U.S. DEPARTMENT OF
ENERGY

UNIVERSITY OF
CALIFORNIA

BERKELEY LAB
LAWRENCE BERKELEY NATIONAL LABORATORY

U.S. DEPARTMENT OF ENERGY

Lawrence Berkeley National Lab Node Health Check – Update

Michael Jennings (*mej@lbl.gov*)

High-Performance Computing Services

Lawrence Berkeley National Laboratory

University of California, Berkeley

HPC Advisory Council – 24 February 2016 – Stanford University

LBNL NHC Overview

NHC provides a reliable, extensible node health check solution:

- 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 driver: Detached mode, watchdog, multi-context
- 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
- Powerful companion tools (`nhc-wrapper`, `nhc-genconf`)

Key Features

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

Recent Developments

- Project renamed from “Warewulf Node Health Check” to “LBNL Node Health Check” for version 1.4.2 release
- Development moved to Git:
  - GitHub: https://github.com/mej/nhc
  - Bitbucket: https://bitbucket.org/mej0/nhc
  - Both trees are kept in sync, so use whichever you prefer!
- Real-time developer communications
  - Slack: https://mej.slack.com/messages/nhc
  - Gitter: https://gitter.im/mej/nhc
- Mailing Lists/Forums on Google Groups (subscription req’d)
  - https://groups.google.com/a/lbl.gov/forum/#!forum/nhc
  - https://groups.google.com/a/lbl.gov/forum/#!forum/nhc-devel
  - nhc@lbl.gov and/or nhc-devel@lbl.gov (+subscribe)

LBNL NHC 1.4.2 New Features

- Support for negating ***any*** match string anywhere (node hostname match strings in configuration files, any match strings used in check parameters)
- `check_net_ping()`: New check for monitoring of connectivity
- `check_ps_*()`: Process owner parameters now accept match strings, allowing them to take full advantage of wildcards, regular expressions, ranges, and/or external dynamic matching (e.g., UNIX group membership-based matching)
- `check_ps_service()`: Additional options to tell NHC to execute stop/start/restart/cycle and/or user-defined actions synchronously (instead of backgrounding them) so that NHC can verify they worked correctly (or fail if not).
  - `-v` (Verify Sync) will only fail check if action fails
  - `-V` (Verify Check) additionally will make sure action appears to have worked (e.g., killed process is gone, started daemon is running, etc.)
- `check_cmd_dmesg()`: New check to validate/verify or catch/flag certain important content in the output of the `dmesg` command. This check is also a working example to facilitate user wrapping of `check_cmd_output()` to generate custom/specific check failure messages.
- New command-line flag: "`-e` *check*" will override config file, execute just the specified *check*, and return success/failure based on that single check. Useful for testing/debugging new checks. Essentially analogous to having a 1-line config file consisting of "`*` `||` *check*" and nothing else.
- `check_fs_mount()`: Create missing mount points as necessary.

LBNL NHC 1.4.2 Fixes/Enhancements

- Silence warnings about watchdog timer and RM detection that were confusing to users and/or no longer useful
- Refactored match string range checking to be comparative rather than iterative. Large ranges will see a huge speed-up!
- Refactored generation and handling of numerical quantities to avoid excessive rounding/truncation errors. For example, truncation of 1.5TB of RAM to 1TB is a significant error! Now uses 1536GB instead.
- Fixed issue with leading zeros in node range expressions causing numbers to be interpreted as octal.
- Fixed typo in `getent` fallback handling that kept it from working in certain cases.
- Simplified the killing of the watchdog timer to avoid apparent BASH bug causing aborts.
- Fixed list of SLURM node states supported by `node-mark-{on,off}line`
- Fix trimming of whitespace in config files to include tabs

External Match Syntax

- Suggested by Stanford for use with Clustershell
- Supports any number of arbitrary external commands
- Replaces %h with double-quoted match string value and %m with double-quoted match string expression
- Examples:

```
### /etc/sysconfig/nhc
NHC_MCHECK_DELIM=( [0]="%" [1]="@" [2]="<>" [3]="~...~" )
NHC_MCHECK_COMMAND=( \
  [0]="grep -E \"^(\"%m\"):\" /etc/group | fgrep -w %h" \
  [1]="wwsh object print -p name -t node -l cluster %m | fgrep -w %h" \
  [2]="wwsh object print -p name -t node -l groups %m | fgrep -w %h" \
  [3]="wwsh node list \"%m\" | fgrep -w %h" \
)
```

```
### /etc/nhc/nhc.conf
       *         || export NHC_CHECK_ALL=1
       *         || check_ps_service -V -S -u root sshd
!<interactive>   || check_ps_time -l -s -k -u !%admins|opers% -m /sshd/ 1s
   @webfarm@     || check_ps_service -V -r -u root httpd
~*.lr? *.mako0~ || check_file_contents /etc/fstab /lustre@o2ib/
```

NHC Future Additions

- New built-in checks
  - User-contributed checks, such as `csc_nvsmi_healthmon()` currently available in `dev` branch on GitHub/BB, come in regularly and are always welcome!
  - SLURM diagnostics
  - Check wrapper for the `lsof` command
  - Rolling reboots
  - More Moab/TORQUE diagnostics
  - Use process lineage to determine authorized user list for SLURM (and TORQUE?)
  - Add minimum CPU time feature to rogue/runaway process checks
  - New `check_ps_user_procs()` check with CLI options and more flexibility
- Data cache timeouts to refresh data for long-running NHC processes
- NHC daemon process to provide file-based access to host data

Configuration Syntax

- Default location: `/etc/nhc/`*`name`*`.conf` (configurable at build time)
- General syntax: *`host_mask`* `||` *`stuff`*
  - *`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}`
    - New “external” match expression: `<io-nodes>`
  - *`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`*
- Any 1-line shell command (just make sure it returns 0 on success!):
  `host_mask` `|| selinuxenabled && die 1 SELinux`
- Entire file is parsed, and matching checks saved, before ANY are run!

Built-In Checks

Command Checks

- `check_cmd_dmesg [-t timeout] [-r rc] [-e cmd] [-M msg] [-m mstr]`
- `check_cmd_output [-t timeout] [-r rc] [-e cmd] [-M msg] [-m mstr]`
- `check_cmd_status [-t timeout] [-r rc] [-e cmd]`

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)]`
- `check_file_stat [-D dev] [-G grpname] [-M modemask] [-N ctime_newer] [-O ctime_older] [-T minor] [-U userid] [-d dev] [-g gid] [-m mode] [-n mtime_newer] [-o mtime_older] [-t major] [-u uid] filename(s)`
- `check_file_test [-G] [-L] [-N] [-O] [-S] [-a] [-b] [-c] [-d] [-e] [-f] [-g] [-h] [-k] [-p] [-r] [-s] [-t] [-u] [-w] [-x] arg(s)`

Built-In Checks (continued)

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*]

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*] [*fudge*]
- check_hw_mem_free [*min_kB*] [*max_kB*]
- check_hw_physmem [*min_kB*] [*max_kB*] [*fudge*]
- check_hw_physmem_free [*min_kB*] [*max_kB*]
- check_hw_swap [*min_kB*] [*max_kB*] [*fudge*]
- check_hw_swap_free [*min_kB*] [*max_kB*]
- check_hw_mcelog

Built-In Checks (continued)

Moab/TORQUE Checks

- `check_moab_sched [-t timeout] [-m [!]match […]] [-v version] [-a alert]`
- `check_moab_rm [-t timeout] [-m [!]match […]]`
- `check_moab_torque [-t timeout] [-m [!]match […]]`

Network Checks

- `check_net_ping [-I interface] [-i interval] [-s size] [-t ttl] [-c count] [-w deadline] [-W timeout] target(s)`
- `check_net_socket [-0] [-a] [-!] [-n name] [-p [!]proto] [-t [!]type] [-s [!]state] [-u [!]user] [-d [!]daemon] [-l [!]locaddr[:locport]] [-r [!]rmtaddr[:rmtport]] [-e action | -E action]`

nVidia HealthMon GPU Check

- `check_nv_healthmon`

Built-In Checks (continued)

Process Checks

- check_ps_daemon *command* [*owner*] [*args*]
- check_ps_blacklist *command* [[!]*owner*] [*args*]
- check_ps_kswapd *cpu_time discrepancy* [*action(s)*]
- check_ps_loadavg [*1m*] [*5m*] [*15m*]
- check_ps_service [-O] [-f] [-V|-v] [-S|-r|-c|-s|-k] [-u [!]*user*] [-d *daemon* | -m *match*] [-e *action* | -E *found_action*] *service*
- check_ps_unauth_users [*action(s)*]
- check_ps_userproc_lineage [*action(s)*]

Resource Consumption Checks

- check_ps_cpu [-O] [-f] [-a] [-l] [-s] [-k] [-K] [-r *renice*] [-u [!]*user*] [-m [!]*match*] [-e *action*] *threshold*
- check_ps_mem [*rc_option(s)*] *threshold*
- check_ps_physmem [*rc_option(s)*] *threshold*
- check_ps_time [*rc_option(s)*] *threshold*

NHC Quick Start Guide

1. Download NHC: https://github.com/mej/nhc/releases/tag/1.4.2
2. Install RPM (or build and install from source)
3. Edit configuration file (default: `/etc/nhc/`*`name`*`.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?)

5-Minute Quick Start Example

Download and install the NHC source

```
# git clone https://github.com/mej/nhc.git
# cd nhc
# ./autogen.sh && make dist && rpmbuild -ta lbnl-nhc*.tar.gz
# rpm -Uvh lbnl-nhc*.noarch.rpm
```

Generate and edit NHC configuration file

```
# nhc-genconf
# cat /etc/nhc/nhc.conf.auto
…
# vi /etc/nhc/nhc.conf
…merge auto-generated checks into sample config…
```

Test NHC execution and configuration

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

5-Minute Quick Start Example

Configure `cron` to run NHC…

```
# cat >> /etc/crontab
*/5 * * * * root /usr/sbin/nhc-wrapper -M someuser@someplace.com
↪ -X 6h -- -n nhc-zathras -a -t 120
```

…and/or configure TORQUE to run NHC…

```
# vi /var/spool/torque/mom_priv/config
$node_check_script /usr/sbin/nhc
$node_check_interval 7
# /etc/init.d/pbs_mom restart
```

…and/or configure SLURM to run NHC

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

NHC Configuration Example

```
### NHC Primary Configuration File (/etc/nhc/nhc.conf)
### Variable settings
       * || export MOABHOMEDIR="/opt/moab"
       * || export PATH="$MOABHOMEDIR/bin:$PATH"
### Filesystem checks
       * || check_fs_mount_rw /
       * || check_fs_mount_rw /tmp
       * || check_fs_mount_rw /home cluster-nas:/export/home nfs
       * || check_fs_mount_ro /soft cluster-nas:/export/soft nfs
       * || check_fs_mount_rw /local /dev/sda2 ext4
       * || check_fs_mount_rw /dev/pts '/(none|devpts)/' devpts
### Process checks
       * || check_ps_service -u root -S sshd
       * || check_ps_service -u root -r crond
   node* || check_ps_unauth_users log syslog
   node* || check_ps_userproc_lineage log syslog
   node* || check_ps_loadavg 40
### Hardware checks
       * || check_hw_physmem_free 1
   node* || check_hw_cpuinfo 2 20 20
   node* || check_hw_physmem 64GB 64GB 3%
   node* || check_hw_swap 4GB 4GB 3%
```

NHC Configuration - Hourly Context

```
### NHC Hourly Out-of-Band Configuration (/etc/nhc/nhc-hourly.conf)

### Variable settings
       * || export NHC_CHECK_ALL=1
       * || export MOABHOMEDIR="/opt/moab"
       * || export PATH="$MOABHOMEDIR/bin:$PATH"

### Filesystem checks
       * || check_fs_mount_rw /
       * || check_fs_mount_rw /tmp
       * || check_fs_mount_rw /home cluster-nas:/export/home nfs
       * || check_fs_mount_ro /soft cluster-nas:/export/soft nfs
       * || check_fs_mount_rw /local /dev/sda2 ext4
       * || check_fs_mount_rw /dev/pts '/(none|devpts)/' devpts
 master* || check_fs_free     /             10%
 master* || check_fs_ifree    /             5k
 master* || check_fs_free     /home         5%
 master* || check_fs_ifree    /home         50k
 master* || check_fs_free     /soft         2%
 master* || check_fs_ifree    /soft         20k
```

NHC Hourly Context (cont'd)

```
### Process checks
       * || check_ps_service -u root -S sshd
       * || check_ps_service -d rpc.statd -r nfslock
       * || check_ps_service -u root -r crond
       * || check_ps_service -r auditd
       * || check_ps_service -d rsyslogd -r syslog
   node* || check_ps_service -u root -r wulfd
   node* || check_ps_unauth_users log syslog
   node* || check_ps_userproc_lineage log syslog
   node* || check_ps_loadavg 40
 master* || check_ps_service -u nobody -r dnsmasq
 master* || check_ps_cpu -s -m '*bad*' -e 'service bad restart' 99%
 master* || check_ps_service -u root -r dsmcad
 master* || check_ps_service -c mysqld
 master* || check_ps_loadavg 24
### Hardware checks
       * || check_hw_physmem_free 1
   node* || check_hw_cpuinfo 2 20 20
   node* || check_hw_physmem 64GB 64GB 3%
   node* || check_hw_swap 4GB 4GB 3%
 master* || check_hw_cpuinfo 2 12 12
 master* || check_hw_physmem 32GB 32GB 3%
 master* || check_hw_swap 18GB 18GB 3%
```

NHC Hourly Context (cont'd)

```
### File checks
       * || check_file_test -r -w -x -d /tmp /var/tmp /scratch
       * || check_file_test -r -s /etc/passwd /etc/group
       * || check_file_stat -m 0666 -u 0 -g 0 -t 1 -T 3 /dev/null
 master* || check_file_stat -n 7200 /var/log/httpd/access.log

### Network checks
   node* || check_net_socket -! -p tcp -l '*:80' -s LISTEN -d httpd
   node* || check_net_ping -i 0.2 -w 2 -c 5 master defgw
 master* || check_net_socket -p tcp -l '*:80' -s LISTEN -d httpd
 master* || check_net_ping -i 0.2 -w 2 -c 5 defgw node0000 node0500

### TORQUE/Moab checks
       * || check_ps_service -u root -r trqauthd
   node* || check_ps_service -u root -r pbs_mom
 master* || check_ps_service -f -u root -r pbs_server
 master* || check_moab_sched -t 10 -v 7.2.3 -m '!/PAUSED/'
 master* || check_moab_rm -t 10
 master* || check_moab_torque -t 10
```

Resource Consumption - CPU

`check_ps_cpu [-0] [-f] [-a] [-l] [-s] [-k] [-K] [-r` *`renice`*`]`
`[-u [!]`*`user`*`] [-m [!]`*`match`*`] [-e` *`action`*`]` *`threshold`*

- Looks at processes' current percentage of CPU usage (`pcpu`)
- Matching is done against each process's `$0` (use `-f` for full match) based on *`match`* (standard match string, possibly negated). Default matches any process.
- Can be non-fatal (`-0`) and may have optional positive/negative owner check (`-u` *`user`* or `-u !`*`user`*). Use `-a` to check all processes.
- Available actions for matching processes:
  - Log to NHC log (`-l`) or syslog (`-s`)
  - Kill process (`-k`) and/or its parent (`-K`)
  - Renice process (`-r`) to specified *`renice`* value
  - Any arbitrary command *`action`* (`-e`)
- Specify numeric *`threshold`* with or without the trailing `%` sign.

Resource Consumption - Memory

`check_ps_mem [-0] [-f] [-a] [-l] [-s] [-k] [-K] [-r` *`renice`*`]`
`[-u [!]`*`user`*`] [-m [!]`*`match`*`] [-e` *`action`*`]` *`threshold`*

- Looks at processes' total memory (RAM+swap) size (`vsz`)
- Matching is done against each process's `$0` (use `-f` for full match) based on *`match`* (standard match string, possibly negated). Default matches any process.
- Can be non-fatal (`-0`) and may have optional positive/negative owner check (`-u` *`user`* or `-u !`*`user`*). Use `-a` to check all processes.
- Available actions for matching processes:
  - Log to NHC log (`-l`) or syslog (`-s`)
  - Kill process (`-k`) and/or its parent (`-K`)
  - Renice process (`-r`) to specified *`renice`* value
  - Any arbitrary command *`action`* (`-e`)
- Specify numeric *`threshold`* with optional byte suffix.

Resource Consumption - RAM

`check_ps_physmem [-0] [-f] [-a] [-l] [-s] [-k] [-K] [-r` *`renice`*`] [-u [!]`*`user`*`] [-m [!]`*`match`*`] [-e `*`action`*`] `*`threshold`*

- Looks at processes' physical RAM size/percentage (`rss` or `pmem`)
- Matching is done against each process's `$0` (use `-f` for full match) based on *`match`* (standard match string, possibly negated). Default matches any process.
- Can be non-fatal (`-0`) and may have optional positive/negative owner check (`-u` *`user`* or `-u !`*`user`*). Use `-a` to check all processes.
- Available actions for matching processes:
  - Log to NHC log (`-l`) or syslog (`-s`)
  - Kill process (`-k`) and/or its parent (`-K`)
  - Renice process (`-r`) to specified *`renice`* value
  - Any arbitrary command *`action`* (`-e`)
- Specify numeric *`threshold`* with optional byte suffix or trailing `%` sign.

Resource Consumption - CPU Time

`check_ps_time [-O] [-f] [-a] [-l] [-s] [-k] [-K] [-r` *`renice`*`]`
`[-u [!]`*`user`*`] [-m [!]`*`match`*`] [-e` *`action`*`]` *`threshold`*

- Looks at processes' lifetime CPU consumption (`bsdtime`)
- Matching is done against each process's `$0` (use `-f` for full match) based on *`match`* (standard match string, possibly negated). Default matches any process.
- Can be non-fatal (`-O`) and may have optional positive/negative owner check (`-u` *`user`* or `-u !`*`user`*). Use `-a` to check all processes.
- Available actions for matching processes:
  - Log to NHC log (`-l`) or syslog (`-s`)
  - Kill process (`-k`) and/or its parent (`-K`)
  - Renice process (`-r`) to specified *`renice`* value
  - Any arbitrary command *`action`* (`-e`)
- Specify numeric *`threshold`* for seconds or use `m`/`s` suffixes (*`XX`*`m`*`YY`*`s`).

System Services

`check_ps_service [-O] [-f] [-V|-v] [-S|-r|-c|-s|-k] [-u [!]`*`user`*`]`
`[-d `*`daemon`*`|-m `*`match`*`] [-e `*`action`*`|-E `*`found_action`*`] `*`service`*

- Like `check_ps_daemon` and `check_ps_blacklist`, but service-based
- Flexible service manipulation via `/sbin/service`; services can be `started` (`-S`), `restarted` (`-r`), or `stopped` (`-s`); also cycled (`-c`) (`stop`/`sleep`/`start`) or killed (`-k`) (direct SIGKILL).
- Arbitrary actions are also possible via `-e` (not found) or `-E` (found)
- Use `-v` (Verify Sync) to perform action in foreground and pass on success. Use `-V` (Verify Check) to also verify expected result of action (e.g., killed process is gone)
- Matching is done against each process's `$0` (use `-f` for full match) based on *`match`* (standard NHC match string). If not specified, defaults to "*`*daemon`*" (or *`*service`*).
- Use of the `-E` (found_action), `-s` (stop) or `-k` (kill) options implies negative logic; i.e., finding a match is a failure rather than a success.
- Can be non-fatal (`-O`) and may have optional positive/negative owner check (`-u` *`user`* or `-u` `!`*`user`*)

Filesystem Status

`check_fs_mount [-0] [-r] [-s [!]`*`src`*`] [-t [!]`*`type`*`] [-o [!]`*`opts`*`]`
`[ -e `*`action`*` | -E `*`action`*` ] [-O `*`mountopts`*`] -f `*`mountpoint`*` […]`

- Checks for given *`mountpoint(s)`* in list of mounted filesystems; fails if not found unless `-0` (non-fatal) option given.
- Matching filesystems may be checked for source/device *`src`*, type *`type`*, and/or mount options *`opts`* (all standard NHC match strings which may be negated).
- Can attempt to (re-)mount missing filesystems, possibly with specified mount options, by executing: `mount -o `*`mountopts mountpoint`*
- Arbitrary actions are also possible via `-e` (if not found) or `-E` (if found)
- Missing mountpoint directories are created as needed
- Any number of filesystems may be checked at the same time.
- Shortcuts to check for rw/ro: `check_fs_mount_rw`, `check_fs_mount_ro`

File Metadata

`check_file_stat` *`option(s) filename(s)`*

- Makes assertions regarding file metadata accessible via `stat` command. Results are cached for efficiency.
- Supports simple matches to device number (`-D` or `-d`), group name (`-G`) or gid (`-g`), mode masks (`-M`) or exact mode (`-m`), device major (`-t`) and minor (`-T`) numbers, owner userid (`-U`) or uid (`-u`). Also supports comparisons against ctime (newer `-N`/older `-O`) and mtime (newer `-n`/older `-o`).

`check_file_test` *`option(s) filename(s)`*

- Uses built-in `test` command to assert file properties (avoids `stat`).
- Supports most available options (`-G -L -N -O -S -a -b -c -d -e -f -g -h -k -p -r -s -u -w -x`).
- Evaluates `test` *`operator argument`* for each combination. The check fails immediately if any returns false.

File Contents & Load Average

`check_file_contents` *`filename match(es)…`*

- Can now make both positive AND negative assertions about file data.
- As usual, matches are globs or regular expressions. Prefix with `!` to negate.

`check_ps_loadavg [`*`limit_1m`*`] [`*`limit_5m`*`] [`*`limit_15m`*`]`

- Places a cap on the 1-, 5-, and/or 15-minute load average.
- Check fails if any load average meets or exceeds its limit (if specified).
- Leave parameter empty (i.e., specify ‘’) for no limit.
- Originally written in front of a live studio audience at MoabCon 2013!

Command Checks

`check_cmd_output [-t` *`timeout`*`] [-r` *`returncode`*`]`
`[-M` *`msg`* `[…]] [-m [!]`*`match`* `[…]] [-e` *`command`*`]` *`command`*

- Executes *`command`* and checks output against *`match`*(es) and/or return code against *`returncode`*.
- Matching is done against each line of output for each *`match`* (standard match string, possibly negated) supplied.
- Multiple `-M` and `-m` options may be passed. *`msg[n]`* corresponds to *`match[n]`*.
- Check fails if any positive match isn't found, any negative match is found, the return code does not match the one supplied, or the command times out. Failure message is the nth *`msg`* supplied via `-M` option(s).

`check_cmd_status [-t` *`timeout`*`] [-r` *`returncode`*`]` *`command`*

- Executes *`command`* and checks return code against *`returncode`*.
- Check fails if return code does not match expected value or the command times out.

Scheduler Health: TORQUE/Moab

`check_moab_sched [-t` *`timeout`*`] [-m [!]`*`match`* `[…]] [-v` *`version`*`] [-a` *`alert`*`]`

- Runs "`mdiag -S -v`" and looks for anomalies in output. Also offers optional *`version`* check and matching against ALERTs in addition to arbitrary string *`match`*es per line.

`check_moab_rm [-t` *`timeout`*`] [-m [!]`*`match`* `[…]]`

- Runs "`mdiag -R -v`" and looks for anomalies in output. Checks all RMs for "Active" state in addition to arbitrary string *`match`*es per line.

`check_moab_torque [-t` *`timeout`*`] [-m [!]`*`match`* `[…]]`

- Runs "`qmgr -c 'print server'`" and looks for anomalies in output. Checks for "`scheduling = True`" in addition to arbitrary string *`match`*es per line.

Compute Node Cleanup – User-centric

`check_ps_unauth_users [action(s)]`

- Looks for processes owned by users not running jobs
- Supported actions:
  - `log` - Write message to NHC's log file
  - `syslog` - Write message to syslog
  - `die` - Error out as normal (this is the default action)
  - `ignore` - Do nothing
  - `kill` - Terminate the process via SIGKILL (<= `$MAX_SYS_UID`)
- Exempts users listed in `$NHC_AUTH_USERS` ("`root nobody`")
- Determines job users under TORQUE similarly to `reaver`; uses job script ownership for SLURM

Compute Node Cleanup – Proc-centric

`check_ps_userproc_lineage [action(s)]`

- Looks for processes not descended from daemon matching `$RM_DAEMON_MATCH` (default "`/\bpbs_mom\b/`" for TORQUE)
- Traces each PID/PPID until finding RM daemon or `init`
- Supported actions:
  - `log` - Write message to NHC's log file
  - `syslog` - Write message to syslog
  - `die` - Error out as normal (this is the default action)
  - `ignore` - Do nothing
  - `kill` - Terminate the process via SIGKILL (<= `$MAX_SYS_UID`)
- Exempts users listed in `$NHC_AUTH_USERS` ("`root nobody`")

Sockets

```
check_net_socket [-0] [-a] [-!] [-n name] [-p [!]proto]
   [-t [!]type] [-s [!]state] [-u [!]user] [-d [!]daemon]
   [-l [!]locaddr[:locport]] [-r [!]rmtaddr[:rmtport]]
   [-e action | -E action]
```

- Searches output of either `netstat` or `ss` for matching sockets based on protocol, type, state, user, daemon, and/or address. Check succeeds if match is found if, and only if, `-!` is not specified. Otherwise, it fails.
- Types are things like `STREAM`, `DGRAM`, or `SEQPACKET`. States are `LISTEN`, `CLOSE_WAIT`, etc.
- Each of the match fields is a standard NHC match string which may be negated.
- Arbitrary actions are also possible via `-e` (not found) or `-E` (found).
- Can be non-fatal (`-0`) and may be set to look for multiple matches (`-a`).

Running NHC via nhc-wrapper

`nhc-wrapper [-V] [-A arg(s)] [-D dir] [-M mailto] [-S subject] [-X timespec] [-L timespec[flags]] [-- arg(s)]`

- Runs `nhc` either once (default) or in a loop (`-L`) with the specified *`arg(s)`*.
- Send output, if any, to *`mailto`* with the subject of *`subject`*. Only report results if they differ from the previous run. Results may be set to expire (i.e., reported again) after a specific amount of time using `-X` *`timespec`*. The *`timespec`* is in the format *`WW`*`d`*`XX`*`h`*`YY`*`m`*`ZZ`*`s`*`QQ`*`f` specifying weeks, days, hours, minutes, and/or seconds along with a "fudge factor" (in seconds).
- The looping mode can be invoked via the `-L` option which runs in a loop in the terminal. A *`timespec`* of the same syntax as above may be specified, and it may be immediately followed by flags (case insensitive): `c` to clear the screen before each run, `r` to display a horizontal line (ruler) before each run, and/or `t` to display a timestamp before each run.

Using Detached Mode

- Set `DETACHED_MODE=1` in config or on command line.
- Forks `nhc` after parsing command line and `/etc/sysconfig/nhc` and reading `$RESULTFILE`
- Background process:
  - Reads config file
  - Runs all checks
  - Records results in `$RESULTFILE`
  - Marks node online/offline if `$MARK_OFFLINE` is 1
- Foreground process:
  - Acts on contents of `$RESULTFILE` from previous execution

What Makes a Cluster "Healthy?"

- Checking only the health of the compute nodes paints an incomplete picture, even from the job perspective:
  - Scheduler/RM services on the master
  - Filesystems' availability/utilization on the master
  - Web servers/services on other hosts
  - Interactive/login node services & availability
  - Data Transfer Nodes (DTNs) for data staging
- Best Practice: Run some checks "out-of-band" (via `nhc-wrapper`, `cron`, `pdsh`, etc.) instead of "in-band" (via RM)
  - Checks which take longer or may hang
  - Checks which need not run as frequently
  - May be either/or…but both are typically useful!
  - In-band checks are "binary" - result is either "good" or "failed"

Running NHC in Multiple Contexts

- Named contexts
  - Use `-n` or `$NAME` (or a symbolic link to `nhc`) to identify contexts
  - Different names use different configs (`/etc/sysconfig/`*`$NAME`* and `/etc/nhc/`*`$NAME`*`.conf`) but the same directories, scripts, and helpers
  - Set `$NHC_CHECK_ALL` and other context-specific variables in the configuration to keep invocation consistent
- Consider sharing context-specific configurations across nodes
  - If no single node needs multiple contexts, one config file will suffice!
  - Otherwise, each context's file can still be reused.

Implementation Strategies

"Best Practices" for multi-faceted system health:

- Use both in- and out-of-band simultaneously
  - In-band checks should be simple and fast; only "urgent" checks
  - Out-of-band checks should run less frequently (hourly?).
  - Consider using `-a` (or `$NHC_CHECK_ALL`) out-of-band (non-binary).
- Install on ALL nodes, including the master
  - Can use the same config file if nodes' hostnames are distinct.
  - Run periodically using `nhc-wrapper`, `crond`, `pdsh`, etc.
  - Make sure to monitor TORQUE/Moab/SLURM daemons' status and, optionally, configuration.
  - Perform checks on the host with the least user/job impact if possible (e.g., check shared filesystems on master).

NHC Resources

Previous Talks

- MoabCon 2012: http://go.lbl.gov/nhc-2012-mc
- SuperComputing 2012: http://go.lbl.gov/nhc-2013-sc
- MoabCon 2013: http://go.lbl.gov/nhc-2013-mc
- HPCAC SHPCEC 2014: http://go.lbl.gov/nhc-2014-hpcac
- MoabCon 2014: http://go.lbl.gov/nhc-2014-mc

Documentation

- In-tree: https://github.com/mej/nhc (choose branch from drop-down)

Source Code

- GH: https://github.com/mej/nhc | BB: https://bitbucket.org/mej0/nhc

Mailing Lists

- nhc@lbl.gov and/or nhc-devel@lbl.gov