U.S. DEPARTMENT OF
ENERGY

UNIVERSITY OF
CALIFORNIA

BERKELEY LAB
LAWRENCE BERKELEY NATIONAL LABORATORY

U.S. DEPARTMENT OF ENERGY

A Community Approach to Compute Node Health

Michael Jennings (*mej@lbl.gov*)

High-Performance Computing Services

Lawrence Berkeley National Laboratory

University of California, Berkeley

HPCAC Stanford Conference — 4 February 2014 — Palo Alto, California, USA

Who We Are: LBNL High-Performance Computing Services

- Highly successful program created to manage PI-owned clusters
- Multi-generation shared institutional system for all researchers
- “Condo Computing” for each generation to encourage sharing
- Project leaders and developers for Warewulf Project
- CalHPC and BRC Programs at UC Berkeley
  - Open to any UCB researcher
  - Keeps UCB clusters in data center and out of closets!
- UC Shared Research Computing Services (ShaRCS)

Our Environment -- LRC Supercluster

- 2200+ nodes in 27 computational clusters/condos
- 900+ jobs daily from 15 different departments
- >40 queues
- 1 TORQUE server for all 2200+ MOMs
- Numerous filesystems
  - 30+ network mounts (Linux NFS, Lustre, NAS)
  - Node-local disk mounts (`/tmp`, `/local`, `/scratch`)
- >30 distinct node types (Vendor, CPU, RAM, swap)

HPCS Clusters at a Glance

Cluster Name	Nodes Up/Total	CPU/Memory Usage	Load (CPUs)	Grid
-ALL-	2245 / 2277	31.97% / 10.16%	7447.92 (22212)	0% 10% 20% 30% 40% 50% 60% 70% 80% 90% 100% Overload
als-xmas	24 / 24	99.62% / 8.94%	47.78 (48)	
alsacc	44 / 44	88.07% / 3.51%	517.19 (592)	
ares	9 / 15	0.00% / 2.74%	0.37 (180)	
asetek0	39 / 39	0.00% / 0.98%	0.27 (780)	
baldur	18 / 18	0.00% / 3.05%	2.70 (72)	
berkelium	14 / 14	0.00% / 1.48%	0.05 (168)	
calhpc	4 / 4	0.00% / 1.47%	0.02 (44)	
catamount	115 / 115	60.33% / 11.03%	1150.95 (1840)	
cortex	15 / 15	10.27% / 12.09%	10.94 (88)	
cumulus	28 / 28	0.00% / 2.79%	0.07 (336)	
explorer	8 / 8	0.00% / 1.11%	0.00 (64)	
fusion	0 / 26	0.00% / 0.00%	0.00 (0)	
hadley	17 / 17	57.65% / 19.44%	87.46 (136)	
hbar	53 / 53	1.96% / 2.80%	6.32 (212)	
henyey	98 / 98	82.36% / 26.34%	727.04 (784)	
jbei	82 / 82	0.02% / 2.16%	5.08 (476)	
jcap	45 / 45	10.69% / 8.46%	347.16 (2880)	
lr1	195 / 195	41.57% / 11.40%	705.87 (1560)	
lr2	173 / 173	6.32% / 4.86%	133.96 (2148)	
lr3	104 / 104	65.79% / 8.12%	1107.22 (1664)	
mako0	272 / 272	0.00% / 1.80%	0.53 (2176)	
mhg	39 / 39	19.87% / 10.88%	404.96 (1096)	
musigny	16 / 16	61.75% / 7.85%	132.82 (208)	
nano	169 / 169	60.94% / 18.34%	351.26 (496)	
natgas	126 / 126	0.00% / 5.87%	1.65 (252)	
ohana	51 / 51	0.12% / 0.81%	0.12 (408)	
phasis	4 / 4	0.00% / 0.77%	0.00 (64)	
riemann	51 / 51	0.71% / 4.33%	4.44 (448)	
scs00	9 / 9	0.00% / 9.94%	4.94 (88)	
vector	9 / 9	0.22% / 1.66%	1.20 (160)	
voltaire	43 / 43	0.00% / 4.18%	0.98 (516)	
vulcan	245 / 245	84.24% / 18.73%	1694.57 (1976)	
yquem	126 / 126	0.00% / 4.96%	0.00 (252)	

The Problem

- New nodes with hyperthreading on
- Filesystems that remount read-only
- Mount failures
- Interfaces that don’t come up
- OOM killer rampage

- Jobs fail.
- Users get angry.

The Solution

- SLURM, TORQUE, PBSPro, and similar resource managers offer hooks for a “node health check” utility
  - Performs tests on nodes periodically as well as before and/or after job execution
  - Jobs will not run if check fails
  - Nodes can be marked down on error
- Other scheduling/RM systems (Univa Grid Engine, IBM Platform LSF) offer similar functionality, but with different naming/invocation schemes

The Problem with the Solution

- No standard -- Most sites use custom, home-grown scripts
  - Often site-specific or overly-customized
  - Usually lacking portability or tweakability
- Unreliable execution, reporting, parent performance
- Need existed for a solution that was both a Framework…
    - Engineered for simplicity and adaptability
    - Easy to read, understand, and apply
  - …and an Implementation
    - Working example of framework
    - Production-worthy product
    - Easy to install, configure, and adapt

Filling the Needs: Warewulf NHC

NHC provides a reliable, flexible, extensible node health check solution: framework + implementation.

- Modular design with dynamic loading and execution
- Shell variables for settings; shell functions/commands for checks
- Single point of administration (configuration); infinite targets
- Feature-rich sample implementation
  - Detached mode
  - Support for multiple resource managers (or none at all)
  - Checks for filesystems, inodes, file contents, hardware configuration, processes, and more – out of the box!
  - Unit tests for the main `nhc` script as well as all checks

Key Features

- 100% native bash framework
- Compatible with RHEL4+
- Single config, infinite targets
- Match config file directives via glob, regex, or pdsh-like range
- Flexible, unrestrictive syntax
- Per-run data cache for speed
- Control via CLI or config
- Run via RM, `cron`, `pdsh`, or all
- SLURM, TORQUE/PBS, SGE/UGE, IBM Platform LSF
- Detached mode for low delay
- Built-in watchdog timer
- Unit tests for framework and every built-in check
- Works with LDAP, NIS, SMB
- 30 checks already built in for hardware, processes, filesystems, jobs, and more
- More checks to come
- Contribute your own checks or ideas for new checks!

NHC Configuration Overview

- Default location: `/etc/nhc/nhc.conf` (configurable at build time)
- General syntax: *`host_mask`* `||` *`stuff`*
  - The host mask can take one of three forms:
    - Glob expression: `*  n*.cluster  io*`
    - Regular expression: `/./  /^n[0-9]+.cluster$/  /^io/`
    - Range expression: `{n00[00-79].a,n01[50-99].a}`
  - The “stuff” can be pretty much anything (`bash`-wise, of course)
- Set variables: *`host_mask`* `|| export` *`var_name`*`=“`*`value`*”
- Run checks: *`host_mask`* `||` *`check_name check_options`*
- Do complex stuff:

```
host_mask || [ `date '+%m%d'` = '0401' ] && reboot
```

Built-In Checks

DMI checks

- check_dmi_data_match [-h *handle*] [-t *type*] [-n | '!'] *string*
- check_dmi_raw_data_match ['!'] *string*

File checks

- check_file_contents *filename* [*match(es)*]

Filesystem checks

- check_fs_mount *mountpoint* [*source*] [*options*]
- check_fs_mount_ro, check_fs_mount_rw
- check_fs_size *filesystem* [*minsize*] [*maxsize*]
- check_fs_used *filesystem* [*maxused*], check_fs_free *filesystem* [*minfree*]
- check_fs_inodes *filesystem* [*min*] [*max*]
- check_fs_iused *filesystem* [*maxused*], check_fs_ifree *filesystem* [*minfree*]

nVidia HealthMon GPU check

- check_nv_healthmon

Built-In Checks (continued)

Hardware checks

- check_hw_cpuinfo *sockets* [*cores*] [*threads*]
- check_hw_eth *device*, check_hw_gm *device*, check_hw_ib *rate* [*device*]
- check_hw_mem *min_kB max_kB*, check_hw_mem_free *min_kB max_kB*
- check_hw_physmem *min_kB max_kB*, check_hw_physmem_free *min_kB max_kB*
- check_hw_swap *min_kB max_kB*, check_hw_swap_free *min_kB max_kB*
- check_hw_mcelog

Process checks

- check_ps_daemon *command* [*owner*] [*args*]
- check_ps_blacklist *command* [[!]*owner*] [*args*]
- check_ps_kswapd *cpu_time discrepancy* [*action(s)*]
- check_ps_service [*option(s)*] *service* 

- check_ps_unauth_users [*action(s)*]
- check_ps_userproc_lineage [*action(s)*]

Detached Mode

Forks `nhc` after parsing `/etc/sysconfig/nhc` and reading `$RESULTFILE`

Background process:

- Runs all checks
- Records results in `$RESULTFILE`
- Marks node on/offline if `$MARK_OFFLINE` is 1

Foreground process:

- Acts on results from previous run

NHC Configuration Example

```
### VARIABLES
     * || export NHC_RM=pbs
     * || export MAX_SYS_UID=499
     * || export PATH="$PATH:/opt/torque/bin"
n*.gpu || export PATH="$PATH:/opt/nv/bin"

### DMI CHECKS
# Make sure we're running the correct BIOS version on all nodes
 * || check_dmi_data_match "BIOS Information: Version: 2.0.1"
# Make sure our RAM is running at the correct bus rate
 * || check_dmi_data_match -t "Memory Device" "*Speed: 1600 MHz"
# Don't allow this bad motherboard back into the cluster!
 * || check_dmi_raw_data_match '!' "/UUID: 0000000-0000-0000-0000/"

### FILE CHECKS
# Assert specific TORQUE settings that are critical to operation
 * || check_file_contents $PBS_SERVER_HOME/mom_priv/config
↪       '/^\$pbsserver master$/' '/^\$spool_as_final_name true$/'
# Validate passwd file
 * || check_file_contents /etc/passwd "root:x:0:0:*" "sshd:*"
```

NHC Configuration (cont'd)

```
### FILESYSTEM CHECKS
# Make sure critical filesystems are present and writeable
 * || check_fs_mount_rw /
 * || check_fs_mount_rw /tmp
 * || check_fs_mount_ro /global/software

# Home directories should be present and NFSv3-mounted
 * || check_fs_mount /home bluearc0:/home nfs '/(^|,)vers=3(,|$)/'

# Enforce filesystem space constraints
 * || check_fs_used / 98%
 * || check_fs_free /tmp 256MB
 * || check_fs_free /local 1%
 * || check_fs_size /tmp 1G

# Check inodes too
 n*.bio || check_fs_inodes /clusterfs/bio 65m
 n*.bio || check_fs_ifree /clusterfs/bio 10k
 n*.bio || check_fs_iused /tmp 99%
```

NHC Configuration (cont'd)

```
### HARDWARE CHECKS
# Check number of physical CPUs, physical cores, virtual cores
*.gen1  || check_hw_cpuinfo 2 8 8
*.gen2  || check_hw_cpuinfo 2 12 12
*.gen3  || check_hw_cpuinfo 2 16 16

# Verify amount of physical RAM, swap, and total memory
{n0[000-099].gen2} || check_hw_physmem 23g 25g
{n0[100-199].gen2} || check_hw_swap 4g 4g
{n0[200-249].gen2} || check_hw_mem 35g 37g

# Free memory minima for all nodes
 * || check_hw_physmem_free 1M
 * || check_hw_swap_free 2G
 * || check_hw_mem_free 2G

# Network devices
  *    || check_hw_eth eth0
 *.c3  || check_hw_ib 40 ib0
 *.c1  || check_hw_gm myri0
```

NHC Configuration (cont'd)

```
### NVIDIA HEALTHMON CHECKS
# Check GPU status on all nodes with GPUs
{n01[64-72].c2,n00[30-39].c3} || check_nv_healthmon

### PROCESS CHECKS
# Check that no sshd processes are owned by normal users
 * || check_ps_blacklist sshd '!root'

# Check that root-owned sshd is running.
 * || check_ps_daemon sshd root

# Check for NUMA node imbalance.
 * || check_ps_kswapd 1800000 100 log syslog

# Warn the admins if any unauthorized users are detected
 * || check_ps_unauth_users syslog

# Also warn for processes not started via pbs_mom
 * || check_ps_userproc_lineage syslog
```

NHC Quick Start Guide

1. Download NHC: http://warewulf.lbl.gov/downloads/releases/
2. Install RPM (or build and install from tarball)
3. Edit configuration file (default: `/etc/nhc/nhc.conf`)
4. Configure launch mechanism:
   - crond – Consider using sample script `nhc.cron`
   - TORQUE – `$node_check_script` & `$node_check_interval`
   - SLURM – `HealthCheckProgram` & `HealthCheckInterval`
   - SGE – Load sensor: `load_sensor` & `load_thresholds`
   - IBM Platform LSF – Run from `cron` (future: load indices?)

Warewulf NHC 5-Minute Quickstart

Download and install the NHC RPM package

```
# cd /etc/yum.repos.d
# wget http://warewulf.lbl.gov/downloads/repo/warewulf-rhel6.repo
# yum install warewulf-nhc
```

Edit NHC configuration file

```
# vi /etc/nhc/nhc.conf
 *  || check_fs_mount_rw /
 *  || check_ps_daemon sshd root
 *  || check_hw_cpuinfo 2 12 24
 *  || check_hw_physmem 1024 1073741824
 *  || check_hw_swap 1 1073741824
 *  || check_hw_mem 1024 1073741824
 *  || check_hw_physmem_free 1
 *  || check_hw_swap_free 1
 *  || check_hw_mem_free 1
```


Warewulf NHC 5-Minute Quickstart

Test NHC execution and config

```
# nhc
# echo $?
# cat /var/log/nhc.log
```

Configure TORQUE to run NHC

```
# vi /var/spool/torque/mom_priv/config
$node_check_script /usr/sbin/nhc
$node_check_interval 1
# /etc/init.d/pbs_mom restart
```

Configure SLURM to run NHC

```
# vi /etc/slurm/slurm.conf
HealthCheckProgram=/usr/sbin/nhc
HealthCheckInterval=300
# /etc/init.d/slurm restart
```


NHC Resources

Previous Talk

- MoabCon 2013: http://go.lbl.gov/nhc-2013-mc

Warewulf Project

- Web Site: http://warewulf.lbl.gov/
- Node Health Check: http://go.lbl.gov/nhc

Mailing Lists:

- warewulf@lbl.gov
- warewulf-devel@lbl.gov