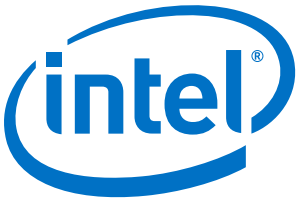

Intel® Cluster Checker User's Guide

Version 3.1.2

January 15, 2016

Disclaimer and Legal Information

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

A "Mission Critical Application" is any application in which failure of the Intel Product could result, directly or indirectly, in personal injury or death. SHOULD YOU PURCHASE OR USE INTEL'S PRODUCTS FOR ANY SUCH MISSION CRITICAL APPLICATION, YOU SHALL INDEMNIFY AND HOLD INTEL AND ITS SUBSIDIARIES, SUBCONTRACTORS AND AFFILIATES, AND THE DIRECTORS, OFFICERS, AND EMPLOYEES OF EACH, HARMLESS AGAINST ALL CLAIMS COSTS, DAMAGES, AND EXPENSES AND REASONABLE ATTORNEYS' FEES ARISING OUT OF, DIRECTLY OR INDIRECTLY, ANY CLAIM OF PRODUCT LIABILITY, PERSONAL INJURY, OR DEATH ARISING IN ANY WAY OUT OF SUCH MISSION CRITICAL APPLICATION, WHETHER OR NOT INTEL OR ITS SUBCONTRACTOR WAS NEGLIGENT IN THE DESIGN, MANUFACTURE, OR WARNING OF THE INTEL PRODUCT OR ANY OF ITS PARTS.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Requires a system with a 64-bit enabled processor, chipset, BIOS and software. Performance will vary depending on the specific hardware and software you use. Consult your PC manufacturer for more information. For more information, visit http://www.intel.com/info/em64t

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or go to: http://www.intel.com/design/literature.htm

Optimization Notice

Intel compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804

Contents

Chapter 1

Introduction

1.1 What is Intel® Cluster Checker?

Intel® Cluster Checker verifies the configuration and performance of Linux-based clusters and checks compliance with the Intel® Cluster Ready architecture specification. If issues are found, Intel® Cluster Checker diagnoses the problems and may provide recommendations on how to repair the cluster.

This guide provides step-by-step instructions for using the tool.

Further support information can be found at `http://www.intel.com/go/cluster`.

1.2 Key Concepts

Intel® Cluster Checker operates on similar principles as a medical diagnosis. In other words, Intel® Cluster Checker is a doctor for clusters. The key conceptional ideas are *signs* and *diagnoses*. Signs are objective indicators based on observations from the cluster. For example, a sign is created when the performance of a node is less than expected or a configuration setting is incorrect. A sign is roughly analogous to a symptom in a medical setting.

Intel® Cluster Checker 2.x transition note: Sub-tests are roughly analogous to signs, but there is no equivalent to diagnoses.

A diagnosis combines one or more signs to identify the root cause of an issue. For example, a diagnosis is created when high MPI latency measurements are associated with Ethernet configuration settings that enable interrupt coalescing. The diagnosis assigns the root cause of the former to the latter.

Each sign and diagnosis has corresponding severity and confidence levels that range from 0 to 100 percent. Higher values indicate the item is more serious or has a higher confidence.

Intel® Cluster Checker 2.x transition note: The confidence and severity level of all reported issues are implicitly 100%.

Chapter 2

Getting Started

Before using Intel® Cluster Checker for the first time, the runtime environment must be setup. Two files are included to setup the runtime environment, `clckvars.sh` for shells with Bourne* syntax and `clckvars.csh` for shells with csh syntax. Source the appropriate file from the command line, for example:

```
source /opt/intel/clck/3.1.2/bin/clckvars.sh
```

Tip: To avoid having to run this command for each new shell, add it to your shell initialization file, e.g., `.bashrc` or `.cshrc`.

Next, create a nodefile containing the list of compute node hostnames or IP addresses, one per line. For example, if the cluster consists of a dedicated head node named `front-end` and four compute nodes named `node1`, `node2`, `node3`, and `node4`, the corresponding nodefile would contain:

```
node1
node2
node3
node4
```

The preceding two steps are one time prerequisites, unless nodes are added to or removed from the cluster. With this out of the way, using Intel® Cluster Checker is a three step process:

1. Collect data
   `clck-collect -a -f nodefile`
2. Analyze the data
   `clck-analyze -f nodefile`
3. Resolve any issues reported in step 2 and repeat steps 1 and 2 until no issues are reported.

Intel® Cluster Checker 2.x transition note: Steps 1 and 2 were previously a single step. Splitting data collection and analysis into separate steps avoids unnecessary and duplicate data collection from problem-free components / nodes.

Intel® Cluster Checker 2.x transition note: The `clck` tool is included for limited command line compatibility with Intel® Cluster Checker 2.x. This tool combines these two steps and automatically runs both data collection and analysis. In most cases, using the separate `clck-collect` and `clck-analyze` programs is recommended. More information can be obtained by running `clck --help`.

Tip: Note that it is usually only necessary to recollect data for items where an issue was detected.

Data collection is described further in chapter 3 and analysis in chapter 4.

Chapter 3

Data Collection

3.1 Overview

Before Intel® Cluster Checker can identify issues, the "vital signs" of the cluster must first be collected. Data can be collected either *on-demand* or *asynchronously*. Several use cases are described in section 3.7. In both modes, the collected data is stored in a database and the data to be collected are defined by the same set of data providers.

With the exception of sections 3.6 and 3.7, this chapter assumes that *on-demand* data collection is used.

Intel® Cluster Checker 2.x transition note: While the Intel® Cluster Checker version 2 collected data is stored in a database, the database is not used for analysis. Fresh data is collected on every run of the tool.

3.2 Running `clck-collect`

In on-demand mode, data is only collected upon request. The `clck-collect` program is the tool used to invoke the data collection.

A typical invocation of `clck-collect` is:

```
clck-collect -a -f nodefile
```

Tip: To collect data on a single node, use a nodefile containing a single line with the node hostname.

This command includes the two elements required at a minimum for on-demand data collection, the set of data providers and the list of nodes. The `-a` command line option specifies that all data providers should be run and the `-f` option specifies the file containing the list of nodes.

Some of the data providers create temporary files and it is required that these temporary files are visible to all nodes in the cluster.

When `clck-collect` is executed by a non privileged user, a temporary directory will be created in the `.clck` directory within the user's home directory to store the temporary files. This value can be over ridden by setting the `CLCK_SHARED_TEMP_DIR` environment variable.

When `clck-collect` is run as root, it is required to set the value of the `CLCK_SHARED_TEMP_DIR` environment variable to the path of a directory that is visible to all nodes in the cluster. Failure to set this environment variable will usually result in an aborted run with a message about not

being able to create a shared temporary directory.

3.3 Selecting Nodes

The nodefile contains a list of cluster node hostnames or IP addresses, one per line.

For compute nodes, the nodefile is a simple list of nodes. For instance, the nodefile provided by a cluster resource manager typically contains just compute nodes and may be used as-is.

The nodefile is specified using the following `clck-collect` command line option.

`-f` *file* / `--nodefile` *file*

Use the specified file for the list of nodes on which to collect data.

Intel® Cluster Checker 2.x transition note: Previous nodefiles are compatible. The type annotation is considered synonymous with role, and the arch and group annotations are ignored.

Intel® Xeon Phi™ coprocessors should be included in the nodefile as "independent" nodes.

However, in some cases, nodes in the nodefile need to be annotated. The # symbol may be used to introduce comments in a nodefile. Annotations are specially formatted comments containing an annotation keyword following by a colon and a value. Annotations may alter the data collection behavior.

3.3.1 Node Roles

The `role` annotation keyword is used to assign a node to one or more roles. A role describes the intended functionality of a node, e.g., a node is a compute node. If no role is explicitly assigned, by default a node is assumed to be a compute node. Valid node role values are given in Table 3.1. The role annotation may be repeated to assign a node multiple roles.

For example, the following nodefile defines 4 nodes: `node1` is a head and compute node, `node2`, `node3`, and `node4` are compute nodes, and `node5` is disabled.

```
node1   # role: head  role: compute
node2   # role: compute
node3   # implicitly assumed to be a compute node
node4
#node5
```

Some data providers will only run on nodes with certain roles, e.g., data providers that measure performance typically only run on compute or enhanced nodes.

3.3.2 Subclusters

Some clusters contain groups of nodes, or subclusters, that are homogeneous within the subcluster but differ from other subclusters. For example, one subcluster may be connected with InfiniBand* while the rest of the cluster uses Ethernet*.

Role	Description
boot	Provides software imaging / provisioning capabilities
compute	Is a compute resource (mutually exclusive with enhanced)
enhanced	Provides enhanced compute resources, e.g., contains additional memory (mutually exclusive with compute)
external	Provides an external network interface
head	Alias for the union of boot, external, job_schedule, login, network_address, and storage
job_schedule	Provides resource manager / job scheduling capabilities
login	Is an interactive login system
network_-address	Provides network address to the cluster, e.g., DHCP
storage	Provides network storage to the cluster, e.g., NFS

Table 3.1 – Node Roles

The `subcluster` annotation keyword is used to assign a node to a subcluster. A node may only belong to a single subcluster. If no subcluster is explicitly assigned, by default the node is placed into the the default subcluster. The subcluster name is an arbitrary string.

For example, the following nodefile defines 2 subclusters, each with 4 compute nodes:

```
node1   # subcluster: eth
node2   # subcluster: eth
node3   # subcluster: eth
node4   # subcluster: eth
node5   # subcluster: ib
node6   # subcluster: ib
node7   # subcluster: ib
node8   # subcluster: ib
```

By default, cluster data providers will not span subclusters. To override this behavior, use the following `clck-collect` command line option:

`-S` / `--ignore-subclusters`

Ignore subclusters when running cluster data providers, i.e., cluster data providers will span subclusters. The default is not to span subclusters.

3.4 Selecting Data Providers

Several `clck-collect` command line options available to specify the data providers to run, including:

`-a` / `--all`

Run all data providers in the default data provider directory (`/opt/intel/clck/3.1.2/provider/etc`).

`-m` *name* / `--module` *name*

Run the individual data provider matching the specified name. For example, `clck-collect -m uname -f nodefile` runs the uname provider from the default data provider directory (`/opt/intel/clck/3.1.2/provider/etc/uname.xml`).

`-p` *file* / `--provider` *file*

Run the individual data provider defined by the specified file. For example, `clckd-collect -p $HOME/myprovider.xml -f nodefile` runs the provider defined by `myprovider.xml`.

`-s` *name* / `--set` *name*

Run the set of data providers matching the specified name. The sets are defined in `/opt/intel/clck/3.1.2/etc/sets.xml`. For example, `clck-collect -s benchmarks -f nodefile` runs all the data providers that use benchmarks to measure performance. Sets are pre-defined for each analysis check to collect all the data necessary for the analysis. The list of sets can be found in Appendix C.

Tip: To check a single aspect of a cluster, combine sets with the ability to exclusively run an analysis check (section 4.6). E.g., `clck-collect -f nodefile -s foo && clck-analyze -I foo`, where `foo` is the name of the check.

Any of these options may be combined to run the union of the specified data providers.

3.5 Other Configuration Options

Other command line configuration options supported by `clck-collect` include:

`-c` *file* / `--config` *file*

Use the specified configuration file. The default is `/opt/intel/clck/3.1.2/etc/clckd.xml`.

`-l` *level* / `--log-level` *level*

Select the output level. Recognized levels, in decreasing order, are `alert`, `critical`, `error`, `warning`, `notice`, `info`, and `debug`. The default value is `info`.

`-z`

This option disables the embedded functionality that directly writes the collected data to the database. Use this option only when the `clck-serverd` service is running, otherwise the data may be lost! See section 3.7 for more information on when this option should, and should not, be used.

In addition to the options already described, the data collection behavior can also be customized via a configuration file. The default configuration file is `/opt/intel/clck/3.1.2/etc/clckd.xml`. The configuration file options are described further in the configuration file itself.

3.6 Asynchronous Data Collection

In addition to on-demand data collection, Intel® Cluster Checker also supports asynchronous data collection. In asynchronous mode, a service runs in the background on all nodes that wakes up periodically to collect data and populate the database all without any manual intervention. Asynchronous data collection is not enabled by default.

Tip: All nodes that have the compute role should run `clckd`, e.g., a head node that is also used as a compute node should run both daemons.

The service takes the form of two daemons, `clckd` and `clck-serverd`. `clckd` runs on each compute node to run the data providers. `clck-serverd` aggregates all the data collected by the `clckd` daemons into a central database. Unlike `clckd`, only one instance of `clck-serverd` should be running, typically on the head node.

3.6.1 Setting Up the Daemons

Tip: If on-demand data collection with a shared database is desired (see section 3.7), then the user can be a local user on the head node without ssh access to the compute nodes.

Tip: It is highly recommended **not** to run the daemons as a privileged user.

In order to run continuously in the background, the daemons need to be configured by a privileged user to start automatically when the nodes boot. This is a one-time setup process. The precise steps vary depending on the Linux* distribution, but the general process is the same.

1. Create a cluster user dedicated to running the daemons. By default, the provided service start-up files assume the user is named `clck`. If a different user name is used, then the service start-up files will need to be edited. Regardless of the name, the user needs to be able to ssh from one node to another without a password.
2. Copy the service start-up files from the appropriate Linux* distribution sub-directory of `/opt/intel/clck/3.1.2/extra` to the corresponding system directory.
3. Use the appropriate commands for the Linux* distribution to enable the services.
4. Use the appropriate commands for the Linux* distribution to start the daemons.

The following subsections describe these steps more precisely for particular Linux* distributions. These steps can only be performed by a privileged user.

3.6.1.1 Intel® Manycore Platform Software Stack

The files corresponding to this Linux* distribution are located in `/opt/intel/clck/3.1.2/extra/mpss`.

1. Copy `/opt/intel/clck/3.1.2/extra/mpss/etc/init.d/clckd` to `/etc/init.d/clckd`
2. If necessary, edit `/etc/init.d/clckd` and `/etc/init.d/clck-serverd` to change the default user and/or install path.

3. `/sbin/chkconfig --add clckd`
4. `/sbin/service clckd start`

3.6.1.2 RedHat Enterprise Linux* 6 and derivatives

The files corresponding to this Linux* distribution are located in `/opt/intel/clck/3.1.2/extra/rhel6`. Perform the following steps on every compute node.

1. Copy `/opt/intel/clck/3.1.2/extra/rhel6/etc/init.d/clckd` to `/etc/init.d/clckd`
2. If necessary, edit `/etc/init.d/clckd` to change the default user and/or install path.
3. `/sbin/chkconfig --add clckd`
4. `/sbin/service clckd start`

On the head node only, repeat the above steps, but substituting `clck-serverd` for `clckd`.

3.6.1.3 SUSE Linux Enterprise Server* 11 and derivatives

The files corresponding to this Linux* distribution are located in `/opt/intel/clck/3.1.2/extra/suse`. Perform the following steps on every compute node.

1. Copy `/opt/intel/clck/3.1.2/extra/suse/etc/init.d/clckd` to `/etc/init.d/clckd`
2. If necessary, edit `/etc/init.d/clckd` to change the default user and/or install path.
3. `/sbin/chkconfig --add clckd`
4. `/sbin/service clckd start`

On the head node only, repeat the above steps, but substituting `clck-serverd` for `clckd`.

3.6.1.4 `systemd` Based Linux* Distributions

The files for Linux* distributions based on `systemd` are located in `/opt/intel/clck/3.1.2/extra/sysd`. Linux* distributions based on `systemd` include RedHat Enterprise Linux* 7 and SUSE Linux Enterprise Server* 12. Perform the following steps on every compute node.

1. Copy `/opt/intel/clck/3.1.2/extra/sysd/etc/systemd/system/clckd.service` to `/etc/systemd/system/clckd.service`
2. If necessary, edit `/etc/systemd/system/clckd.service` to change the default user and/or install path.

3. `/usr/bin/systemctl enable clckd.service`

4. `/usr/bin/systemctl start clckd.service`

On the head node only, repeat the above steps, but substituting `clck-serverd` for `clckd`.

3.6.2 Discovery and Data Aggregation

By default, `clck-serverd` continuously sends UDP broadcast messages announcing itself as the aggregation point for data. As long as `clckd` is also configured to listen to the same broadcast, all collected data will be sent to `clck-serverd` using the connection information contained in the broadcast. Alternatively, `clckd` can be manually configured with the hostname and port of the `clck-serverd` accumulation service.

Tip: Not all clusters are configured to support UDP broadcast. In this case, set the `<accumulate_host>` option in the configuration file to the hostname of system running `clck-serverd` and set the `<enable_discovery>` option to `false`.

3.6.3 Ad Hoc Clusters

Most data providers collect information from a single node. The collected information pertains to the host on which the data provider was run. However, data providers that collect information such as network bandwidth and latency require multiple nodes. In order to run these cluster data providers in asynchronous mode, an ad hoc cluster needs to be formed.

A broadcast method, similar to the discovery protocol described in section 3.6.2, is used to form ad hoc clusters of two or more nodes. One `clckd` daemon initiates the request for creation of an ad hoc cluster and other, idle `clckd` daemons decide whether to join the ad hoc cluster or not. If an ad hoc cluster of sufficient size can be formed, the cluster data provider is run, following which the ad hoc cluster is dissolved. If an ad hoc cluster could not be formed, then `clckd` will try again later.

Tip: Ad hoc clusters are the source of the requirement in section 3.6.1 that the `clck` user must a cluster user with shared ssh keys. If the ad hoc cluster behavior is not desired, it can be disabled by setting the `<enable_adhoc_clusters>` option in the configuration file to false, and a local user can be used to run `clckd`. On-demand data collection should be used to gather the still necessary cluster data provider information, e.g., `clck-collect -s cluster -f nodefile`.

3.6.4 Performance Impact and Sleep Mode

When `clckd` is idle, the performance impact is negligible. The service will not wake up if the current node load average is above a threshold value. This technique enforces that `clckd` only run data providers when the node is idle.

In addition, `clckd` can be sent a signal that will put the it into a “deep sleep” mode. In this mode, while the `clckd` service is still running, it remains idle regardless of the current load average. No data providers are executed, and this `clckd` instance ignores requests by other `clckd` daemons to form ad hoc clusters. This daemon will remain in “deep sleep” mode until another signal is passed to it.

The `clckd` daemon can be put to sleep using the command `service clckd sleep`.

And similarly, it can be brought out of sleep and made to resume its normal functionality with the command `service clckd wakeup`.

Tip: These commands can be added to cluster resource manager prologue and epilogue scripts to force `clckd` to remain idle while a job is running.

Note that the above operations can only be performed by a privileged user.

3.7 Data Collection Use Cases

This section describes the three most typical data collection use cases, as well as their advantages and disadvantages.

3.7.1 Completely On-Demand Data Collection

In this case, the `clck-collect` program is manually invoked to collect data from the cluster (see section 3.2 for specifics). This provides complete control over the what, where, and when of data collection. On-demand data collection may be invoked by any user, and by default, the resulting database is specific to the user (`$HOME/.clck/3.1.2/clck.db`).

Intel® Cluster Checker 2.x transition note: This use case is the most similar to how version 2 operates.

The primary disadvantage of completely on-demand data collection is that the value of the analysis will be lessened if the data is collected too infrequently. One way to address this is to always collect fresh data prior to running the analysis.

Completely on-demand data collection is the default use case.

3.7.2 On-Demand Data Collection with a Shared Database

On a cluster with multiple users of Intel® Cluster Checker, it is inefficient for each user to have their own private database. One way to setup a common, shared database would be to modify the `<database_file>` setting in `/opt/intel/clck/3.1.2/etc/clckd.xml` to use a file in shared, global writable location.

Tip: In asynchronous mode, or on-demand mode with a shared database, the default database file location should be changed from `$HOME/.clck/3.1.2/clck.db` to a shared location such as `/var/db/clck/clck.db`.

Starting the `clck-serverd` service on the head node (see section 3.6 for specifics) takes this one step further. Data is still collected on-demand using the `clck-collect` program, but this service aggregates the collected data from multiple users and centralizes writing to the database.

Tip: In this case, the `-z` command line option of `clck-collect` must be used. See section 3.2 for more information.

However, the `clck-serverd` service must be initially setup by a privileged user, but can run under any user account.

3.7.3 Completely Asynchronous Data Collection

Tip: While in this use case the primary data collection mechanism is assumed to be the `clckd` service, data can also be collected on-demand, e.g, to update a particular database entry after resolving a issue but before the next asynchronous update has occurred.

The asynchronous data collection use case completely automates the data collection process. This process is also highly scalable since data collection can occur asynchronously. The `clckd` service is continuously running on all the compute nodes. The service periodically wakes up, collects new data and publishes it to the central database, then goes back to sleep. This guarantees a steady stream of fresh data for the analysis phase.

The primary disadvantage of completely asynchronous data collection is that it can potentially interfere with jobs. While the idle `clckd` service load is negligible and the service will defer waking up to collect data if it detects a node is busy, it does not currently communicate with the cluster resource manager to reserve nodes and thus may still impact running jobs. In particular, if a node is allocated to a job after `clckd` has woken up and started a performance benchmark, `clckd` will not be aware of this until after the benchmark has completed. In the meantime, the benchmark will not only interfere with the job, but will also produce invalid benchmark results that may lead to incorrect analysis.

The service must also be initially setup by a privileged user, but can run under any user account.

Tip: The `clckd` service can be placed in “deep sleep” mode to avoid this issue. However, the service must be periodically manually woken up otherwise no data will be collected at all. See section 3.6.4 for more information.

Chapter 4

Analysis

4.1 Overview

Given a populated database (see Chapter 3), Intel® Cluster Checker analyzes the data to identify issues, diagnose the problems, and in some cases, provide recommendations on how to repair the cluster. Invoke the `clck-analyze` program to perform the analysis. The analysis evaluates the collected data using an embedded expert system.

4.2 Running `clck-analyze`

A typical invocation of `clck-analyze` is:

```
clck-analyze -f nodefile
```

Without any command line arguments, all of the nodes in the database are analyzed using default settings. To analyze a subset of nodes, or to assign node roles, a nodefile should be provided (see section 3.3). See `clck-analyze --help` for more information.

The output will contain descriptions of any issues that were identified by the analysis, including node(s) and the severity and confidence percentages.

```
<issue description>
  [ Id: <message identifier> ]
  [ Severity: <percentage>; Confidence: <percentage> ]
  [ N node(s): <node name(s)>]
  [ Database Row Id(s): <database row identifier list> ]
  [ Remedy = <remedy description, if available>]
```

The message id is an unique identifier for the issue type. The id can be used to suppress the issue, if desired (see section 4.8).

Values for severity range from 0 - 100. Higher severity values indicate a higher significance of the identified issues on the cluster. Lower severity levels indicate that the diagnosis is less of a problem.

Values for confidence range from 0 - 100. Higher confidence values indicate a higher level of certainty of the diagnosis. Lower confidence values indicate that the diagnosis is less certain.

The number and names of the nodes affected by the issue are shown. Node names in parentheses indicate that the issue applies to a set of

nodes, e.g., MPI latency between a pair of nodes.

The database row id is a list of database entries containing the raw data that led to the issue. Database row ids are only included when “debug” output is enabled (see section 4.4).

Some issues may recommend a remedy to resolve the issue.

Depending on the cluster configuration, some remedies may not be appropriate. If unsure, review any suggested remedies with an expert before implementing them. Some remedies may require privileged cluster access.

Issues fall into one of three categories: diagnoses, diagnosed signs, and undiagnosed signs.

Diagnoses

Diagnoses describe the root cause of an issue. For example, MPI performance is substandard because some network setting is misconfigured. The typical process to reach a diagnosis is by combining one or more signs, in this example, a sign for substandard MPI performance and another sign for a misconfigured network setting. By default, diagnoses are included in the output.

Diagnosed signs

When a sign has contributed to a diagnosis, it is referred to as a diagnosed sign. By default, diagnosed signs are not displayed in the output. Diagnosed sign output can be enabled using the `-p diagnosed_signs` command line option (see section 4.4).

Undiagnosed signs

Signs that have not contributed to a diagnosis are referred to as undiagnosed signs. For example, a node may be observed to be an outlier with respect to memory bandwidth, but no reason for the substandard performance could be found. By default, undiagnosed signs are included in the output.

Each reported issue should be investigated, and either resolved or suppressed (see section 4.8). Once the issue is resolved, fresh data should be collected and the analysis repeated. When no issues are reported, the cluster has been successfully verified with Intel® Cluster Checker.

4.3 Selecting Nodes

By default, all the nodes contained in the database will be analyzed. If a nodefile is supplied, then the list of nodes contained in the nodefile will be used instead. The nodefile contains a list of cluster node hostnames or IP addresses, one per line. Additional, optional nodefile annotations can also be specified and may alter the analysis (see section 3.3), e.g., some checks only apply to compute nodes and ignore non-compute nodes.

The hostnames in the nodefile must correspond to the hostname values in the database. Partial hostnames in the nodefile will automatically be matched to hostnames in the database if they are identical other than the suffix, e.g., a nodefile containing “node01” would match “node01.cluster” in the database.

`-f` *file* / `--nodefile` *file*

Use the specified file for the list of nodes to analyze, overriding the database.

4.4 Configuration Options

Most of the command line options described in this chapter can also be set in the configuration file. The configuration file variants are not included here, but see the Reference Guide for more information about the configuration file options.

Command line configuration options supported by `clck-analyze` include:

`-c` *file* / `--config` *file*

Use the specified configuration file. The default is `/opt/intel/clck/3.1.2/etc/clck.xml`.

`-d` / `--debug`

Enable "debug" output, i.e., print out the database row ids for each issue (see section 4.2).

Intel® Cluster Checker 2.x transition note: The database entries provide nearly the same information as the "debug" files. For a given database row id, the corresponding database entry can be viewed using the `clckdb` tool, e.g., `clckdb --rowid 17`.

`-D` *file* / `--db` *file*

Use the specified database file. The default is `$HOME/.clck/3.1.2/clck.db`.

`-g` / `--no-color`

`-G` / `--color`

Enable or disable color output. The default is to use color output.

`-o` *file* / `--logfile` *file*

Output will be simultaneously printed to the terminal as well as the specified file.

`-p` *type* / `--print` *type*

`-P` *type* / `--no-print` *type*

Enable or disable output of the specified type. Type is one of `diagnoses`, `diagnosed_signs`, or `undiagnosed_signs`. The options may be specified multiple times to enable or disable more than one output type. The default is to print diagnoses and undiagnosed signs, and not to print diagnosed signs.

Run `clck-analyze --help` for additional command line options.

4.5 Mode Selection

Intel® Cluster Checker can be run in certification mode, compliance mode or health mode.

In health mode, the functionality and uniformity of the cluster is checked. Health mode is the default.

In compliance mode, Intel® Cluster Checker verifies that the requirements specified by the Intel® Cluster Ready specification are met.

Certification mode is effectively the union of compliance and health modes. A successful certification run should not be interpreted as certifying Intel® Cluster Ready compliance as other requirements must also be met.

The mode can be selected on the `clck-analyze` command line using:

`-m` *mode* / `--mode` *mode*

Select the mode. Mode is `certification`, `compliance`, or `health`. The default is health mode.

The Intel® Cluster Ready specification version can be set on the command line using:

`-C` *version* / `--icr` *version*

Specify the Intel® Cluster Ready specification version. This option is only relevant in certification or compliance mode. The only supported version of the Intel® Cluster Ready architecture specification is 1.3.1. The default is 1.3.1.

4.6 Check Selection

Intel® Cluster Checker contains multiple checks. By default, the analysis is performed on "all" set defined in `/opt/intel/clck/3.1.2/etc/sets.xml`. The set of checks can be changed by modifying the configuration file, or by using the following command line options. The list of checks may be found in Appendix B. Alternatively, it is also possible to specify a set from a list of pre-defined sets similar to `clck-collect` (section 3.4). The list of sets can be found in Appendix C.

`-e` *check* / `--exclude` *check*

Exclude the specified check. This option may specified more than once to exclude multiple checks.

`-i` *check* / `--include` *check*

Include the specified check, in addition to the checks defined in `/opt/intel/clck/3.1.2/etc/clck.xml`. This option may be specified more than once to include multiple checks.

`-I` *check* / `--include_only` *check*

Exclusively include the specified check, excluding the checks defined in `/opt/intel/clck/3.1.2/etc/clck.xml`. This option may be specified more than once to exclusively include multiple checks.

Tip: To check a single aspect of a cluster, combine exclusive includes with the ability to collect data using pre-defined sets (section 3.4). E.g., `clck-collect -f nodefile -s foo && clck-analyze -I foo`, where `foo` is the name of the check.

`-s` *set_name* / `--set` *set_name*

Exclusively run the specified set of checks. The sets are definded in `/opt/intel/clck/3.1.2/etc/sets.xml`. This option may be specified more than once to exclusively include multiple sets. This option may also be combined with the options to include or exclude individual checks.

4.7 Filtering

When troubleshooting an issue, it can be useful to filter them so that only the issue at hand is displayed. To efficiently triage issues, a filter based on severity can also be useful.

Several command line options are available to filter the reported issues, including:

`-N` *node* / `--node-exclude` *node*

Exclude all issues for the specified node.

`-n` *node* / `--node-include` *node*

Include only issues for the specified node.

`-x` *severity* / `--min-severity` *severity*

Exclude all issues with a severity level less than the value specified.

4.8 Suppressions

Intel® Cluster Checker 2.x transition note: Suppressions can be used to achieve similar functionality to the `exclude_module` capability.

In some cases, while the diagnosis may be correct, the behavior is actually intended and should not be flagged. Such issues can be suppressed by adding an entry to the configuration file.

The base suppression format is:

```
<configuration>
  ...
  <suppressions>
    <suppress>
      <confidence> </confidence>
      <id> </id>
      <node_id> </node_id>
      <severity> </severity>
    </suppress>
    ...
  </suppressions>
  ...
</configuration>
```

Multiple suppressions may be specified.

`<confidence>` *num* `</confidence>`

Suppress all issues with a confidence level less than the specified value. The default is 0.

`<id>` *string* `</id>`

Suppress all issues matching the specified message id string. The default is empty, meaning suppress all message ids that match the other tags.

`<node_id>` *hostname* `</node_id>`

Suppress all issues corresponding to the specified node. The default is empty, meaning suppress all nodes that match the other tags.

`<severity>` *num* `</severity>`

Suppress all issues with a severity level less than the specified value. The default is 0.

If a tag is omitted, then the default value is used. There is implicit AND logic between tags within a suppression.

The following example will suppress all issues from node4 as well as any issues with message id `example-id` and with a confidence level of less than 50% on any node.

```
<configuration>
  ...
  <suppressions>
    <suppress>
      <node_id>node4</node_id>
    </suppress>
    <suppress>
      <confidence>50</confidence>
      <id>example-id</id>
    </suppress>
  </suppressions>
  ...
</configuration>
```

Appendix A

Database Schema

The database consists of a single table named `clck_1`. The table contains columns described in Table A.1.

Name	SQLite Type	Description
rowid	INTEGER	Unique row ID
row_timestamp	INTEGER	Timestamp when the row was inserted (seconds since the UNIX epoch)
Provider	TEXT	Data provider name
Hostname	TEXT	Hostname of the node where the data provider ran
num_nodes	INTEGER	Number of nodes used by the data provider
node_names	TEXT	Comma separated list of nodes used by the data provider. Empty if num_nodes = 1.
Exit_status	INTEGER	Exit status of the data provider
Timestamp	INTEGER	Timestamp when the data provider started (seconds since the UNIX epoch)
Duration	REAL	Data provider walltime (seconds)
Encoding	INTEGER	Encoding format of the STDOUT and STDERR columns. 0 = no encoding, 1 = base64 encoding.
Stdout_size	INTEGER	Standard output size (number of bytes)
STDOUT	BLOB	Data provider standard output
Stderr_size	INTEGER	Standard error size (number of bytes)
STDERR	BLOB	Data provider standard error
OptionID	TEXT	The ID of the option set with which the provider was run
Version	INTEGER	Output format version of the data provider

Table A.1 – Database schema.

The data definition language definition of the database is:

```
CREATE TABLE clck_1 (
    rowid         INTEGER PRIMARY KEY,
    row_timestamp INTEGER DEFAULT ( strftime( '%s', 'now' )  ),
    Provider      TEXT,
```

```
    Hostname      TEXT,
    num_nodes     INTEGER,
    node_names    TEXT,
    Exit_status   INTEGER,
    Timestamp     INTEGER,
    Duration      REAL,
    Encoding      INTEGER,
    Stdout_size   INTEGER,
    STDOUT        BLOB,
    Stderr_size   INTEGER,
    STDERR        BLOB,
    OptionID      TEXT,
    Version       INTEGER
);
```

The Intel® Cluster Checker database is a standard SQLite* database and any SQLite* compatible tool may be used to browse the database contents. In addition, the `clckdb` utility is provided with Intel® Cluster Checker (see `clckdb -h` for more information).

Appendix B

List of Checks

Check	Description	Default State
all_to_all	IP address consistency	Enabled
cpu	Intel® Cluster Ready CPU compliance	Enabled
datconf	InfiniBand* DAPL configuration	Enabled
dgemm	Floating point performance	Enabled
environment	Environment variables	Enabled
ethernet	Ethernet driver uniformity and wellness	Enabled
heartbeat	Verify data is recent	Enabled
hpl	High Performance Linpack*	Enabled
icr_cluster	Intel® Cluster Ready minimum node count compliance	Enabled
icr_version	Intel® Cluster Ready version compliance	Enabled
imb_pinpgong	MPI performance	Enabled
infiniband	InfiniBand* uniformity and wellness	Enabled
intel_cluster_runtimes	Intel® Cluster Ready runtime library compliance	Enabled
iozone	Disk I/O performance	Enabled
java	Java* uniformity and functionality	Enabled
kernel	Linux* kernel	Enabled
kernel_param	Kernel parameter uniformity	Enabled
libraries	Intel® Cluster Ready runtime library compliance	Enabled
lshw	Hardware uniformity	Enabled
memory	Memory compliance	Enabled
miccheck	Intel® Xeon Phi™ coprocessor uniformity and wellness	Enabled
micinfo	Intel® Xeon Phi™ coprocessor uniformity and wellness	Enabled
mount	Mount point compliance and uniformity	Enabled
mpi_internode	Multi-node MPI functionality	Enabled
mpi_local	Single-node MPI functionality	Enabled
ntp	Clock synchronization	Enabled
offload_phi	Intel® Xeon Phi™ coprocessor functionality	Enabled
opa	Intel® OPA uniformity and wellness	Enabled

Check	Description	Default State
perl	Perl* compliance, uniformity, and functionality	Enabled
process	Process table	Enabled
python	Python* compliance, uniformity, and functionality	Enabled
rpm	RPM uniformity	Enabled
shells	Shell compliance	Enabled
storage	Disk capacity	Enabled
stream	Memory bandwidth performance	Enabled
tcl	Tcl compliance, uniformity, and functionality	Enabled
x11_tools	X11 tool compliance	Enabled

Appendix C

List of Sets

The file `/opt/intel/clck/3.1.2/etc/sets.xml` contains both the analysis sets used by the analyzer and the provider sets used by the collector.

Both the analyzer and the collector support the following thematic sets.

Set	Description
benchmarks	Benchmarks
clock	Clock synchronization
hardware	Hardware uniformity, compliance and wellness
infiniband	InfiniBand* uniformity and wellness
mpi	MPI functionality
tools	Runtime environment compliance, uniformity and functionality
xeon_phi_coproces-sor	Intel® Xeon Phi™ coprocessor uniformity, functionality and wellness

In addition to the above sets, the ”all” set is defined for the analyzer which includes all the checks defined in Appendix B.

The following additional sets are defined for data collection.

Set	Description
cluster	Execute all providers that require a minimum of two nodes to run
single	Execute all providers that run only on a single node
exclude_hpl	Execute all providers with the exception of High Performance Linpack*

Set	Description
Sets for each individual check	Each check listed in Appendix B has a corresponding provider set with the same name. This set collects all data required to perform the respective analysis check