-CERN and non-LHC projects)
- Current projects include ALICE, ATLAS, CUORE, Daya Bay, IceCube, KamLAND, Majorana, and STAR
- Prior projects include BaBaR, CDF, Planck, SNO, and SNFactory

The Challenge

- PDSF must support multiple applications for multiple projects
  - Many are only tested or certified on one Linux distribution release
- Projects have their own communities with different requirements and recommendations
- Simultaneously satisfying the certification constraints of all these projects is challenging
  - We need a way to provide customized environments for each project
  - Our answer is CHOS

What is CHOS?

- CHOS ("CHroot OS") is a software package which provides a mechanism for simultaneously supporting multiple Linux environments on a single Linux system
- Users choose the tree (e.g., SL 6, SL 5, Debian 6) best suited to their application
- CHOS was written by Shane Canon and has been in use on PDSF since 2004.

Other solutions

Dynamic Provisioning

Dynamic Provisioning

- Reboot nodes into an appropriate bare-metal OS prior to each job
- We must maintain multiple bare-metal boot environments
- Jobs requiring different environments can’t share a node

Boot Environments

- Maintaining multiple boot environments is a non-trivial undertaking
- We must keep configuration in accordance with **site policy**
  - Install security patches
  - Maintain configuration and packages for all services (e.g., shared filesystems, batch system, monitoring)
- Nodes will be leaving and joining shared services (including parallel filesystems) with each reboot

Full Virtualization

Full virtualization (e.g., KVM)

- Jobs requiring different environments can now share a node
- We still must maintain multiple boot environments
- If we run one job per core on a 100-node cluster with 24 cores per node, we will have 2400 VMs to manage
  - Each VM mounts and unmounts parallel filesystems
  - Each VM will be joining and leaving shared services with each reboot
  - Shared services (including filesystems) must maintain state for all these VMs

Containers

Containers (e.g., OpenVZ, LXC)

- Jobs requiring different environments can share a node
- We still must maintain multiple boot environments
- 2400 containers are almost as hard to manage as 2400 VMs
- `http://openvz.org/`
- `http://lxc.sf.net/`

chroot

What about a simple “chroot”?

- Minimal overhead
- No support daemons
- Serious usability issues
  - chroot is a privileged operation
- Poor scalability
  - We must maintain access to all shared filesystems within each chroot
  - Maintaining many environments requires many mounts or a symlink farm

The CHOS Solution

The CHOS solution

CHOS provides the simplicity of the chroot solution, but adds important features.

- Users can manually change environments
  - This is as simple as running "`env CHOS=debian5 chos`"
- PAM integration
  - CHOSes a user into the correct environment upon login
- Batch system integration
  - Tested with SGE/UGE and TORQUE+Moab/Maui
- Only one chroot directory is needed

The CHOS solution

- CHOS fulfills most of the use cases for virtualization in HPC with minimal administrative overhead and negligible performance impact
- Users do not interact directly with the “base” OS
- CHOS provides a **seamless user experience**
  - Users manipulate only one file (`$HOME/.chos`), and the desired environment is automatically activated for all interactive and batch work

An example

- Consider an application written and tested for Debian 5 that we want to run on a Scientific Linux 6 HPC system
- We could recompile and test for SL6
- Or, we could run inside a Debian 5 CHOS environment
  - From the application's point of view, we **are** running a Debian 5 userland on an SL6 kernel

User Benefits

- We can support software requiring invasive changes (e.g., swapping stock EL RPMs for customized versions)
- We can support software which only runs on (or is only certified on) Enterprise Linux X.y
- We can provide persistent software stacks
- We can provide reproducible environments for repeatable production runs
  - This allows us to validate prior computations.
  - This is a strong selling point for VMs. CHOS provides similar flexibility with less overhead.

Sysadmin Benefits

- The base OS is sysadmin-friendly
  - It can be updated at will.
  - We can maintain a minimalist design methodology.
  - The PDSF base OS image is less than 300 MB
    - This includes support for GPFS, CVMFS, and monitoring daemons
- No support daemons are required for CHOS

Sysadmin Benefits

- No privileged processes need to run in CHOS
  - No setuid bits are required
  - CHOS is exclusively for user applications
- CHOS environments can live on shared filesystems
- CHOS environments share the same kernel
  - User applications rarely care which kernel version is under the hood
  - Most kernel interfaces have remained been stable enough for our needs
- Small and understandable codebase
  - ~2000 lines (excluding build system)

Requirements

- Must arrange for access to required privileged functionality
  - Setuid binaries are generally unavailable in CHOS
- Must port to new kernels as needed
- Must provide user documentation and training

Under the Hood

- CHOS creates a symbolic link at /proc/chos/link with a contextual target
- The CHOS kernel module maps PIDs to CHOS link targets
- New processes inherit the CHOS target from parents

Under the Hood

- The “chos” utility triggers an environment switch:
  1. The requested environment name is written to /proc/chos/setchos
  2. /proc/chos/link is mapped to the desired environment path
  3. The user is chrooted into /chos/
- /chos/ contains shared directories, and multiple links pointing through /proc/chos/link

/chos/

/chos/ when CHOS is not set:

```
/chos/bin   → /proc/chos/link/bin → /bin/
/chos/etc   → /proc/chos/link/etc → /etc/
/chos/lib   → /proc/chos/link/lib → /lib/
/chos/usr   → /proc/chos/link/usr → /usr/
/chos/proc  → /local/proc/
/chos/tmp   → /local/tmp/
/chos/var   → /local/var/
/chos/dev/      # Common device nodes
/chos/local/    # Mountpoint for the real root tree
```

/chos/

/chos/ when CHOS is sl6:

```
/chos/bin    → /proc/chos/link/bin → /os/sl6/bin/
/chos/etc    → /proc/chos/link/etc → /os/sl6/etc/
/chos/lib    → /proc/chos/link/lib → /os/sl6/lib/
/chos/usr    → /proc/chos/link/usr → /os/sl6/usr/
/chos/proc   → /local/proc/
/chos/tmp    → /local/tmp/
/chos/var    → /local/var/
/chos/dev/      # Common device nodes
/chos/local/    # Mountpoint for the real root tree
```

/chos/

/chos/ when CHOS is deb5:

```
/chos/bin   → /proc/chos/link/bin → /os/deb5/bin/
/chos/etc   → /proc/chos/link/etc → /os/deb5/etc/
/chos/lib   → /proc/chos/link/lib → /os/deb5/lib/
/chos/usr   → /proc/chos/link/usr → /os/deb5/usr/
/chos/proc  → /local/proc/
/chos/tmp   → /local/tmp/
/chos/var   → /local/var/
/chos/dev/      # Common device nodes
/chos/local/    # Mountpoint for the real root tree
```

CHOS on PDSF

- CHOS has been in production on PDSF since 2004. Current environments are:
  - SL 5.3
  - SL 6.2
- In the past, we supported:
  - SL 4.4 (32-bit and 64-bit)
  - SL 3.0.2
  - Fedora Core 2
  - Red Hat 9
  - Red Hat 8
  - Red Hat 7.3
  - Red Hat 7.2
  - Red Hat 6.2

Active Development

- CHOS is an active project distributed under a modified BSD license
- Want to use CHOS on your system or for a project? We can help.
- The code is publicly available on GitHub:
  - Contributions and collaborations are welcome
  - `https://github.com/scanon/chos/`

Future plans

- Build a secure mechanism for users to provide their own CHOS environments
- Provide scripts to help prepare a filesystem hierarchy for use with CHOS
- Simplify the build, deployment, and configuration process
- Explore and possibly adapt techniques used by LXC

A Future Use Case

1. User configures workstation to properly run an application
2. User runs CHOS helper scripts to transform the workstation's file tree into a CHOS environment
3. User transfers this CHOS environment to an HPC system
4. User selects that environment to launch the application in production

A Future Use Case

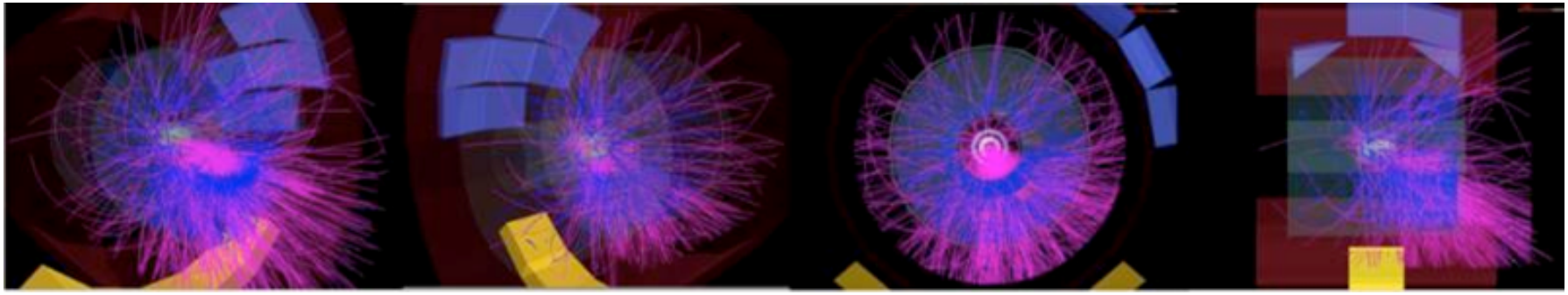
- This would allow user communities to support their own computing environments
- Sysadmins focus on the base OS, core services, filesystems, monitoring, and batch system

Summary

- CHOS enables us to concurrently support multiple Linux environments on a single Linux system
  - Rich computing environments for users
  - Lean, maintainable base OS for sysadmins
  - PAM and batch system integration provide a seamless user experience
- CHOS has been in production on PDSF for over eight years
- CHOS is under active development, with new features on the horizon
- Alternatives to virtualization exist, and CHOS is one of them

Additional Resources

- Original CHOS paper:
  - http://indico.cern.ch/getFile.py/access?contribId=476&sessionId=10&resId=1&materialId=paper&confId=0
- PDSF CHOS User documentation:
  - http://www.nersc.gov/users/computational-systems/pdsf/software-and-tools/chos/



Questions?