Cray's Node Health Checker: An Overview

Jason Sollom, *Cray, Inc.*

ABSTRACT: *As the size of Cray systems increases, the mean time to failure of an individual compute node decreases. As a result, Cray's Node Health Checker (NHC) is playing an ever increasing role in ensuring job completion and system administrator sanity. NHC is a system management tool that runs a series of built-in and system administrator defined health tests on compute nodes. NHC can be configured to automatically sequester unhealthy nodes, dump them for future debugging, and return them to the pool of available nodes via a reboot. Sequestering unhealthy nodes prevents them from causing job failure and lost compute time. Rebooting them increases the availability of nodes while simultaneously decreasing administrator intervention. This paper provides an overview of NHC.*

KEYWORDS: Node health checker, system management

Introduction

The Node Health Checker (NHC) is a system management tool that can be configured to detect a variety of problems on compute nodes that would render them unable to successfully run an application. Using NHC, system administrators can set configurable options to deal with these "unhealthy" nodes:

- Direct NHC to automatically reboot nodes, thereby erasing the problem.
- Direct NHC to sequester nodes from the rest of the system, preventing subsequent applications from running on them. Then, system administrators can manually assess the node's problem.
- Configure NHC to "dump" the nodes to an off-node location, so that the problem can be debugged later.

Without NHC

Without NHC, an unhealthy node can go undetected. Application after application could be assigned to this node, and each would fail while the node remained unhealthy. Because a parallel application can run across potentially tens of thousands of nodes, one unhealthy node can waste the efforts of the healthy nodes assigned to that application. Unhealthy nodes can waste large amounts of valuable compute time.

System Administrators are left to trial and error methods to discover unhealthy nodes. Each administrator has to devise his own discovery method which might range from looking for suspicious error messages in log files to running a "discovery" application on nodes to flush out problems.

With NHC

NHC can detect unhealthy nodes and deal with them. This prevents subsequent applications from falling victim to an unhealthy compute node, which improves system throughput. If so configured, NHC can reboot unhealthy compute nodes, immediately returning them to the pool of healthy compute nodes. Returning nodes to this "resource pool" improves system availability.

Compute Node Resource Pool

Compute nodes that are available to run applications are said to be part of the "resource pool". A node is only part of the resource pool if it is in the UP state. If it is in any other state, it is not part of the resource pool and will not be allocated to run an application.

NHC removes nodes from the resource pool by changing their states from UP to some other state. The different states have different meanings. See Table 1.

State	Meaning
UP	The node is healthy and can run applications.
DOWN	The node failed to boot or experienced a hardware problem.
ADMINDOWN	NHC detected a problem with the node. The node must be manually fixed.
SUSPECT	NHC detected a problem with the node and is monitoring it to see if it recovers. If the node recovers, it will be returned to the UP state.
UNAVAIL	NHC has scheduled the node to be rebooted.

Table 1. Node States and Meanings

When and Where NHC is Launched

NHC can be launched at one of three different times.

1. NHC can be launched when the compute node is booted or rebooted. Launching at this time is termed "NHC on Boot". NHC on Boot is launched locally on the compute node.
2. The Application Level Placement Scheduler (ALPS) launches NHC at the termination of an application. (ALPS launches applications via a command called *aprun*, so NHC is launched after every *aprun*.) Because launching via ALPS is the most frequent case, it is termed simply "NHC". ALPS launches NHC from a service node, typically the login node.
3. System administrators can manually launch NHC from any service node.

NHC on Boot[1]

NHC on boot detects problems on a newly booted node, so it protects against problems that happen at boot/reboot time.

When a node boots, the NHC on boot process is run as the last part of the boot process. The last run-level script initiates NHC on boot. NHC on boot uses its own configuration file, which is stored on the compute node, to determine which NHC health tests to run on the node. (See the Configuration File section for more details about the configuration file.) If any of these tests fail, NHC on boot prevents the node from booting. The node will remain in a DOWN state indefinitely. System

[1] The NHC on Boot feature will be supported in the Cray Linux Environment (CLE) 4.0 release.

administrator intervention is required to move the node out of this state. Any nodes that fail to boot do not join the resource pool.

NHC on boot can detect software problems such as a crucial services like ALPS failed to start on the node or that a needed file system was inoperable. It also can detect other problems like a GPU that is not functioning properly.

NHC (Launched by ALPS)

NHC launched by ALPS protects against problems that occur *after* the node has booted. These problems may have been caused by the application that just finished running on the node or by some other system problem.

ALPS launches applications from service nodes. Following the termination of the application, ALPS launches NHC from the same service node that the application was launched from. NHC will check on the nodes allocated to that application.

Applications can terminate either normally, meaning they exited successfully, or abnormally meaning they encountered an error. By default, NHC is configured to execute only when the application experienced an abnormal termination. However, NHC can be configured to execute after *every* application termination, normal or abnormal.

Starting on the service node, NHC creates a binary fan-out tree to contact each node allocated to the application. Any nodes that NHC cannot contact are initially left out of the fan-out tree. They will be dealt with later. From the service node, NHC pushes its configuration file down to the NHC daemon, named *xtnhd,* which is resident on each of the compute nodes. The configuration file contains the list of health tests to perform on the node. The NHC daemon launches these tests and reports the results back up the fan-out tree. This phase of testing is termed "*Normal Mode*". If a node fails any of the tests, NHC deals with this failure according to the action associated with the failed test. NHC may give the test multiple chances to pass if a second mode termed "*Suspect Mode*" is enabled. The actions associated with NHC health tests are discussed in the 'NHC Tests' section.

NHC (Manually launched by a system administrator)

A system administrator can also manually launch NHC from a service node.

No matter how NHC is launched, system administrators can set options to deal with unhealthy nodes by setting variables in the configuration files. See the 'Configuration Files' section for more information.

Normal Mode versus Suspect Mode

NHC starts its testing in Normal Mode. Normal Mode determines which nodes are healthy and which nodes may not be healthy. Nodes that fail health tests and nodes that are not contactable are considered unhealthy.

The purpose of Suspect Mode is to monitor unhealthy nodes and give them a chance to recover from transient problems. While in Suspect Mode, NHC continually reruns any failing health checks. If a node passes all of the health tests before Suspect Mode ends, it is returned to the UP state, meaning it is readmitted to the resource pool. If a health test is still failing at the end of Suspect Mode, NHC takes the action associated with the failing test. (See the Actions subsection under the 'NHC Tests' section.) The unhealthy node will not be allowed to run applications until it has returned to health either by the NHC action (i.e. a reboot) or System Administrator action.

Allowing nodes a chance to recover from potentially transient problems has benefits. NHC returns the node to the resource pool as soon as it recovers. This maximizes the number of nodes in the resource pool. If the nodes were not allowed time to recover but were immediately taken out of the resource pool, System Administrators could waste time debugging a node whose transient problem had vanished before the time of inspection.

From the NHC Configuration file, Suspect Mode can be disabled or enabled, and NHC behaves differently in each case. Suspect Mode is enabled by default.

Suspect Mode Disabled

If Suspect Mode is disabled, Normal Mode ends when all of the initial health tests have finished running. NHC deals with any node that fails a health test by executing the action associated with the failing test. Any node that NHC was unable to contact is set to the ADMINDOWN state. Then, NHC will exit, and ALPS is free to release the reservation of compute nodes. All healthy nodes return to the resource pool, and NHC has dealt with all the unhealthy nodes, effectively excluding them from the pool.

Figure 1 shows the state transitions for a node when Suspect Mode is disabled and the failing test has an action of ADMINDOWN. Figure 2 shows the state transitions for a node when Suspect Mode is disabled and the failing test has an action of REBOOT.

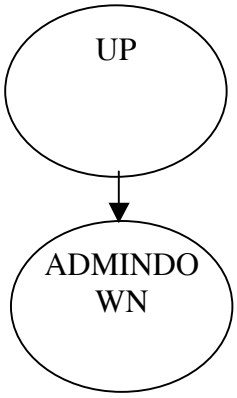

Figure 1: Node State Transition;
Suspect Mode: Disabled; Test Action: Admindown

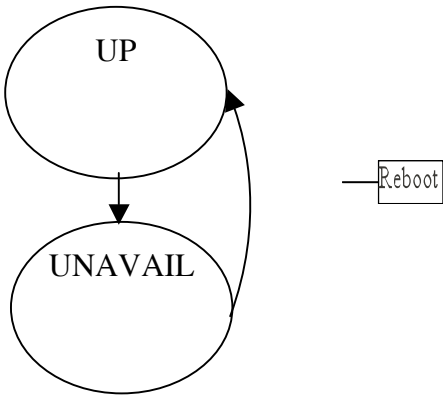

Figure 2: Node State Transition;
Suspect Mode: Disabled; Test Action: Reboot

Suspect Mode Enabled

If Suspect Mode is enabled, it imposes a maximum duration for Normal Mode. Normal Mode either ends normally or when it times out, whichever happens first. The *suspectbegin* variable in the NHC configuration file specifies the number of seconds before Normal Mode times out and Suspect Mode begins.

Any node that fails a health test or was not able to be contacted is set to the SUSPECT state. Any node still running a health test when Normal Mode expired is also set to SUSPECT state. All of the nodes in SUSPECT state are currently unhealthy nodes. If there are any unhealthy nodes, NHC exits Normal Mode and enters Suspect Mode to continue monitoring these unhealthy nodes. If there are no unhealthy nodes, Normal Mode simply exits. When NHC exits Normal Mode, ALPS is free to release the reservation. When the reservation is released, all healthy nodes are returned to the resource pool, and all unhealthy nodes are marked as SUSPECT, so they are excluded from the resource pool.

Starting out in Suspect Mode, NHC tears down the original fan-out tree and creates a new one composed only of unhealthy nodes. It attempts to contact nodes that it failed to contact previously. If it fails to contact any nodes, these nodes are left out of the new fan-out tree. When the Normal Mode fan-out tree is torn down all of the old NHC health tests from Normal Mode are killed. The tests are relaunched in Suspect Mode. NHC monitors these unhealthy nodes in Suspect Mode until one of two things occurs. If all of the SUSPECT nodes either recover or enter unrecoverable states, then Suspect Mode ends. Otherwise, Suspect Mode ends when it reaches a final time out specified by the *suspectend* variable in the NHC configuration file.

In Suspect Mode, NHC continually re-runs the failing health tests until they all pass or Suspect Mode times out. Any node that passes all of the tests has "recovered". NHC can also identify a node that encounters an unrecoverable situation, like a hardware failure.

Periodically within Suspect Mode, NHC will tear down and recreate its fan-out tree in the presence of recovered or unrecoverable nodes. Each time the fan-out tree is torn down all of the old NHC health tests it launched are killed. Tearing down the fan-out tree prevents an unrecoverable node from blocking communication by child nodes below it in the fan-out tree. This is necessary because nodes can enter unrecoverable situations after the NHC fan-out tree has been established. Once all of the health tests have finished on a recovered node, NHC returns it to the resource pool by changing its state to UP. Tearing down the fan-out tree when recovered nodes are detected, allows these recovered nodes to be speedily returned to the resource pool. A new fan-out composed of only unhealthy nodes is then recreated.

Periodically, NHC will also tear-down the fan-out tree and attempt to create a new fan-out tree that includes the nodes it was previously unable to contact.

At the end of Suspect Mode, any remaining nodes that are still failing health tests or that still have not been contacted are dealt with according to the action associated with their fail tests. (See the Actions subsection under the 'NHC Tests' section.)

Figure 3 shows the state transitions for a node when Suspect Mode is enabled and the failing test has an action of ADMINDOWN. Figure 4 shows the state transitions for a node when Suspect Mode is enabled and the failing test has an action of REBOOT.

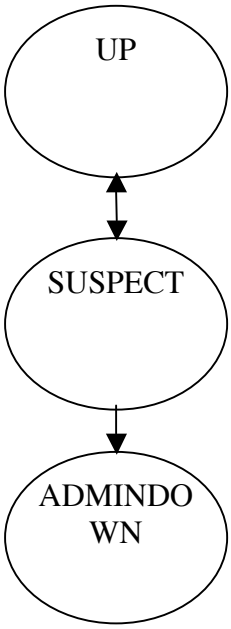

Figure 3: Node State Transition;
Suspect Mode: Enabled; Test Action: Admindown

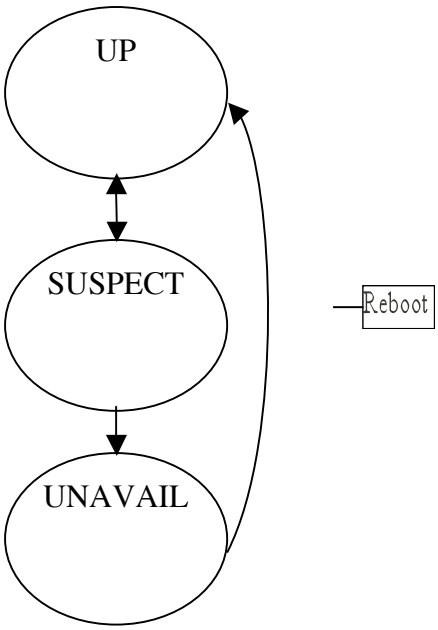

Figure 4: Node State Transition;
Suspect Mode: Enabled; Test Action: Reboot

NHC Tests

Test Actions

In the NHC configuration file, each NHC health test can be assigned an action that determines what occurs when the test fails.

If Suspect Mode is disabled and a test fails, then NHC executes the action at the end of Normal Mode. If Suspect Mode is enabled, then NHC executes the action if the test is still failing at the end of Suspect Mode.

An explanation of the terms *dumped* and *rebooted* is helpful to understand the NHC actions.

If a node is *dumped*, it means the node was sent a non-maskable interrupt (NMI) and then the node's memory is saved to a file. Nodes are dumped to help debug problems encountered by that node.

If a node is *rebooted*, the node is immediately marked UNAVAIL rather than ADMINDOWN and rebooted. The state UNAVAIL is used for rebooting because the ADMINDOWN state persists across a reboot, keeping the node out of the resource pool, but the UNAVAIL state does not persist. Once a node in the UNAVAIL state has been successfully rebooted, it returns to the UP state.

The dump/reboot daemon (*dumpd)*, handles dumping and rebooting compute nodes. See the 'Dumping and Rebooting Nodes' section for more information.

Table 2 lists the specified action, what NHC does, and the final state that a node ends up in after the execution of the list action.

Test action	What NHC does	Final state of the node
LOG	A warning is printed to the console	UP
ADMINDOWN	1. A warning is printed to the console 2. The node is set to the ADMINDOWN state	ADMINDOWN
DUMP	1. A warning is printed to the console 2. The node is set to the ADMINDOWN state 3. The node is dumped	ADMINDOWN
REBOOT	1. A warning is printed to the console 2. The node is set to the UNAVAIL state 3. The node is rebooted	UP
DUMP-REBOOT	1. A warning is printed to the console 2. The node is set to the UNAVAIL state 3. The node is dumped 4. The node is rebooted	UP

Table 2. Test action, NHC actions, and final node state

NOTE: The LOG action is **informational only**. LOG causes a warning to be issued when a test has failed, but nothing else happens. If a test with the LOG action fails, that failure will not cause a node to be removed from the resource pool (i.e. it will neither be marked SUSPECT or ADMINDOWN.) Such a failure will **not** cause NHC to enter into Suspect Mode. The LOG action should be used only for a test whose failure does not prevent the node from being usable by future applications.

Specific NHC Tests

There are five specific NHC health tests and one generic, customizable test.

ALPS Test

The ALPS test checks that the ALPS daemon is ready to receive application requests (i.e. *aprun requests*). If the daemon does not respond to NHC's query, the test fails.

Application Test

The Application test checks that all the processes associated with the application have exited. If the Application test detects application processes on the node, it polls until either all the processes exit or the Application test's time-out value is reached. If the process fails to exit within the time-out value, the test fails. The application is identified by its application ID (APID). ALPS provides NHC with the APID and this value is transmitted to the Application test. The time-out value is an argument to the test and can be set in the NHC configuration file.

Memory Test[2]

The Memory test checks that the amount of free memory is within a specified amount of the total memory available on the node. If the amount of free memory is less than this threshold, the test fails. The threshold is an argument to the Memory test and can be set in the NHC configuration file. The threshold is an absolute value of memory specified in megabytes.

The Memory test is dependent upon the Application Test because the application will be consuming some memory until it exits. The Memory test will not run until the Application Test successfully passes.

File Systems Test[2]

The File Systems test checks the mount point arguments given to it in the NHC configuration file. If the File Systems test is given no arguments, then it checks all of the mount points listed in the /etc/fstab file on the compute node.

[2] As of the CLE-3.0 release, this test can be run in Suspect Mode. Prior to that, it could only be run in Normal Mode (i.e. to enable this test Suspect Mode must have been disable.)

If a mount point is read-only, the test simply checks that it can access the '.' and '..' elements at that mount point. If the mount point is writable, the test creates a .nodehealth sub-directory at the mount point and writes a file to it. It then deletes the file. If any of these read, write, or delete operations fail, the test fails.

In the NHC configuration file, mount points can be excluded from being checked. The exclusion option is typically used when certain mount points in the /etc/fstab should not be checked.

GPU Test[3]

This test checks the health of a GPU present on the node by running a simple CUDA code on the GPU. The GPU test can optionally check that the amount of non-free memory on the GPU does not exceed a certain threshold, similar to the Memory test. The threshold is specified as an argument to the test in the NHC configuration file. If the GPU is determined to be unhealthy, then the test fails.

Plugin Test[1] (formerly called the Site Script Test)

This is the customizable NHC health test. The NHC Plugin test can launch any program that it has access to on the compute node. This program could be resident in the compute node's boot root file system or in a mounted file system. If the program launched by the Plugin test returns a non-zero value, it is considered a failure. A return value of zero indicates success. The Plugin test allows sites to create their own customised health tests.

The Cray document "Writing a Node Health Checker (NHC) Plugin Test," publication S-0023, details how to write a Plugin test.

Timed Values

Each test also has three other attributes associated with it: WarnTime, TestTime, and RestartTime

WarnTime

If the NHC health test runs for more than *WarnTime* seconds, then NHC issues a warning stating that the test is still running. This gives system administrators an early warning that the node may be unhealthy. It allows them time to take corrective action to remedy the problem.

TestTime

If the NHC health test runs for more than *TestTime* seconds, then the test fails.

RestartTime

In Suspect Mode, tests that fail are restarted. NHC waits for *RestartTime* seconds before restarting the tests. This attribute prevents tests from being restarted too quickly and causing more problems as a result. For example, restarting the NHC File System test too frequently may put additional pressure on a file system that is already unhealthy.

Dumping and Rebooting Nodes[4]

As alluded to in the 'NHC Tests' section, NHC can cause nodes to be dumped and/or rebooted by sending requests to the dump/reboot daemon, *dumpd*. This daemon is configurable, so actions associated with DUMP and REBOOT requests can be tailored by the site. For example, a site can control things like the maximum number of nodes dumped and the maximum amount of disk space dumps may occupy before all dumping ceases.

Error Reporting

NHC writes all of its errors to the console file which is accessible on the System Management Workstation (SMW). Each line is prefaced with <node_health: VERSION>, so that it is easily recognizable. VERSION refers to the current version of NHC, which is an internal version number and is independent of the Cray Linux Environment (CLE) version.

[3] This test will be supported in the CLE-4.0 release.

[4] Dumping and rebooting is supported in the CLE 3.1UP03 release.

Configuration Files

NHC

Most of NHC's behavior is configurable via the NHC configuration file located on the shared root (in file /etc/opt/cray/nodehealth/nodehealth.conf).

Because it is located on the shared root, all changes made to the file are immediately visible on all service nodes. Any subsequent invocation of NHC will see the changed file and act accordingly.

With the configuration file, much of NHC's behavior can be specified. Examples include:

- NHC can be configured on or off.
- Suspect Mode can be enabled or disabled.
- The durations of Normal Mode and Suspect Mode can be set.
- Attributes can be set for each test.
- NHC can be told how long to wait for responses from nodes before they should be considered as not contactable.
- The maximum number of nodes to dump per APID can be specified.
- NHC can be configured to check on nodes after every termination, normal or abnormal. (By default NHC checks on nodes where an application has terminated abnormally.)

The NHC configuration file is well commented and describes the various configuration options available.

NHC on Boot

The NHC on Boot configuration file is independent of the configuration file used for NHC when it is launched by ALPS (Previous section). This configuration file is stored on each of the compute nodes and is provided by the boot image delivered to each compute node. This file controls NHC on Boot's behavior in a similar fashion as mentioned in the previous section.

The Application Test is unnecessary for NHC on Boot because there are no applications running at node boot.

Use Cases

Use Case 1: Application fails; NHC detects a failure; Suspect Mode is enabled. Node recovers

Actors: ALPS, Application, NHC, node state

Trigger: ALPS launches an application which terminates abnormally. Upon termination, ALPS launches NHC.

Description: NHC detects a problem. It enters Suspect Mode and waits for the node to recover. The node recovers prior to Suspect Mode timing out. NHC returns the node to the resource pool.

Preconditions:

1. The node started in the UP state.
2. The application terminated abnormally.
3. There was a problem on the node that NHC can detect, such as an application hung on the node.
4. The action associated with the failed NHC health test is not the LOG action.
5. Suspect Mode is enabled.

Postconditions:

1. The transient problem is gone.
2. The node is returned to the UP state.

Normal Flow:

NHC enters Normal Mode and starts testing the health of the node. A NHC health test fails. NHC changes the node's state to SUSPECT. NHC enters Suspect Mode. It continues to re-test the node until the all of the tests pass. The node is set to the UP state.

Use Case 2: Application fails; NHC detects a problem; Suspect Mode is enabled. Node is set to ADMINDOWN.

Actors: ALPS, Application, NHC, node state

Trigger: ALPS launches an application which terminates abnormally. Upon termination, ALPS launches NHC.

Description: NHC detects a problem. It enters Suspect Mode and waits for the node to recover. The node never recovers, so NHC sets it to ADMINDOWN.

Preconditions:

1. The node started in the UP state.
2. The application terminated abnormally.

3. There was a problem on the node that NHC can detect, such as an application hung on the node.
4. The action associated with the failed NHC health test is the ADMINDOWN action.
5. Suspect Mode is enabled.

Postconditions:

1. The problem remains.
2. The node state is set to ADMINDOWN.

Normal Flow:

NHC enters Normal Mode and starts testing the health of the node. A NHC health test fails. NHC changes the node's state to SUSPECT. NHC enters Suspect Mode. It continues to re-test the node, but the test continues to fail. Suspect Mode reaches its final time-out. The node's state is set to ADMINDOWN.

Use Case 3: Application fails; NHC detects a problem; Suspect Mode is enabled. Node is set to ADMINDOWN and dumped.

Actors: ALPS, Application, NHC, node state

Trigger: ALPS launches an application which terminates abnormally. Upon termination, ALPS launches NHC.

Description: NHC detects a problem. It enters Suspect Mode and waits for the node to recover. The node never recovers, so NHC sets it to ADMINDOWN and dumps it.

Preconditions:

1. The node started in the UP state.
2. The application terminated abnormally.
3. There was a problem on the node that NHC can detect, such as an application hung on the node.
4. The action associated with the NHC health test that will fail is the DUMP action.
5. Suspect Mode is enabled.

Postconditions:

1. The problem remains.
2. The node state is set to ADMINDOWN.
3. The node is dumped.

Normal Flow:

NHC enters Normal Mode and starts testing the health of the node. A NHC health test fails. NHC changes the node's state to SUSPECT. NHC enters Suspect Mode. It continues to re-test the node, but the test continues to fail. Suspect Mode reaches its final time-out. The node's state is set to ADMINDOWN. The node is dumped.

Use Case 4: Application fails; NHC detects a problem; Suspect Mode is enabled. Node is set to rebooted.

Actors: ALPS, Application, NHC, node state

Trigger: ALPS launches an application which terminates abnormally. Upon termination, ALPS launches NHC.

Description: NHC detects a problem. It enters Suspect Mode and waits for the node to recover. The node never recovers, so NHC reboots the node.

Preconditions:

1. The node started in the UP state.
2. The application terminated abnormally.
3. There was a problem on the node that NHC can detect, such as an application hung on the node.
4. The action associated with the NHC health test that will fail is the REBOOT action.
5. Suspect Mode is enabled.

Postconditions:

1. The problem remains.
2. The node is rebooted.
3. When the reboot completes, the node is returned to the UP state and the problem is erased.

Normal Flow:

NHC enters Normal Mode and starts testing the health of the node. A NHC health test fails. NHC changes the node's state to SUSPECT. NHC enters Suspect Mode. It continues to re-test the node, but the test continues to fail. Suspect Mode reaches its final time-out. The node's state is set to UNAVAIL. The node is rebooted. When the reboot completes, the node is returned to the UP state and the problem is erased.

Use Case 5: Application fails; NHC detects a problem; Suspect Mode is enabled. Node is dumped and rebooted.

Actors: ALPS, Application, NHC, node state

Trigger: ALPS launches an application which terminates abnormally. Upon termination, ALPS launches NHC.

Description: NHC detects a problem. It enters Suspect Mode and waits for the node to recover. The node never recovers, so NHC dumps it and reboots it.

Preconditions:

1. The node started in the UP state.
2. The application terminated abnormally.
3. There was a problem on the node that NHC can detect, such as an application hung on the node.
4. The action associated with the NHC health test that will fail is the DUMP-REBOOT action.
5. Suspect Mode is enabled.

Postconditions:

1. The problem remains.
2. The node state is set to UNAVAIL.
3. The node is dumped.
4. The node is rebooted.
5. When the reboot completes, the node is returned to the UP state and the problem is erased.

Normal Flow:

NHC enters Normal Mode and starts testing the health of the node. A NHC health test fails. NHC changes the node's state to SUSPECT. NHC enters Suspect Mode. It continues to re-test the node, but the test continues to fail. Suspect Mode reaches its final time-out. The node's state is set to UNAVAIL. The node is dumped. The node is rebooted. When the reboot completes, the node is returned to the UP state and the problem is erased.

Use Case 6: Application fails; NHC detects a failure; Suspect Mode is disabled. Node is set to ADMINDOWN.

Actors: ALPS, Application, NHC, node state

Trigger: ALPS launches an application which terminates abnormally. Upon termination, ALPS launches NHC.

Description: NHC detects a problem. NHC sets the node's state to ADMINDOWN.

Preconditions:

1. The node started in the UP state.
2. The application terminated abnormally.
3. There was a problem on the node that NHC can detect, such as an application hung on the node.
4. The action associated with the NHC health test that will fail is the ADMINDOWN action.
5. Suspect Mode is disabled.

Postconditions:

1. The problem remains.
2. The node state is set to ADMINDOWN.

Normal Flow:

NHC enters Normal Mode and starts testing the health of the node. A NHC health test fails. NHC changes the node's state to ADMINDOWN.

Use Case 7: Application succeeds; NHC is configured to not check on applications that succeed.

Actors: ALPS, Application, NHC, node state

Trigger: ALPS launches an application which terminates normally. Upon termination, ALPS launches NHC.

Description: The application exited normally. NHC was configured to **not** check on nodes when their applications terminate normally. NHC exits.

Preconditions:

1. The node started in the UP state.
2. The application terminated normally.
3. NHC is configured to **not** check on nodes when the application terminates normally.

Postconditions:

1. The node remains in the UP state.

Normal Flow:

NHC is configured to **not** check on nodes when the application terminates normally. NHC determines the application terminated normally. NHC exits without doing any checking.

Use Case 8-13:

Actors: ALPS, Application, NHC, node state

Trigger: ALPS launches an application which terminates normally. Upon termination, ALPS launches NHC.

Description: These six Use Cases are the same as Use Cases 1-6 except that NHC is configured to check on nodes whose applications have terminated normally rather than abnormally. The applications in these Use Cases all terminated normally, but a problem still existed on the node causing a NHC health test to fail. While there is less chance that a node will be unhealthy after an application terminates normally, the possibility still exists. Therefore, NHC can be configured to check on nodes after a normal termination.

NHC Security: SSL

The NHC daemon on the compute nodes authenticates requests from the NHC client on the service nodes using the Secured Sockets Layer (SSL) protocol. While enabling NHC's SSL security feature is optional, Cray recommends enabling it. Only root users can launch NHC. The NHC daemon will reject any unauthenticated requests. The Cray document "Managing System Software for Cray XE and Cray XT Systems, " publication S-2393, details enabling SSL.

Service Node Crash Recovery

If a service node crashes while NHC is doing a health check, then that NHC invocation will be lost. However, when the service node is rebooted, a NHC recovery script is run that automatically relaunches any NHC checks that were in progress at the time of the crash. As a result, all lost checks are redone, and no nodes are stranded in SUSPECT state.

If the service node is not rebooted, the NHC documentation describes how to manually recover.

Additional Information

NHC information can be found in the following Cray documentation:

- "Managing System Software for Cray XE and Cray XT Systems, " publication S-2393
- Man pages: intro_NHC(8), nhc_recovery(8), xtcheckhealth(8), xtcleanup_after(8), dumpd(8)
- "Writing a Node Health Checker (NHC) Plugin Test," publication S-0023

Acronyms

NHC: Node Health Checker
ALPS: Application Level Placement Scheduler

Acknowledgements

The author would like to acknowledge the editorial efforts of his co-workers: Scott Wermager, Kent Thomson, David Henseler, Brad Stevens, Peggy Gazzola, and Linda Mikkelsen.

About the Author

Jason Sollom works as a developer in the Cray Management Software (CMS) group.