Node Health Check (NHC)

for Linux Clusters Institute Introductory Workshop

Mike Renfro, PhD

Information Technology Services, Tennessee Tech University

2025-02-14

Acknowledgments

- Thanks to David Akin from Oklahoma University

The Goal of NHC

- We'd prefer everything about our environments to be totally reliable, but that's not realistic
- If we can't have total reliability, we should at least know what parts aren't working correctly
- Then we can minimize those parts' impact on the users' experience

So:

- Run a set of site-specific health checks on each node, mark unhealthy nodes as unavailable for jobs.
- Run checks as efficiently as possible to minimize performance impact.

(Also a glorious example of what `bash` can do without external programs.)

Analogy: Test-driven development

We might not know if a program code is "right", but we can definitely tell if it's **wrong**.

So fix it until it's not wrong any more.

Installation

- https://github.com/mej/nhc/releases/ for RPMs for EL7 and EL8
- `nhc-ohpc` package for OpenHPC
- https://github.com/basvandervlies/nhc-debian
- https://github.com/mej/nhc/blob/master/README.md#installation

Assuming Slurm, set:

- `HealthCheckInterval`
- `HealthCheckProgram`
- (optionally) `HealthCheckNodeState`

in `slurm.conf`.

Starting points

Start with any of:

- default nhc.conf
- empty nhc.conf
- output of `nhc-genconf`

`nhc-genconf` creates `nhc.conf.auto` using the current system's installed hardware, and can be helpful as a starting point for hardware-related health checks.

Configuration file syntax

```
target || check command with parameters
```

`target`: a *match string*, one of:

- glob or wildcard (e.g., `node*`)
- regular expression surrounded by `/` characters (e.g., `/node0[0-3][0-9]/` for `node000` through `node039`)
- node range expression surrounded by `{}` characters (e.g., `{node0[01-42]}` for `node001` through `node042`)

`check command with parameters`: any valid shell command that returns success or failure, including built-in NHC check functions and site-specific check functions.

Check arbitrary command results

- `check_cmd_output` executes a command and compares each line of its output against any match strings passed in.
- `check_cmd_status` executes a command and compares its exit status against a desired value.

Both of these will also fail if the command exceeds a timeout duration.

Check DMI (BIOS) data

- `check_dmi_data_match` uses parsed, structured data taken from the output of the `dmidecode` command to allow the administrator to make very specific assertions regarding the contents of the DMI data.
- `check_dmi_raw_data_match` is basically like a grep on the raw output of the `dmidecode` command.

Check file attributes or content

- `check_file_contents` looks at the specified file and allows one or more (possibly negated) match strings to be applied to the contents of the file.
- `check_file_stat` allows the user to assert specific properties on one or more files, directories, and/or other filesystem objects based on metadata returned by the Linux/Unix `stat` command.
- `check_file_test` allows the user to assert very simple attributes on one or more files, directories, and/or other filesystem objects based on tests which can be performed via the shell's built-in `test` command.

Check filesystems (inodes)

- `check_fs_inodes` ensures that the specified mountpoint has enough inodes.
- `check_fs_ifree` ensures that the specified mountpoint has enough **free** inodes.
- `check_fs_iused` ensures that the specified mountpoint hasn't **exceeded** a number of inodes.

Check filesystems (mounts)

- `check_fs_mount` examines the list of mounted filesystems on the local machine to verify that the specified entry is **present**.
- `check_fs_mount_ro` checks that a particular filesystem is mounted **read-only**.
- `check_fs_mount_rw` checks that a particular filesystem is mounted **read-write**.

Check filesystems (storage)

- `check_fs_free` checks that a particular filesystem has enough space available.
- `check_fs_size` checks that a particular filesystem is large enough.
- `check_fs_used` checks that a particular filesystem's storage hasn't exceeded a threshold.

Check hardware configuration

- `check_hw_cpuinfo` compares the OS-detected CPU(s) to ensure that the correct number of physical sockets, execution cores, and "threads" (or "virtual cores") are present and functioning on the system.
- `check_hw_eth` verifies that a particular Ethernet device is available.
- `check_hw_gm` verifies that the specified Myrinet device is available.
- `check_hw_ib` determines whether or not an active Infiniband link is present with the specified data rate (in Gb/sec).
- `check_hw_mcelog` queries the running mcelog daemon, if present. If the daemon is not running or has detected no errors, the check passes.

Check memory configuration

Verifying memory-related values fall within a specified range:

- `check_hw_mem` for total system memory (RAM + swap).
- `check_hw_mem_free` for free system memory (RAM + swap).
- `check_hw_physmem` for total physical memory (RAM).
- `check_hw_physmem_free` for free physical memory (RAM).
- `check_hw_swap` for total virtual memory (swap).
- `check_hw_swap_free` for free virtual memory (swap).

Check Moab/TORQUE status

For those of us using Moab and/or TORQUE, three specialty command checks, basically running a Moab/TORQUE command through `check_cmd_output`:

- `check_moab_sched` examines the output of `mdiag -S -v`.
- `check_moab_rm` examines the output of `mdiag -R -v`.
- `check_moab_torque` examines the output of `qmgr -c 'print server'`.

Check networking function

- `check_net_ping` provides an NHC-based wrapper around the standard Linux/UNIX `ping` command.
- `check_net_socket` checks for listening IP or Unix sockets.

Check NVIDIA GPU health

- `check_nv_healthmon` checks for problems with any NVIDIA Tesla GPU devices on the system.

Check process health (resource consumption)

Flagging any/all processes where a usage metric exceeds a threshold:

- `check_ps_cpu` for current percentage of CPU utilization.
- `check_ps_mem` for total memory consumption (including both physical and virtual memory).
- `check_ps_physmem` for physical memory consumption (i.e., resident RAM only).
- `check_ps_time` for total utilization of CPU time.

Check overall system health and performance

- `check_ps_kswapd` compares the accumulated CPU time (in seconds) between kswapd kernel threads to make sure there's no imbalance among different NUMA nodes (which could be an early symptom of failure).
- `check_ps_loadavg` looks at the 1-minute, 5-minute, and 15-minute load averages.

Check for rogue processes

- `check_ps_service` checks for running processes and process ownership. It can stop or kill processes that shouldn't be running, or start, cycle (stop and start), or restart services that should be running.
- `check_ps_unauth_users` checks user IDs on all running processes. If a process is owned by a regular user who **isn't** running a job, the process can be killed, the node can be marked as unhealthy, or the event can be logged.
- `check_ps_userproc_lineage` is similar to `check_ps_unauth_users`, except that it will kill any regular user processes that **weren't** started by a running job.

TN Tech's setup

Two cluster purchases, shared file server.

Impulse (2016) 44 nodes, dual Intel E5-2680v4, 64–896 GB RAM, 56 Gb Infiniband, 4 K80 GPUs. Hostnames (in Slurm format): `node0[01-42]`, `gpunode00[1-2]`.

Warp 1 (2021) 10 nodes, dual AMD 7713, 512 GB RAM, 100 Gb Infiniband, 20 A100 GPUs. Hostnames (in Slurm format): `gpunode0[03-12]`.

Need to make extensive use of node name patterns to avoid excessive copy/paste in `nhc.conf`.

CPUs

Impulse nodes have 2 14-core E5-2680v4

```
{node0[01-42]}     || check_hw_cpuinfo 2 28 28
{gpunode00[1-2]}   || check_hw_cpuinfo 2 28 28
```

Warp 1 nodes have 2 64-core EPYC 7713

```
{gpunode0[03-12]} || check_hw_cpuinfo 2 128 128
```

RAM

Impulse nodes have 64–896 GB, depending:

```
{node0[01-22]}     || check_hw_physmem 64GB 64GB 2GB
{node0[23-34]}     || check_hw_physmem 128GB 128GB 2GB
{node0[35-40]}     || check_hw_physmem 256GB 256GB 2GB
node041            || check_hw_physmem 384GB 384GB 2GB
node042            || check_hw_physmem 896GB 896GB 2GB
{gpunode00[1-2]}   || check_hw_physmem 384GB 384GB 2GB
```

Warp1 nodes have 512 GB:

```
{gpunode0[03-12]} || check_hw_physmem 512GB 512GB 2GB
```

In all cases, allow for ± 2 GB memory installed.

Swap and available memory

All nodes should have 2 GB of swap (±7%), and we should have some free memory (of either real RAM or swap):

```
* || check_hw_swap 2GB 2GB 7%
* || check_hw_mem_free 1MB
```

GPUs

All GPU nodes have at least 2 GPU devices.

```
gpunode*          || check_file_test -c -O -G -r
  -w /dev/nvidia(0,1)
```

But *Impulse* GPU nodes have 4

```
{gpunode00[1-2]} || check_file_test -c -O -G -r
  -w /dev/nvidia{2,3}
```

(each of these should be on one line.)

Ethernet

All nodes have an eth0, even if unused

```
*                   || check_hw_eth eth0
```

Impulse R730s also have an active eth2

```
{node04[1-2]}       || check_hw_eth eth2
{gpunode00[1-2]}    || check_hw_eth eth2
```

(Could also use a match string of: `{node04[1-2],gpunode00[1-2]}`.)

Infiniband

All nodes have an active ib0

```
*                       || check_hw_eth ib0
```

Impulse nodes have 56 Gb ConnectX-4

```
{node0[01-42]}      || check_hw_ib 56 mlx4_0:1
{gpunode00[1-2]}    || check_hw_ib 56 mlx4_0:1
```

Warp 1 nodes have 100 Gb ConnectX-5

```
{gpunode0[03-12]} || check_hw_ib 100 mlx5_0:1
```

Local filesystems

All nodes should have their root filesystem mounted read/write.

```
* || check_fs_mount_rw -f /
```

Remote filesystem example

All nodes should mount the `nfs4` export `files:/mnt/homes` read-write as `/home/tntech.edu`. If mount is absent, restart the systemd service that mounts the storage.

```
* || check_fs_mount_rw -t "nfs4"
  -s "files:/mnt/homes" -f "/home/tntech.edu"
  -e "systemctl restart home-tntech.edu.mount"
```

(should all be on one line)

File checks

Make sure temporary directories are directories, read/write/execute, and sticky:

```
* || check_file_test -r -w -x -d -k /tmp /var/tmp
```

These should always be readable and should never be empty:

```
* || check_file_test -r -s /etc/passwd /etc/group
```

Assert common properties for `/dev/null` (which occasionally gets clobbered):

```
* || check_file_test -c -r -w /dev/null
```

Miscellaneous

Check `sssd` and Active Directory:

```
* || check_cmd_status -t 30 -r 0 /usr/bin/id administrator
```