Online CPU Diagnostics Monitor™ Version 2.0 User's Guide

Sun Microsystems, Inc.
www.sun.com

Part No. 819-3208-10
August 2005, Revision A

Send comments about this document to: docfeedback@sun.com

Contents

1. Product Overview 1

What is the Online CPU Diagnostics Monitor? 1

Online CPU Diagnostics Monitor Architecture 2

CDM Components 2

CPU Diagnostics Monitor Daemon 3

CPU Diagnostics Test 3

Configuration File 4

2. Installing the Online CPU Diagnostics Monitor Software 7

CDM Packages 7

Installation Requirements 8

Installing the CDM Software 8

▼ To Install the CDM Software 8

Detecting the Version of the Installation 9

Removing the CDM Software 10

3. Configuring and Using CDM 11

Starting and Stopping CDM 11

Error Handling 12

Log Files 13

Runtime Resource Consumption and Default Scheduling Intervals 13

Scheduling Tests at Specific Times 15

Customizing Tests on Demand at Arbitrary Check Points 16

A. Online Manual Page for `cputst` 19

Name 19

Synopsis 19

Description 19

Options 20

Examples 20

Exit Status 20

Attributes 21

See Also 21

B. Online Manual Page for `cpudiagd` 23

Name 23

Synopsis 23

Description 23

Options 25

Exit Status 25

Environment Variables 25

Files 25

Attributes 26

See Also 26

C. Online Manual Page for `cpudiagd.conf` 29

Name 29

Synopsis 29

Description 29

Environment Variables 31

Examples 32

Files 33

See Also 33

D. `cpudiagd` **Startup Script 35**

E. Legal Notice 37

Tables

TABLE 1-1 `cpudiagd` Command-Line Syntax 3

TABLE 1-2 `cputst` Command-Line Syntax 4

TABLE 2-1 CDM Packages 7

TABLE 3-1 Default Configuration and Resource Consumption 14

TABLE A-1 `cputst` Options 20

TABLE A-2 Attributes for `cputst` 21

TABLE B-1 `cpudiagd` Options 25

TABLE B-2 Files Used by `cpudiagd` 25

TABLE B-3 Attributes for `cpudiagd` 26

TABLE C-1 Files Used by `cpudiagd.conf` 33

Preface

The *Online CPU Diagnostics Monitor Version 2.0 User's Guide* describes how to install, configure, and use the Online CPU Diagnostics Monitor (CDM) software. The primary audience of this manual is advanced system end users.

How This Document Is Organized

This document is organized as follows:

- Chapter 1 provides a general overview of the CDM software.
- Chapter 2 describes how to install and remove the CDM software.
- Chapter 3 describes how to configure and use the CDM software.
- Appendix A provides the online manual page for `cputst`(1M), the online CPU test.
- Appendix B provides the online manual page for `cpudiagd`(1M), the CDM daemon.
- Appendix C provides the online manual page for `cpudiagd.conf`(4), the CDM daemon configuration file.
- Appendix D provides the `/etc/init.d/cpudiagd` startup script.

Using UNIX Commands

This document might not contain information about basic UNIX® commands and procedures such as shutting down the system, booting the system, and configuring devices. Refer to the following for this information:

- Software documentation that you received with your system
- Solaris™ Operating System documentation, which is at:

  http://docs.sun.com

Shell Prompts

Shell	Prompt
C shell	*machine-name*%
C shell superuser	*machine-name*#
Bourne shell and Korn shell	$
Bourne shell and Korn shell superuser	#

Typographic Conventions

Typeface[1]	Meaning	Examples
`AaBbCc123`	The names of commands, files, and directories; on-screen computer output	Edit your `.login` file. Use `ls -a` to list all files. `% You have mail.`
`AaBbCc123`	What you type, when contrasted with on-screen computer output	`% su` `Password:`
AaBbCc123	Book titles, new words or terms, words to be emphasized. Replace command-line variables with real names or values.	Read Chapter 6 in the *User's Guide*. These are called *class* options. You *must* be superuser to do this. To delete a file, type `rm` *filename*.

1. The settings on your browser might differ from these settings.

Documentation, Support, and Training

Sun Function	URL	Description
Documentation	http://www.sun.com/documentation/	Download PDF and HTML documents, and order printed documents
Support and Training	http://www.sun.com/supportraining/	Obtain technical support, download patches, and learn about Sun courses

Sun Welcomes Your Comments

Sun is interested in improving its documentation and welcomes your comments and suggestions. You can submit your comments by going to:

http://www.sun.com/hwdocs/feedback

Please include the title and part number of your document with your feedback:

Online CPU Diagnostics Monitor Version 2.0 User's Guide, part number 819-3208-10

CHAPTER 1

Product Overview

This chapter provides an overview of the Online CPU Diagnostics Monitor software and architecture, and includes the following sections:

- "What is the Online CPU Diagnostics Monitor?" on page 1
- "Online CPU Diagnostics Monitor Architecture" on page 2

What is the Online CPU Diagnostics Monitor?

The Online CPU Diagnostics Monitor (CDM) is an online CPU diagnostic program for platforms based on the UltraSPARC® III and UltraSPARC-IV families of processors. CDM continuously verifies proper functioning of processors in the system. CDM provides additional high reliability to systems by detecting and taking action on detected CPUs.

CDM is designed to run on production systems. Because CDM provides high reliability against CPU failures, it is highly recommended to install CDM on production systems.

Online CPU Diagnostics Monitor Architecture

The following diagram depicts the general architecture of CDM.

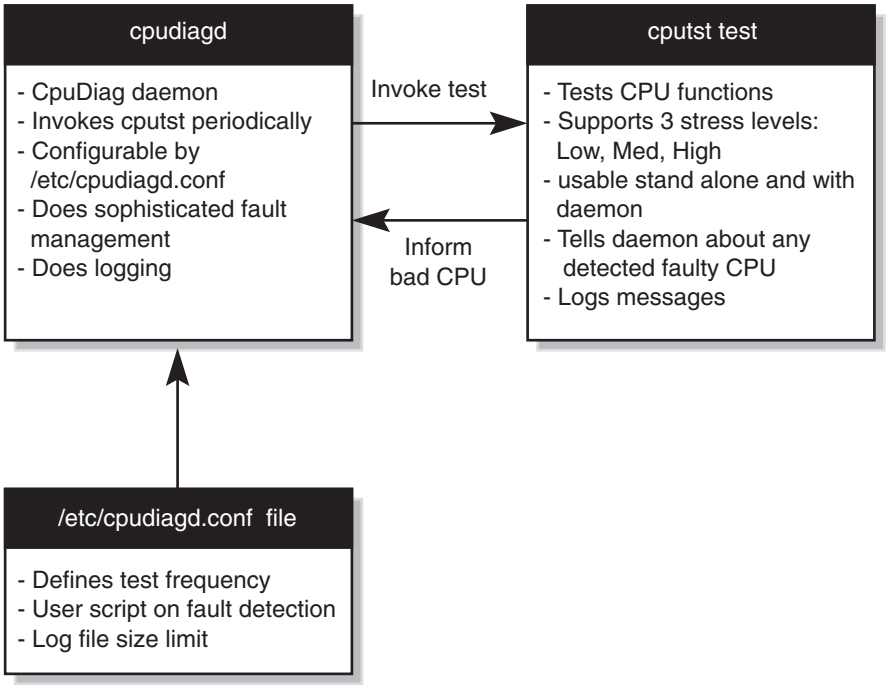

FIGURE 1-1 CDM General Architecture

CDM performs CPU diagnostics periodically on the system and takes the appropriate actions such as offlining the detected CPU and logging error messages. The primary components of CDM consist of a CPU diagnostics test, a daemon to schedule the test, and a user configuration file.

CDM Components

The main components and interfaces of CDM are:

- `cputst` – CPU diagnostics test
- `cpudiagd` – CPU diagnostics monitor daemon
- `/etc/cpudiagd.conf` – CPU diagnostics monitor configuration file

CPU Diagnostics Monitor Daemon

The CPU Diagnostics monitor daemon `/usr/lib/sparcv9/cpudiagd` is started from a system startup script during the boot sequence. The daemon reads the configuration file on bringup and is responsible for scheduling tests and performing fault management based on test results.

The `cpudiagd` daemon invocation syntax is as follows:

```
# /usr/lib/sparcv9/cpudiagd [-vdi]
```

TABLE 1-1 `cpudiagd` Command-Line Syntax

Option	Description
`-v`	Verbose Mode. Prints verbose messages to `stdout`. Specifying verbose mode also places the daemon in non-daemon mode.
`-d`	Invokes the daemon to run in the foreground and operate as non-daemon.
`-i`	Performs initial testing during system boot before becoming a daemon.

`cpudiagd` Return Exit Code

`cpudiagd` exits with error code 0 on success and 1 on error.

CPU Diagnostics Test

The CPU diagnostics test `/usr/platform/sun4u/sbin/sparcv9+vis2/cputst` is provided for all platforms supporting `sparcv9+vis2` instruction architecture.

The test supports options to test at three levels of varying stress: low, medium and high. The test invocation syntax is as follows:

```
# cputst  [ -s 1|2|3 ] [-d dev_id] [-vnh]
```

TABLE 1-2 `cputst` Command-Line Syntax

Option	Description
`-s 1|2|3`	Specifies one of the following stress level: • 1 – Perform low stress testing • 2 – Perform medium stress testing • 3 – Perform high stress testing By default, medium stress level testing is performed.
`-d dev_id`	Specifies the processor ID to be tested. If this option is not specified, all CPUs are tested sequentially.
`-v`	Prints verbose messages to `stdout`.
`-n`	Notifies the `cpudiagd` daemon on fault. If this option is not specified, the errors are printed to `stderr`, and no information about detected CPUs is communicated to the `cpudiagd` daemon.
`-h`	Displays help message.

`cputst` Return Exit Code

`cputst` exits with one of the following exit codes:

- 0 – Indicates successful execution—the CPU is functioning properly.
- 1 – At least one CPU failed.
- 2 – Internal software error. (`malloc` failure, etc.)
- 3 – Command-line usage error

Configuration File

The configuration file `/etc/cpudiagd.conf` can be used to configure CDM. This file is read by the `cpudiagd` daemon on startup. You can force reconfiguration after updating the configuration file by sending `SIGHUP` to the daemon.

For a detailed description of the parameters supported by this configuration file, see Appendix C or the online manual page for `cpudiagd.conf`(4). The supported parameters in the configuration files are:

- `CPU_TEST_FREQ_MIN_STRESS=DEFAULT|[0-9]+[smh]`
- `CPU_TEST_FREQ_MED_STRESS=DEFAULT|[0-9]+[smh]`
- `CPU_TEST_FREQ_HIGH_STRESS=DEFAULT|[0-9]+[smh]`

The above parameters can be used to schedule `cputst` at different stress levels: low, medium and high respectively. The suffixes `s`, `m`, and `h` are used for seconds, minutes, and hours respectively. The value of `0` disables testing at the specified stress level. By default, the tests are scheduled at the system-defined scheduling intervals. (See TABLE 3-1 on page 14 for the system-defined scheduling intervals.)

CPU_ON_FLT_EXEC=command_line

This parameter is used to specify a user-provided script/binary that will be executed after detecting a potentially faulty CPU.

The detected processor ID is passed to the script by setting the environment variable `CPU_ID_FAILED` to the decimal value of the processor ID.

LOG_MAX_NUM_BACKUPS=[0-9]+

This parameter specifies the number of maximum backup logs that need to be maintained for CDM-specific information and error logs. The minimum value is `1` and the maximum value is `100`. By default, only one backup log is maintained.

LOG_MAX_SIZE=[0-9]+

This parameter is the maximum log size in Kbytes. The minimum value is `1` and the maximum value is `1000000`, which means the minimum log size is 1 Kbyte and the maximum size is 1 Gbyte. The default value is 1000 which means 1 Mbyte.

LOG_ENABLE_INFO_STATS = yes/no

This parameter enables/disables the logging of test execution statistics information. By default, it is enabled (which is equivalent to specifying yes). However if power management is enabled, the test execution informational logging is disabled by default.

Specifying `LOG_ENABLE_INFO_STATS=no` will disable this parameter.

Note – All blank lines are ignored. All lines for which the first nonwhite character is a pound sign (#) are treated as comments.

CHAPTER 2

Installing the Online CPU Diagnostics Monitor Software

This chapter describes how to install and remove the Online CPU Diagnostics Monitor (CDM) software, and includes the following sections:

- "CDM Packages" on page 7
- "Installation Requirements" on page 8
- "Installing the CDM Software" on page 8
- "Detecting the Version of the Installation" on page 9
- "Removing the CDM Software" on page 10

CDM Packages

The following is a list of the CDM packages:

TABLE 2-1 CDM Packages

Package Name	Description
`SUNWcdiar`	CDM (root) package that contains the startup scripts and the default configuration file (`cpudiagd.conf`).
`SUNWcdiax`	CDM 64-bit package that contains the CPU diagnostics test, `cputst` and the CDM daemon, `cpudiagd`.
`SUNWcdiam`	CDM package that contains the online manual pages for `cputst`(1M), `cpudiagd`(1M) and `cpudiagd.conf`(4).

Installation Requirements

The CDM software is supported on SPARC 64-bit only and is not compatible with 32-bit platforms. The hardware platform must support UltraSPARC III or UltraSPARC IV families of processors and must support the `sparcv9+vis2` instruction set. A Solaris 8, 9, or 10 Operating System must be installed (with core system support software group `SUNWCreq` at a minimum) before installing CDM. The operating system must support SPARC 64-bit. To verify that CDM is supported on your hardware platform, verify that the following command output includes `sparcv9+vis2`.

```
# /usr/bin/isalist
sparcv9+vis2 ...
```

You must be superuser to install the CDM software. There must be at least 2 Mbytes of available disk space in both root (`/`) and `/usr` partitions. There must be at least 1 Mbyte available disk space in the `/var` partition.

Installing the CDM Software

There are several utilities available for installing packages. This section describes how to use the `pkgadd` utility to install CDM from the file system directory containing the CDM packages:

▼ To Install the CDM Software

1. **Become superuser.**
2. **Use the `pkgadd` command to install the packages in the following order:**

```
# pkgadd -d path_to_packages_directory SUNWcdiax SUNWcdiar SUNWcdiam
```

Note – `SUNWcdiar` depends on `SUNWcdiax`—thus, the packages must be installed in the above order.

Note – Installing CDM software also starts the online CPU diagnostics monitoring by starting the `cpudiagd` daemon.

Detecting the Version of the Installation

To verify the version of the installed software at any time, use one of the following methods.

- **To individually examine the version of cputst and cpudiagd binaries:**

```
# /usr/platform/sun4u/sbin/sparcv9+vis2/cputst -v
```

The test verbose messages displayed should indicate the `cputst` version number as 2.0 if the latest version is installed.

The `cpudiagd` binary version number can be detected by examining the informational log file:

```
# tail /var/cpudiag/log/info.log
```

The entries in the informational log message file contain the version number of the installed software. The last message entry by `cpudiagd` should contain version 2.0 if the latest version is installed.

- **To use the `pkginfo` command to indicate the correct installed version of the packages (2.0 is the latest):**

```
# /usr/bin/pkginfo -l SUNWcdiax SUNWcdiar SUNWcdiam
```

Removing the CDM Software

Use the `pkgrm` command to remove the installed CDM packages in the following order:

```
# pkgrm SUNWcdiam SUNWcdiar SUNWcdiax
```

CHAPTER 3

Configuring and Using CDM

This chapter describes how to configure and use the CDM software, and includes the following sections:

- "Starting and Stopping CDM" on page 11
- "Error Handling" on page 12
- "Log Files" on page 13
- "Runtime Resource Consumption and Default Scheduling Intervals" on page 13

Starting and Stopping CDM

The `cpudiagd` daemon is automatically started during system startup from the `/etc/rc2.d/S22cpudiagd` startup script. This link is a hard link to the `/etc/init.d/cpudiagd` script.

See Appendix D for the contents of `/etc/init.d/cpudiagd` script.

To manually start CDM directly, type the following as superuser:

```
# /usr/lib/sparcv9/cpudiagd
```

To perform `cpudiagd` in verbose mode and in the foreground, type the following as superuser:

```
# /usr/lib/sparcv9/cpudiagd -v
```

Note – The `/etc/init.d/cpudiagd` file is used as a system startup script, not the binary.

During system startup, the `/usr/lib/sparcv9/cpudiagd` daemon is invoked with the `-i` option which performs initial low stress CPU testing on all CPUs in parallel before becoming the daemon.

To stop running the `cpudiagd` daemon and the `cputst` processes, type the following as superuser:

```
# /etc/init.d/cpudiagd stop
```

To stop only the daemon, type the following as superuser:

```
# pkill -x cpudiagd
```

To terminate only the `cputst`, type the following as superuser:

```
# pkill -x cputst
```

Error Handling

When a CPU is detected by the `cputst`, it communicates the information about the detected CPU to the daemon. Then the following sequence of operations take place:

1. The daemon logs an error message about the detected fault using the `syslog`(3C) mechanism which logs the error message into the `/var/adm/messages` file by default. In addition, the CDM-specific error log is updated.
2. The daemon creates a detected CPU history file (`/var/cpudiag/data/bad_cpu_id.X` [`X` is the decimal processor ID]). This file is used by the daemon to recognize the detected CPU as a suspected faulty CPU across reboots until this file is manually deleted by the system administrator after the faulty CPU is replaced.
3. The CDM software performs an initial attempt to offline the detected faulty CPU. The offline attempt fails if any process is bound to the faulty CPU. The offline attempt always fails on a single processor system.
4. If the user has provided a binary/script to be run when a fault is detected, it is invoked. This script could, for example, notify the system administrator about the fault and shut down any user applications that may be explicitly bound to the faulty CPU.

See Appendix C or the online manual pages for `cpudiagd.conf`(4) for more information on specifying the command line to be executed on fault detection.

5. The CDM software attempts to offline the faulty CPU again. This reattempt to offline is likely to succeed if the binary/script has shut down the user applications and all processes bound to the faulty CPU were terminated.
6. If the offline reattempt fails and if the detected CPU is still online, an appropriate error message is logged.
7. Whenever the machine is rebooted, the CDM software invokes the daemon with the `-i` option from the startup script. If there is a suspected faulty CPU indicated by a detected CPU history file (`/var/cpudiag/data/bad_cpu_id.X` [X is the processor ID]) as created from Step 2, the CDM software performs `cputst` with the high stress mode to test the CPU.

   If the CPU is found faulty, the CDM software performs Step 2 to Step 6.

Log Files

CDM errors are logged using the `syslog` mechanism (see the online manual page for `syslog`(3C)) which logs the messages in the `/var/adm/messages` file by default. In addition, the errors are logged in a CDM-specific error log file, `/var/cpudiag/log/error.log`. The informational messages such as test execution start, end, and elapsed time statistics are logged in a CDM-specific information log file, `/var/cpudiag/log/info.log`.

The growth size of the log files and the number of additional backup logs are configurable with the configuration file parameters. See Appendix C or the online manual page for `cpudiagd.conf`(4) for more information.

Runtime Resource Consumption and Default Scheduling Intervals

CPU testing (`cputst`) can be invoked with three levels of stress (low, med, or high). The operations performed in all three modes are the same. The difference between low, med, and high stress is defined by how many times an operation or set of operations is performed.

`cputst` performs complex mathematical operations, like operations with matrices. The stress level is also reflected in the size of the matrices with which `cputst` operates.

Low stress tests provide functional coverage of the floating point and caches in the CPU. High stress tests simulate a floating point intensive application.

There is a better chance to catch errors in the high stress mode, than in the low stress mode. However, the run time increases with the stress level.

The approximate memory resources consumed by the `cputst` and approximate runtimes are listed in TABLE 3-1. The runtimes vary greatly depending on the hardware platform and the system load. The figures in TABLE 3-1 are approximate and were taken from a system with 1200 MHz UltraSPARC III family of processors with no user applications running at the same time.

TABLE 3-1 Default Configuration and Resource Consumption

Stress Level	Memory	Run Time	Test Interval
Low	4 MB	100 milliseconds/CPU	Disabled
Medium	8 MB	1.2 seconds/CPU	15 minutes
High	130 MB	90 seconds/CPU	Disabled

When using the default configuration, the expected performance impact of CDM is either negligible or approximately 0.1% of 24 hour CPU usage for most applications measured in a steady state.

The `cputst` invoked by the daemon during normal operation tests all CPUs in the system sequentially. Testing at different stress levels is scheduled independently – thus, they could partially overlap. Even if a very low test interval is configured, the daemon does not perform a new `cputst` invocation until the previous invocation completes the testing on the same stress level. For example, if the high stress testing frequency is specified as 5 seconds, the invocation of a new `cputst` high stress testing would still be delayed until the previous invocation is completed.

The CPU usage of the CPU that is under testing is close to 100% during the brief time of the test run. The average CPU usage of the testing with default parameters have been measured to be less than 0.2% on systems based on the UltraSPARC III family of processors.

Note – Some SPARC desktop models satisfy the United States Environmental Protection Agencys ENERGY STAR® Memorandum of Understanding #3 guidelines. On these systems, by default, power management is enabled and the online CPU testing will be done only when the CPUs are not in power save mode. However, if the `/etc/cpudiagd.conf` file is explicitly modified to specify test intervals, the testing is done exactly as configured. See the online manual page for `power.conf`(4) for more information about power management.

Scheduling Tests at Specific Times

In most cases, changing the `/etc/cpudiagd.conf` file is sufficient for changing scheduling parameters. However, if you choose to schedule testing only during specific times of the day, this can be achieved by disabling the testing in the `cpudiagd.conf` entry, and making a `crontab`(1) entry to invoke the test.

For example, to invoke the `cputst` program in high stress mode at 1:00 AM daily, the following entry needs to be made using `crontab`(1): (See the `crontab`(1) online manual page)

```
0 1 * * * /usr/platform/sun4u/sbin/sparcv9+vis2/cputst -s 3 -n
```

Note – Because the `cputst` program is invoked with the `-n` option, automatic notification is sent to the `cpudiagd` daemon, which then takes appropriate action if any CPU failures are detected.

To disable medium stress testing invoked by the daemon, add the following entry to the `/etc/cpudiagd.conf` file:

```
CPU_TEST_FREQ_MED_STRESS=0
```

See the `cpudiagd.conf`(4) online manual page for more details.

Customizing Tests on Demand at Arbitrary Check Points

In some customized environments, performing some quick testing on demand could be desirable. Such quick testing can be done using an administrative script to run tests on all CPUs in parallel.

See the following examples:

Example 1:

In addition to the active CDM running periodic testing in the background, the following script may be executed by an application at certain definite points to verify the reliability of the CPU at that instant:

```
#!/bin/sh
#
# Example 1: Perform medium stress test in parallel on all CPUs.
#
for cpu in /usr/sbin/psrinfo | grep on-line | cut -f1
do
/usr/platform/sun4u/sbin/sparcv9+vis2/cputst -s 2 -d $cpu -n &
done
wait
# End of the script
```

Note – Since `cputst` is invoked with the `-n` option, the daemon is notified of any fault.

Note – The test is invoked in parallel on all CPUs in the background. By default, when the `-d` option is not used, the testing is done sequentially over one CPU at a time, which can take a significantly longer time.

Note – The `cpudiagd` daemon is capable of handling multiple simultaneous fault notifications from different instances of the `cputst` processes. Thus, even if multiple `cputst` processes are running at the same time (some invoked by `cpudiagd` and some from another script), this is properly handled by `cpudiagd` daemon.

Example 2:

Completely disable `cputst` invocations from the `cpudiagd` daemon and let other administrative scripts do testing at different points of time on demand.

To disable test invocations, specify interval 0 in `/etc/cpudiagd.conf` for all low, medium, and high stress test frequencies. (See the `cpudiagd.conf`(4) online manual page.)

Develop the custom script to do testing similar to the script in Example 1.

Example 3:

Completely disable running CDM – perform testing and fault management by administrative scripts.

The `/etc/rc2.d/S22cpudiagd` script can be modified to disable starting the `/usr/lib/sparcv9/cpudiagd` daemon. This will completely disable CDM.

A script similar to the following can be used for testing and fault management without making use of the `cpudiagd` daemon for fault management:

```
#!/bin/sh
CPUTST=/usr/platform/sun4u/sbin/sparcv9+vis2/cputst
for cpu in /usr/sbin/psrinfo | grep on-line | cut -f1
do
( $CPUTST -s 2 -d $cpu >> /var/adm/cpuerrs.log 2>&1 ;
if [ $? -eq 1 ]; then
(date; echo Error: CPU $cpu is found faulty ) >>
/var/adm/cpuerrs.log
# Include application specific fault management actions here.
# Do: "psradm -f $cpu" to take faulty cpu offline, etc, if desired
fi
) &
done
wait
# End of script
```

Note – `cputst` is invoked without the `-n` option, so the `cpudiagd` daemon does not need to run and the daemon is not notified of CPU failures. When the `-n` option is not specified, the `/var/cpudiag/data/bad_cpu_id.X` (where *X* = `processor_id`) file is not created after a fault detection because it is done only by the daemon.

Note – The test exit status 1 is used to detect CPU failure.

Example 4:

Perform medium stress testing from a script continuously with five second intervals between test completion and a new invocation. Do testing sequentially on one CPU at a time to minimize any performance impact during this time. Also, disable test invocations by the CDM daemon.

Disable test invocation by the daemon by modifying `/etc/cpudiagd.conf` to specify corresponding test intervals as `0`.

Use a script similar to the following:

```
#!/bin/sh
while true;
do
/usr/platform/sun4u/sbin/sparcv9+vis2/cputst -s 2 -n
>>/var/adm/cpuerrs.log
sleep 5
done
# End of script
```

Note – `cputst` is invoked with the `-n` option, which requires the `cpudiagd` daemon to be running.

Note – Because no `-d` option is specified, the testing is done sequentially over all CPUs, one CPU at a time.

APPENDIX **A**

Online Manual Page for `cputst`

This appendix provides the online manual page for `cputst`.

Name

`cputst` - CPU Test for Online CPU Diagnostics Monitor

Synopsis

```
/usr/platform/sun4u/sbin/sparcv9+vis2/cputst [-vnh] [ -s
<stress_level> ] [ -d <dev_id> ]
```

Description

`cputst` primarily exercises and tests the Floating Point Unit (FPU) on Sun platforms supporting a minimum of UltraSPARC III family of processors. The FPU functionality is checked by performing FPU-oriented operations with single and double precision numbers. The computed results are verified against known good results.

Options

TABLE A-1 `cputst` Options

Option	Description
`-s <stress_level>`	1=low, 2=medium, 3=high. The default is 2.
`-d <dev_id>`	Specify the `processor_id` of the CPU to be tested. By default, all CPUs in the system are tested.
`-v`	Verbose Mode – the default is Verbose off.
`-n`	Notify `cpudiagd`(1M) for subsequent fault management. The default is no notification.
`-h`	Display help message.

Examples

Example 1: Using `cputst` to test all the CPUs with medium stress and without notification.

```
# ./cputst
```

Example 2: Using `cputst` to test CPU 20 with low stress, without notification, and with verbose messages.

```
# ./cputst -s 1 -v -d 20
```

Example 3: Using `cputst` to test all the CPUs, with medium stress, and with notification.

```
# ./cputst -n
```

Exit Status

- 0 – No failures or errors are detected in the system.
- 1 – Failures or errors are detected in the system.
- 2 – An internal `cputst` error occurred.

Attributes

See `attributes`(5) for descriptions of the following attributes:

TABLE A-2 Attributes for `cputst`

Attribute Type	Attribute Value
Availability	`SUNWcdiax`
Interface Stability	Evolving

See Also

`cpudiagd`(1M), `cpudiagd.conf`(4)

APPENDIX **B**

Online Manual Page for `cpudiagd`

This appendix provides the online manual page for `cpudiagd`(1M).

Name

`cpudiagd` - Online CPU Diagnostics Monitor daemon

Synopsis

```
/usr/lib/sparcv9/cpudiagd [ -vdi ]
```

Description

The Online CPU Diagnostics Monitor daemon runs in the background and schedules periodic executions of the CPU diagnostics test `cputst`(1M) to monitor CPUs in the system to provide high reliability. If a CPU which is not functioning properly is detected, it is immediately taken offline if possible.

The daemon is started from a system startup script. It reads the `/etc/cpudiagd.conf` configuration file on startup. Users can send a `SIGHUP` signal to the daemon to force reconfiguration after updating the configuration file. For the description of the configuration file, see `cpudiagd.conf`(4).

The daemon schedules the CPU diagnostics test `cputst`(1M) periodically on system-defined default intervals. The frequency of scheduling the test can also be explicitly configured using the `cpudiagd.conf` file.

The `cputst` communicates information about a detected CPU to the daemon. On detecting the fault, the daemon attempts to offline the detected CPU immediately. Then the daemon creates a detected CPU history data file (explained later).

Then the daemon executes any user configured CPU fault action executable, if specified in the `cpudiagd.conf`(4) configuration file.

After executing the fault action executable, the daemon reattempts to offline the detected CPU again if it is not already offline.

One of the primary reasons for not being able to take the detected CPU offline is that processes are bound to the CPU. To force successful CPU offline in such cases, the user fault action script could kill/unbind any processes bound to the detected CPU, if that is desired. The environment variable `CPU_ID_FAILED` is used to export the detected processor ID to the user executable. See the online manual pages for `cpudiagd.conf`(4) for more information.

Information about detected CPU is maintained in a detected CPU history data file `/var/cpudiag/data/bad_cpu_id.`*X* where *X* = processor ID of the detected CPU. This can be easily processed by any other script to recognize the detected CPU processor ID. The contents of the detected CPU history file includes information about the timestamp of the detection on that CPU.

The daemon recognizes the existence of the file and assumes the CPU as indicated by the suffix of the file is a suspected detected component. It runs high stress testing during system startup (when invoked with `-i flag`) on detected CPUs. If any system monitoring tool is to consume this data file, then the format of the contents should not be assumed to be stable.

After replacing detected CPU, user should manually remove this file to prevent additional system startup delay of around 1.5 to 2 minutes due to testing on any detected CPU. If any detected CPU history data file is left over, then appropriate warnings are displayed and logged during startup of the daemon.

All critical errors such as those related to detection and offline of detected CPUs are logged using `syslog`(3C) as well as in `/var/cpudiag/log/error.log` file. The informational messages such as test execution statistics are maintained in the `/var/cpudiag/log/info.log` file. The maximum growth size of the log files and the number of additional backup logs are user-configurable using `cpudiagd.conf` file entries.

In the event of a dynamic reconfiguration operation to remove any CPU board, `cpudiagd` temporarily suspends scheduling of any CPU testing until the operation is complete. `cpudiagd` is notified of CPU board dynamic reconfiguration events by the `rcmscript`(4) plug-in script `/usr/platform/sun4u/lib/rcm/scripts/SUNW,cpudiagd`

Options

The following options are supported:

TABLE B-1 `cpudiagd` Options

Option	Description
`-v`	Prints Verbose messages to `stdout`. Also implies the `-d` option.
`-d`	Runs in non-daemon mode; used for debugging.
`-i`	Invoked during early system startup. It performs minimum initial testing before becoming daemon. Use of this option is discouraged unless started from early system startup sequence from `rc` script.

Exit Status

`cpudiagd` exits with `0` on success and exits with `1` on failure.

Environment Variables

CPU_ID_FAILED

The processor ID of the detected CPU. This environment variable is exported to the user script/binary that is specified to be executed on fault detection.

Files

`cpudiagd` uses the following files:

TABLE B-2 Files Used by `cpudiagd`

File	Description
`/etc/cpudiagd.conf`	User configuration file
`/var/cpudiag/log`	Log files directory
`/var/cpudiag/log/error.log`	Log file containing Error Messages
`/var/cpudiag/log/info.log`	Log file containing Informational Messages

TABLE B-2 Files Used by `cpudiagd` *(Continued)*

File	Description
`/var/cpudiag/log/error.log.0`, `/var/cpudiag/log/error.log.1`, etc.	Backup error logs
`/var/cpudiag/log/info.log.0`, `/var/cpudiag/log/info.log.1`, etc.	Backup information logs
`/var/cpudiag/data`	Data files directory
`/var/cpudiag/data/bad_cpu_id.`*X*	Detected CPU history data file where *X* = processor ID of the detected CPU
`/etc/init.d/cpudiagd`	Start/stop script
`/var/run/cpudiagd_door`	Door file used for communication with `cputst`(1M)
`/var/run/cpudiagd.pid`	Contains PID of the daemon
`/usr/platform/sun4u/lib/rcm/scripts/SUNW,cpudiagd`	`rcmscript`(4) plug-in script for `cpudiagd`

Attributes

See `attributes`(5) for descriptions of the following attributes:

TABLE B-3 Attributes for `cpudiagd`

Attribute	Description
Availability	`SUNWcdiax`
Interface Stability	Evolving

See Also

`cputst`(1M), `cpudiagd.conf`(4)

Notes

Some SPARC desktop models satisfy the United States Environmental Protection Agencys ENERGY STAR(R) Memorandum of Understanding #3 guidelines. On these systems, by default, power management is enabled and the online CPU testing will be done only when the CPUs are not in power save mode. However, if the `/etc/cpudiagd.conf` file is explicitly modified to specify test intervals, the testing is done exactly as configured. See man `power.conf`(4) for more information about power management.

APPENDIX **C**

Online Manual Page for `cpudiagd.conf`

This appendix provides the online manual page for `cpudiagd.conf`(4)

File format – `cpudiagd.conf`(4)

Name

`cpudiagd.conf` - configuration file for `cpudiagd`(1M)

Synopsis

```
/etc/cpudiagd.conf
```

Description

The file `/etc/cpudiagd.conf` contains information used by the online CPU diagnostics daemon, `cpudiagd`(1M). The daemon runs CPU diagnostics test `cputst`(1M) in the background periodically. The daemon reads this file during startup. It also rereads this file if it receives a `SIGHUP` signal.

Each parameter entry in the configuration file is of the form Name=Value. The following entries are supported in the configuration file:

- `CPU_TEST_FREQ_MIN_STRESS=DEFAULT|[0-9]+[smh]`
- `CPU_TEST_FREQ_MED_STRESS=DEFAULT|[0-9]+[smh]`
- `CPU_TEST_FREQ_HIGH_STRESS=DEFAULT|[0-9]+[smh]`

These parameters specify the time interval on which CPU diagnostics test `cputst`(1M) is scheduled. The test can run at three stress levels low, medium, and high. The time interval for scheduling tests at each of the low, medium and high levels can be configured independently by using the above three parameters respectively.

The value for the previous parameters could either be the string `DEFAULT` or a non-negative integer followed by one of the letters `s`, `m` or h which specifies seconds, minutes, and hours respectively. For example, `30s` specifies 30 seconds and `15m` specifies 15 minutes and `12h` specifies 12 hours.

If the value is specified as DEFAULT, the system schedules the tests at the system defined default intervals.

If the value is specified as `0`, the invocation of the test at the corresponding stress level is disabled.

CPU_ON_FLT_EXEC=command_line

This parameter specifies the user configurable script/binary that should be executed after detecting a detected CPU. The command can have optional command line options specified at the end.

After detecting a CPU that is not functioning properly, an initial offline attempt of the CPU is performed and then the user specified command is invoked irrespective of whether the initial offline attempt had succeeded or not. After executing the user specified executable, one more offline attempt of the CPU is be performed if the CPU has not already been offlined.

The detected processor ID is passed to the script by setting environment variable `CPU_ID_FAILED` to the decimal value of the processor ID.

The command line specified should be the absolute executable path name with optional command line arguments.

The user-provided script can be used to notify the system administrator of the detected CPU. The user-provided script can also enable shutting down the user applications to potentially make the subsequent CPU offline attempt by `cpudiagd` daemon more likely to succeed. Note that the primary reason for a CPU offline attempt to fail is that processes are bound to the detected CPU.

LOG_MAX_NUM_BACKUPS=[0-9]+

This parameter specifies a number of maximum backup logs that need to be maintained for CPU diagnostics monitor specific information and error logs. The minimum value is 1 and maximum value is 100. By default only one backup log is maintained.

LOG_MAX_SIZE=[0-9]+

This parameter specifies the maximum log size in Kbytes. The minimum value is `1` and maximum value is `1000000`, which means the minimum log size is 1 Kbyte and the maximum size is 1 Gbyte. The default value is 1000 which means 1 Mbyte.

LOG_ENABLE_INFO_STATS=yes/no

This parameter enables/disables the logging of test execution statistics information. By default, it is enabled (which is equivalent to specifying "`yes`"). However if power management is enabled, the test execution informational logging is disabled by default.

Specifying "`LOG_ENABLE_INFO_STATS=no`" will disable this feature.

Note – Blank lines in the `cpudiagd.conf` file are ignored. Lines for which the first nonwhite character is a pound sign (#) are treated as comments.

Environment Variables

CPU_ID_FAILED

The processor ID of the detected detected CPU. This environment variable is exported to the user script/binary that is specified to be executed on fault detection. See the description of `CPU_ON_FLT_EXEC` parameter.

Examples

Example 1: A Sample `cpudiagd.conf` Configuration File

```
# Example configuration file start.
CPU_TEST_FREQ_MIN_STRESS=30s
CPU_TEST_FREQ_MED_STRESS=15m
CPU_TEST_FREQ_HIGH_STRESS=6h
CPU_ON_FLT_EXEC=/home/admin/bin/cpuflt.sh
# end of example configuration file.
```

The above configuration file specifies that the `cputst`(1M) should be scheduled to execute at minimum stress level once in 30 seconds, at medium stress level once at 15 minutes, at high stress level once in 6 hours. This configuration file also specifies that the script `/home/admin/bin/cpuflt.sh` should be executed on detecting a CPU that is not functioning properly.

Example 2: Another Sample `cpudiagd.conf` Configuration File

```
# Example configuration file start.
    CPU_ON_FLT_EXEC=/home/admin/bin/killprocs.sh
# end of example configuration file.
```

Example `killprocs.sh` script:

```
#!/bin/sh
    #

    if [ "`psrinfo -s $CPU_ID_FAILED`" != "1" ]; then
      exit 0     # bad cpu is not online now.
    fi

    # The bad CPU is still online.
    # kill all processes bound to any CPU (not only the faulty one).
    # (not necessary to kill all bound processes, but example works).
    BOUND_PIDS=`/usr/sbin/pbind -q |/usr/bin/tr ':' ' ' |
                /usr/bin/awk '{print $3;}'`

    kill -9 $BOUND_PIDS
    /usr/sbin/psradm -f $CPU_ID_FAILED  # offline the faulty CPU now.

    # script end.
```

The above script would kill all the bound processes. The pbind(1M) command can be used to unbind the bound processes if unbinding is desired over killing.

Files

cpudiagd.conf(4) uses the following files:

TABLE C-1 Files Used by cpudiagd.conf

File	Description
/etc/cpudiagd.conf	cpudiagd configuration file
/var/cpudiag/log	Log files directory
/var/cpudiag/log/error.log	Error log file
/var/cpudiag/log/info.log	Information log file
error.log.0, error.log.1, etc.	Backup error logs
info.log.0, info.log.1, etc.	Backup information logs

See Also

cpudiagd(1M), cputst(1M)

APPENDIX **D**

`cpudiagd` Startup Script

This appendix provides the `/etc/init.d/cpudiagd` startup script.

```
#!/sbin/sh
#
#
# Copyright 2003 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#
#ident  "@(#)init.cpudiagd  1.10  05/06/06  SMI"

isalist=`/usr/bin/isalist`
DAEMON=

#
# Automatically enabled on sparcv9+vis2 arch (supported by UltraSparc-III )
#

case "$isalist" in
        *sparcv9+vis2*)
                DAEMON=/usr/lib/sparcv9/cpudiagd
                ;;
        *)
                DAEMON=
                ;;
esac

case "$1" in
'start')
    # not supported on local zones
    if [ -x /sbin/zonename ]; then
      if [ "`/sbin/zonename`" != "global" ];  then
    exit 0
      fi
    fi
```

```
    if [ -x "$DAEMON" ];
    then
    echo 'Starting cpudiagd ... \c'
       $DAEMON -i
    echo 'done.'
    fi
    ;;

'stop')
     /usr/bin/pkill -x -u 0 '(cputst|cpudiagd)'
    ;;
*)
    echo "Usage: $0 { start | stop }"
    exit 1
    ;;
esac
exit 0
```

APPENDIX **E**

Legal Notice

vigueur dans d'autres pays dans le domaine des exportations et importations. Les utilisations, ou utilisateurs finaux, pour des armes nucléaires,des missiles, des armes biologiques et chimiques ou du nucléaire maritime, directement ou indirectement, sont strictement interdites. Les exportations ou réexportations vers les pays sous embargo américain, ou vers des entités figurant sur les listes d'exclusion d'exportation américaines, y compris, mais de manière non exhaustive, la liste de personnes qui font objet d'un ordre de ne pas participer, d'une façon directe ou indirecte, aux exportations des produits ou des services qui sont régis par la législation américaine en matière de contrôle des exportations et la liste de ressortissants spécifiquement désignés, sont rigoureusement interdites.