ParaStation HealthChecker

Software Product Detailed Description

Product:	ParaStation HealthChecker
Date:	June 2012
Document number:	PSHC-1.0-4en

This software product description documents the functionality provided by the ParaStation HealthChecker as well as the system prerequisites required for installation and operation, licensing scheme and other useful information.

Overview

The ParaStation HealthChecker as part of the *ParaStationV5* cluster suite is an elaborated test suite to ensure the usability of compute and service nodes and the system in general of a compute cluster. It includes a framework to define and run a set of pre-defined *tests*. Each particular test analyzes a particular function on a node or within the system. It returns only a single state, which might be OK, WARNING or FAILED. Each test is optimized to very fast analyze in a non-destructive way an aspect of the system. Tests may use all available information. This include in particular tools for reading the state of a RAID controllers, reading operating system parameters, IPMI or SNMP queries and more.

Each test is run within the *framework* provided by the HealthChecker. This framework ensures that the test itself does not disturb the system. E.g., it monitors each test with a timeout and kills hanging tests. The framework is designed to run on a large number of nodes in parallel and not to stress the network or global file system. Therefore it scales to thousands of nodes.

Primarily, the HealthChecker analyzes cluster nodes, but tests for system-wide problems, like Infiniband fabric analytics, are available, too.

Each test is configurable with respect to the system it is testing. E.g. different nodes may have different limits for a particular parameter. Configurations may be grouped within classes describing a group of nodes with common parameters. Commonly used classes are *compute-nodes*, *admin-nodes*, *login-nodes*, etc.

Tests and test sets

Currently, about 50 single tests are implemented. New tests may be added using commonly used scripting languages like any shell (*bash)*, *perl* or *python*.

The following is an incomplete list of available tests, which are known to address problematic aspects:

- BIOS Version and installation date
- Number of available CPUs
- CPU speed and type
- Running services (e.g. *syslog*, *xinetd*, *pbs_mom* ....)
- Disk space
- Disk health (*smart*)
- Infiniband bandwidth, error counters and connectivity
- Kernel version
- Working ldap
- Checksum of critical configuration files
- Free memory (memory leakage in the kernel!)
- Memory bus speed and size
- Mounted file systems
- DNS configuration and availability
- Network counters, Ethernet connectivity and speed
- Software versions

Tests are grouped together within *testsets*. Each testset defines a set of tests for a particular purpose. Commonly used testsets are:

Name	*Description*	*Appr. runtime*
reboot	Tests which are executed on every reboot of a node.	20 sec
stress-test	Stress testing the hardware after a hardware failure or after a reinstall of a node. Especially the memory and network is tested.	30 min
epilogue	Tests to check if a job did not trash a node, e.g. leaving orphaned processes or eating up memory in a non-reclaimable way.	10 sec
prologue	Tests to check if a node is ready to run a new job.	15 sec
ib-test	Tests Infiniband connectivity and error counters	10 sec
manual	Almost all tests run while manually testing a node. This does not include a stress test.	1 min

The tests contained in a particular testset may be run manually, periodically or event driven using the command *pshealthcheck*. Events driving a health check run are:

- Reboot: while rebooting a node, all tests for the testset *reboot* are executed. This ensures that all necessary resources are available after booting a node. Resources include available memory, mounted file systems, number of CPUs, network connections, network configuration (speed) and many more. This is especially useful during bring-up phase of a system and after replacing a node.
- Job start: while starting up a new job, the batch queuing system runs the ParaStation HealthChecker with the testset *prologue*. These tests ensure that all resources required to successfully run a job are in place.
- Job termination: while terminating a job, the batch queuing system runs the HealthChecker with the testset *epilogue*.

Any testset, especially *manual* and *stress-test*, may be run by the administrator at every time to check the overall state of a particular node. For non-compute nodes, like file servers or administration nodes, periodical health checks may be run. For compute nodes, this probably makes no sense, as these tests may disturb jobs too much.

All results of all runs are stored within logfiles, therefore the reason for bouncing a job or setting a node off-line may be found easily.

Automated actions

Actions might be defined to be executed before or after a HealthChecker run. Actions executed before can be used to prepare the system for the run, e.g. killing remaining user processes in test set epilogue. Actions executed after a run get its result and can behave accordingly. By default, such actions only doing something in case of an error include

- Notification of the administrators by email.
- Setting a node off-line within the batch queuing system, if a reboot, prologue or epilogue test set failed.
- Creating a trouble ticket.
- Notification of hardware replacement.

Tests may be run by the administrators in a test-only mode, therefore not automatically triggering the configured actions.

The default action for compute nodes links the ParaStation HealthChecker with the installed batch queuing system. If at least one of the tests failed, the node is automatically taken off-line and an appropriate note is added to the node properties within the batch queuing system. For example, the message

```
pshealthcheck - 2010-02-18 00:26:51 - (1/23) - memory_free - ts prologue
```

is added to the node's properties. Therefore, bad nodes are automatically removed from the list of available nodes and no longer used for future jobs. The administrator may easily find the reason why this node is taken off-line and when this happened.

Monitoring Nodes

As explained in the previous chapter, compute nodes are monitored on a regular basis using the HealthChecker. Each time a node is rebooted, the testset *reboot* is executed. This ensures that all the initialization and configuration done by the BIOS and the operating system detected all the hardware components and initialized them properly. On large systems, sporadically failures while initializing hardware or software components sum up and may prevent a job from successfully being executed. Typically found issues include speed of the Ethernet network, memory initialization or missing CPUs or cores on a node. In addition, it ensures that all system services are up and running.

Corresponding to the *reboot* testset, the testsets *prologue* and *epilogue* are run on each node of the node list while a job is starting up or terminating, respectively. This ensures that all resources typically required by a job are available at job start-up and still available after job termination. As explained before, bad nodes are automatically taken off-line. In addition, jobs are “bounced back” to the batch queuing system if a prologue check for a particular node failed. The job is typically immediately rescheduled without using this particular node.

Monitoring the Infiniband fabric

The ParaStation HealthChecker includes a module to periodically analyze the state of the entire Infiniband fabric. The test reads all the error counters available within the fabric using standard Infiniband tools. The counters are reset afterwards, therefore periodic samples of all counters are collected.

All counters are stored within a flat file database indexed by the node's GUID and port. This data files hold all counters for a particular node for all periods and may be analyzed using the usual tools like *grep*. Tools are available to search the database for entries exceeding predefined thresholds.

While collecting all counters for a sampling period, a predefined set of counters is compared against appropriate thresholds. Whenever a value exceeds the threshold, all related information is written to a logfile. Additional actions may include disabling the IB port, if this is a fabric-internal port or taking a compute node off-line, if a compute node is involved. In addition, a new trouble ticket may be opened if not yet existing or the ticket number of an existing ticket will be reported.

Reported information include the faulty GUID and port number, the error counter names and values, the port label and the other end's port label. If applicable, node names are reported, too. Therefore, bad IB nodes can easily mapped to compute nodes or cables.

In addition, port parameters are also monitored. All ports not running with proper link width or speed are reported.

Beside analyzing error counters within the fabric, the availability of the subnet manager is monitored. Fail-over of redundant subnet managers are also detected.

Monitoring MCE errors

Machine check event (MCE) errors may be monitored by the ParaStation HealthChecker module *mceloghandler* to set nodes automatically off-line and create appropriate tickets.

Installation prerequisites

To install and operate the ParaStation HealthChecker, the following prerequisites must be met:

Supported hardware architectures and operating systems:

ParaStation HealthChecker is supported on all major Linux distributions and hardware platforms:

Architecture	*Linux distributions*
x86_64	OpenSuSE, SLES11, RHEL5, RHEL6, Fedora, CentOS, Scientific Linux
Intel IA32 and AMD Athlon	OpenSuSE, SLES11, RHEL5, RHEL6, Fedora, CentOS, Scientific Linux

Supported are single-processor nodes as well as SMP nodes. SMT is also supported.

Free disk space: The installation of all ParaStation HealthChecker packages requires approximately 10MB free disk space per node. Administrator privileges are required for installation and execution.

Running *pshealthcheck* in combination with a batch queuing system is currently tested with Torque/Moab. For automated ticket handling, Trac is supported (see http://trac.edgewall.org).

Media

The ParaStation HealthChecker is available as RPM packages ready for installation on the target system. Please contact support@par-tec.com how to obtain the packages.

License

ParaStation HealthChecker is not freeware, although it is available in source form! Details can be found in the ParaStation license agreement on www.par-tec.com.

Support

After signing a support contract, support for all packages is granted for the agreed period of time. The maximum response time is next business day. Support is performed by telephone, email, and/or remote login. On-site support at the installation site is not included.

The support comprises all *ParaStationV5* components as well as the open source software utilized (if applicable). Other open source software tools that have been provided free of charge are only supported if resources are available, a general claim cannot be advanced on the basis of this support agreement.

Scope of delivery

ParaStation HealthChecker comprises the following components:

- Software packages (pshealthcheck, pshealthcheck-binhelpers, pshealthcheck-config),
- Documentation (ParaStation Healthchecker Administrator’s Guide) in electronic format,
- Support as agreed in the support contract.

Copyright

ParTec, ParaStation and *ParaStationV5* are registered trademarks of ParTec Cluster Competence Center GmbH. All other product and brand names are trademarks or registered trademarks of their respective owners.

The information in this version of the software product detailed description is valid as from the time of publishing. Errors & omissions excluded.

Further information

For further information about ParaStation HealthChecker visit http://www.par-tec.com or send an email to sales@par-tec.com.