



NightTune User's Guide

Version 3.8

(RedHawk™ Linux®)

Copyright 2013,2014,2018 by Concurrent Real-Time. All rights reserved. This publication or any part thereof is intended for use with Concurrent Real-Time products by Concurrent Real-Time personnel, customers, and end-users. It may not be reproduced in any form without the written permission of the publisher.

Concurrent Computer Corporation and its logo are registered trademarks of Concurrent Computer Corporation. All other Concurrent product names are trademarks of Concurrent while all other product names are trademarks or registered trademarks of their respective owners.

Linux[®] is used pursuant to a sublicense from the Linux Mark Institute.

NightStar's integrated help system is based on Assistant, a Qt[®] utility. Qt is a registered trademark of Digia Plc and/or its subsidiaries.

NVIDIA[®] CUDA[™] is a trademark of NVIDIA Corporation.

Scope of Manual

This guide is designed to assist you in getting started with use of NightTune™, a process and system analysis and tuning tool.

Structure of Manual

This manual consists of four chapters, two appendices and an index. A brief description of the contents of each of the parts of the manual follows.

- Chapter 1 introduces NightTune, its command line options, and system requirements.
- Chapter 2 describes NightTune display windows.
- Chapter 3 describes each of NightTune's functional display panels.
- Chapter 4 describe NightTune's logging capabilities.
- Chapter 5 describes how to operate NightTune to execute specific monitoring and tuning tasks.
- Appendix A describes how to install and configure licenses.
- Appendix B describes dependencies of certain features on the RedHawk kernel.

Syntax Notation

The following notation is used throughout this guide:

italic

Books, reference cards, and items that the user must specify appear in *italic* type. Special terms and comments in code may also appear in *italic*.

list bold

User input appears in **list bold** type and must be entered exactly as shown. Names of directories, files, commands, options and man page references also appear in **list bold** type.

`list`

Operating system and program output such as prompts and messages and listings of files and programs appears in `list` type. Keywords also appear in `list` type.

`window`

Keyboard sequences and window features such as push buttons, radio buttons, menu items, labels, and titles appear in `window` type.

`[]`

Brackets enclose command options and arguments that are optional. You do not type the brackets if you choose to specify such option or arguments.

`{ }`

Braces enclose mutually exclusive choices separated by the pipe (`|`) character, where one choice must be selected. You do not type the braces or the pipe character with the choice.

`...`

An ellipsis follows an item that can be repeated.

Referenced Publications

The following publications are referenced in this document:

0898004	<i>RedHawk Linux User's Guide</i>
0898008	<i>NightStar RT Installation Guide</i>
0898009	<i>NightStar RT Tutorial</i>
0898395	<i>NightView™ RT User's Guide</i>
0898398	<i>NightTrace™ RT User's Guide</i>
0898458	<i>NightSim™ RT User's Guide</i>
0898465	<i>NightProbe™ RT User's Guide</i>

Contents

Chapter 1 Introducing NightTune

Point and Click Operation	1-2
Local or Remote Operation	1-3
Capabilities	1-3
Command Line Options.	1-6

Chapter 2 NightTune Windows

Menu Bar	2-1
File	2-2
View	2-7
Monitor	2-10
Tools	2-15
Help	2-17
Toolbar	2-18
Pages	2-20
Preferences.	2-22
General Preferences	2-23
Font Preferences	2-25
Control Buttons	2-27
NightStar Global Fonts Dialog.	2-29

Chapter 3 NightTune Panels

System Status Panel.	3-1
System Status Context Menu	3-3
Single Process Panel	3-6
Context Switches Panel	3-7
Context Switches Text Pane	3-7
Context Switches Bar Graph Pane	3-8
Context Switches Line Graph Pane	3-8
CPU Shielding and Binding Panel.	3-10
CPU Shielding Operations	3-11
CPU Shielding and Binding Drag and Drop Operations	3-13
CPU Shielding and Binding Context Menu	3-14
CPU Usage Panel	3-16
CPU Usage Text Pane.	3-16
CPU Usage Bar Graph Pane.	3-18
CPU Usage Line Graph Pane	3-18
CUDA Panel	3-20
CUDA Text Pane	3-20
CUDA Bar Graph Pane	3-24
CUDA Line Graph Pane.	3-26
CUDA Configuration Panel.	3-27
Driver & Library.	3-28

- Devices 3-28
 - GPU Identification 3-28
 - GPU Configuration 3-28
 - Block Configuration 3-32
 - GPU Capabilities 3-32
 - Memory 3-34
 - Texture Memory 3-34
 - Host/Remote Memory Capabilities 3-36
 - Hardware Information 3-37
 - CUDA Configuration Panel Context Menu 3-38
 - CUDA Devices Configuration Dialog 3-39
- Disk Activity Panel 3-42
 - Disk Activity Text Pane 3-42
 - Disk Activity Bar Graph Pane 3-44
 - Disk Activity Line Graph Pane 3-45
- Interrupt Activity Panel and Interrupt Activity Detail Panel 3-45
 - Interrupt (Detail) Activity Text Pane 3-46
 - Interrupt Control Drag and Drop Operations 3-47
 - Interrupt (Detail) Activity Bar Graph Pane 3-48
 - Interrupt (Detail) Activity Line Graph Pane 3-49
 - Interrupt (Detail) Activity Context Menu 3-50
 - Interrupt Affinity Dialog 3-51
- Kernel Activity Panel 3-53
 - Kernel Activity Context Menu 3-54
 - Required Privileges for the Kernel Activity Panel 3-55
- Memory Activity Panel 3-55
 - Memory Activity Text Pane 3-55
 - Memory Activity Bar Graph Pane 3-56
 - Memory Activity Line Graph Pane 3-57
- Kernel Virtual Memory Panel 3-57
 - Kernel Virtual Memory Text Pane 3-57
 - Kernel Virtual Memory Bar Graph Pane 3-58
 - Kernel Virtual Memory Line Graph Pane 3-59
- Physical Memory Panel 3-60
 - Physical Memory Text Pane 3-60
 - Subdivision by Usage 3-61
 - Subdivision by Recency 3-61
 - Subdivision by Low vs. High Memory 3-62
 - Physical Memory Bar Graph Pane 3-62
 - Physical Memory Line Graph Pane 3-64
- Swap Panel 3-65
 - Swap Text Pane 3-65
 - Swap Bar Graph Pane 3-66
 - Swap Line Graph Pane 3-66
- Network Activity Panel 3-67
 - Network Activity Text Pane 3-67
 - Network Activity Bar Graph Pane 3-68
 - Network Activity Line Graphs 3-69
- NUMA Panel 3-70
 - NUMA Text Pane 3-70
 - Subdivision by Usage 3-71
 - Subdivision by Recency 3-71
 - Subdivision by Low vs. High Memory 3-72
 - NUMA Bar Graph Pane 3-73

NUMA Line Graph Pane	3-74
NUMA Activity Panel	3-75
NUMA Activity Text Pane	3-75
NUMA Activity Bar Graph Pane	3-76
NUMA Activity Line Graph Pane	3-77
NUMA Configuration Panel	3-77
NUMA Configuration CPUs Pane	3-77
NUMA Configuration Distance Pane.	3-78
NUMA Configuration Context Menu.	3-79
PCI Configuration Panel	3-79
PCI Configuration Context Menu	3-82
Process List Panel	3-83
Process List Drag and Drop Operations	3-85
Find Bar	3-86
Process List Context Menu.	3-87
Filter Processes Dialog	3-91
Trace Output Window	3-96
Debug Process.	3-97
Process Scheduler Dialog	3-97
Process Scheduling Operations	3-100
Process Page Locking Dialog	3-101
Process Details Window	3-101
Memory Usage Tab	3-102
Subdivision by Usage	3-103
Subdivision by Shared / Nonshared	3-103
Subdivision by Residency.	3-103
Subdivision by NUMA node.	3-104
Memory Maps	3-104
Memory Tab	3-105
File Descriptors Tab	3-110
Signals Tab.	3-112
Capabilities Tab	3-114
Environment Tab	3-115
Limits Tab	3-116
Process Fields Menu	3-117
Single Process Activity Panel	3-121
Single Process Activity Context Menu.	3-123
Configure Activity Dialog	3-124
Single Process Counters Panel	3-125
Single Process Counters Text Pane	3-126
Single Process Counters Bar Graph Pane.	3-129
Single Process Counters Line Graph Pane.	3-130
Single Process Counters Context Menu.	3-131
Configure Counters Dialog.	3-134
Required Privileges for the Single Process Activity/Counters Panels.	3-135

Chapter 4 NightTune Logging

Logging Dialog	4-1
General Tab.	4-2
Processes Tab	4-6
System Tab	4-9
Logging Status Bar	4-11

Logging Examples	4-12
Configuring Logging	4-12
Using a Logging Configuration in the GUI	4-15
Process Logging	4-18
Batch Mode Logging	4-20
Batch Mode Remote Logging	4-22

Chapter 5 Guide to Operations

Monitoring User Processes	5-2
Selecting the User Process	5-2
Customizing the Process Information	5-4
Changing User Process Scheduling Attributes	5-6
Using Drag and Drop to Change Process CPU Affinity	5-7
Shielding a CPU	5-10
Changing the CPU Affinity of an Interrupt	5-14
Using Drag and Drop to Change Interrupt CPU Affinity	5-17

Appendix A NightStar RTLicensing

License Keys	A-1
License Requests	A-2
License Server	A-2
License Reports	A-3
Firewall Configuration for Floating Licenses	A-3
License Support	A-4

Appendix B Kernel Dependencies

Advantages for NightView	B-1
Advantages for NightTrace	B-1
Advantages for NightProbe	B-2
Advantages for NightTune	B-3
Frequency Based Scheduler	B-3

Index

Illustrations

Figure 2-1. File Menu	2-2
Figure 2-2. View Menu	2-4
Figure 2-3. Monitor Menu	2-5
Figure 2-4. Tools Menu	2-8
Figure 2-5. Help Menu	2-10
Figure 2-6. NightTune Main Tool Bar	2-11
Figure 2-7. Preferences Dialog	2-13
Figure 3-1. Context Switches Text Pane	3-2
Figure 3-2. Context Switches Bar Graph Pane	3-3
Figure 3-3. Context Switches Line Graph Pane	3-4
Figure 3-4. Context Switches Pop-up Menu	3-5
Figure 3-5. CPU Shielding and Binding Panel	3-6

Figure 3-6. CPU Shielding Dialog	3-8
Figure 3-7. CPU Shielding and Binding Pop-up Menu	3-11
Figure 3-8. CPU Usage Text Pane	3-13
Figure 3-9. CPU Usage Bar Graph Pane	3-14
Figure 3-10. CPU Usage Line Graph Pane	3-15
Figure 3-11. CPU Usage Pop-up Menu	3-16
Figure 3-12. Disk Activity Text Pane	3-17
Figure 3-13. Disk Activity Bar Graph Pane	3-19
Figure 3-14. Disk Activity Line Graph Pane	3-20
Figure 3-15. Disk Activity Pop-up Menu	3-21
Figure 3-16. Interrupt Activity Text Pane	3-22
Figure 3-17. Interrupt Affinity Dialog	3-24
Figure 3-18. Interrupt Activity Bar Graph Pane	3-26
Figure 3-19. Interrupt Activity Line Graph Pane	3-27
Figure 3-20. Interrupt Activity Pop-up Menu	3-28
Figure 3-21. Memory Activity Text Pane	3-30
Figure 3-22. Memory Activity Bar Graph Pane	3-31
Figure 3-23. Memory Activity Line Graph Pane	3-32
Figure 3-24. Memory Activity Pop-up Menu	3-33
Figure 3-25. Kernel Virtual Memory Text Pane	3-34
Figure 3-26. Kernel Virtual Memory Bar Graph Pane	3-35
Figure 3-27. Kernel Virtual Memory Line Graph Pane	3-36
Figure 3-28. Kernel Virtual Memory Pop-up Menu	3-37
Figure 3-29. Physical Memory Text Pane	3-39
Figure 3-30. Physical Memory Bar Graph Pane	3-41
Figure 3-31. Physical Memory Line Graph Pane	3-42
Figure 3-32. Physical Memory Pop-up Menu	3-43
Figure 3-33. Swap Memory Text Pane	3-44
Figure 3-34. Swap Memory Bar Graph Pane	3-45
Figure 3-35. Swap Memory Line Graph Pane	3-46
Figure 3-36. Swap Memory Pop-up Menu	3-47
Figure 3-37. Network Activity Text Pane	3-48
Figure 3-38. Network Activity Bar Graph Pane	3-50
Figure 3-39. Network Activity Line Graph Pane	3-51
Figure 3-40. Network Activity Pop-up Menu	3-52
Figure 3-41. NUMA Text Pane	3-54
Figure 3-42. NUMA Bar Graph Pane	3-56
Figure 3-43. NUMA Line Graph Pane	3-57
Figure 3-44. NUMA Pop-up Menu	3-58
Figure 3-45. NUMA Activity Text Pane	3-59
Figure 3-46. NUMA Activity Bar Graph Pane	3-60
Figure 3-47. NUMA Activity Line Graph Pane	3-61
Figure 3-48. NUMA Activity Pop-up Menu	3-62
Figure 3-49. NUMA Configuration Text Pane	3-63
Figure 3-50. NUMA Configuration Distance Pane	3-64
Figure 3-51. NUMA Configuration Pop-up Menu	3-64
Figure 3-52. Process List Panel	3-66
Figure 3-53. Process List Pop-up Menu	3-68
Figure 3-54. Process List Trace System Calls Dialog	3-70
Figure 3-55. Trace Library Calls Dialog	3-71
Figure 3-56. Process Scheduler Dialog	3-72
Figure 3-57. Process Details Memory Usage Tab	3-76
Figure 3-58. Process Details Memory Tab	3-78
Figure 3-59. Process Details File Descriptors Tab	3-80

Figure 3-60. Process Details Signals Tab	3-81
Figure 3-61. Process Details Capabilities Tab	3-83
Figure 3-62. Process Details Environment Tab	3-84
Figure 3-63. Process Fields Menu	3-85
Figure 3-64. Systems Menu	3-87
Figure 4-1. Monitoring User Processes	4-2
Figure 4-2. Selecting the User Process	4-3
Figure 4-3. Monitoring Multi-threaded Processes	4-4
Figure 4-4. Customizing the Process Information	4-5
Figure 4-5. Process Scheduler Dialog	4-6
Figure 4-6. Changing User Process Scheduling Attributes	4-7
Figure 4-7. Bound Processes	4-8
Figure 4-8. Using Drag and Drop to Change Process CPU Affinity	4-9
Figure 4-9. CPU Shielding and Binding Panel	4-10
Figure 4-10. Shielding a CPU	4-12
Figure 4-11. Error Shielding CPU	4-13
Figure 4-12. Maximum Shielding of a CPU	4-14
Figure 4-13. Monitoring Interrupt Activity	4-15
Figure 4-14. Interrupt Affinity Dialog	4-16
Figure 4-15. Changing the CPU Affinity of an Interrupt	4-17

Introducing NightTune

NightTune's graphical user interface (GUI) provides a powerful and intuitive point-and-click style of operation that allows you to analyze and adjust system activities with ease.

NightTune has the following features:

- Process monitoring and tuning

Through the **Process List** panel you can monitor their CPU time, memory size, scheduling parameters, CPU affinity and other attributes of individual processes. You can monitor the same attributes of individual threads within a process. You can modify the scheduling parameters and CPU affinity with the **Process Scheduler** dialog or with drag-and-drop operations to the **CPU Shielding and Binding** panel. You can also lock a process's pages into memory. Finally, you can view their memory usage, file descriptors, signals, capabilities, and environment variables.

- CPU control

You can monitor the status of a **CPU**, including information about shielding and process and interrupt bindings. If you have the appropriate privileges, you can set shielding and hyper-threading attributes for each CPU.

- System monitoring and tuning

With the system activity panels, you can monitor:

- Context Switches
- CPU Usage
- CPU Shielding and Binding
- CUDA
- CUDA Configuration
- Disk Activity
- Interrupt Activity
- Interrupt Detail Activity
- Kernel Activity
- Memory Activity
- Kernel Memory
- Physical Memory
- Swap Space

- Network Activity
- NUMA Memory
- NUMA Memory Activity
- NUMA Memory Configuration
- PCI Configuration

Information is available both numerically and graphically. The Interrupt Activity panel allows you to change the CPU affinity for individual interrupts.

- Single process monitoring

With the single process panels, you can monitor a single user-specified process to view the following:

- Single Process Activity
- Single Process Counters

- Tuning Activity log

This facility provides optional logging of tuning activity to a user-specified log file.

- System State Restoration

This facility allows you to reapply tuning changes you made to CPUs, Interrupts, Processes, or Threads in a subsequent NightTune session -- either in unattended batch mode, or through the graphical user interface.

- Comprehensive online help

The help system includes context-sensitive help, accessible by clicking on any item in NightTune's window.

NightTune uses the NightStar License Manager (NSLM) to control access to the NightStar RT tools. See "NightStar RTLicensing" on page A-1 for more information.

Point and Click Operation

NightTune provides a point-and-click interface. Most operations utilize the mouse.

NightTune makes extensive use of right-click context menus and drag-and-drop actions.

Local or Remote Operation

NightTune can operate on local or remote systems. When operating remotely, the graphical user interface runs on the host system where NightTune is invoked and communicates to a NightTune server process which is launched automatically on the remote target system.

You can also operate remotely when using NightTune without the graphical user interface -- logging system and process activity or restoring system state. You can do this for multiple remote systems from a single command invocation.

The Pluggable Authentication Module (PAM) is used to authenticate connection requests to remote systems (see `pam(8)`). The file `/etc/pam.d/ntune` is installed as part of the NightTune product; it defines the specific authentication mechanisms that are used for each target system. For security reasons, NightTune encrypts usernames and passwords during all authentication requests and never stores authentication information to disk.

See “Command Line Options” on page 1-6 for more information on remote operation of NightTune.

Capabilities

Most operations within NightTune do not require any special privileges, but some do have such requirements.

Linux provides a means to grant otherwise unprivileged users the authority to perform certain privileged operations. The Pluggable Authentication Module (see `pam_capability(8)`) is used to manage sets of capabilities, called *roles*, required for various activities.

For the features shown in the following table, you must either be the `root` user, or have the capability as shown.

Table 1-1.

Feature	Required Capability
Modify process scheduling attributes	CAP_SYS_NICE (for SCHED_FIFO and SCHED_RR policies)
Adjust CPU Shielding	CAP_SYS_NICE
Adjust IRQ CPU affinity	CAP_SYS_NICE
Kernel Activity Panel	CAP_SYS_NICE (not strictly required -- see “Required Privileges for the Kernel Activity Panel” on page 3-55)

Table 1-1.

Feature	Required Capability
Locking Process Pages	CAP_IPC_LOCK, CAP_SYS_NICE (for locking other users' processes' pages)
Single Process Activity Panel	CAP_SYS_PTRACE (for monitoring other users' processes)
Single Process Counters Panel	CAP_SYS_PTRACE (for monitoring other users' processes)
Process System Call Tracing	CAP_SYS_PTRACE (for tracing other users' processes)
Sending Signals	CAP_SYS_NICE (for signalling other users's processes)
Process Details	CAP_SYS_NICE (for viewing details of other users' processes -- some limitations remain even with this capability)

As such, for ease of use, Linux systems should be configured with an `ntuneuser` role which provides the `CAP_SYS_NICE`, `CAP_SYS_PTRACE`, and `CAP_IPC_LOCK` capabilities.

Edit `/etc/security/capability.conf` and define the `ntuneuser` role (if it is not already defined as shown here) in the "ROLES" section:

```
role ntuneuser CAP_SYS_NICE CAP_SYS_PTRACE CAP_IPC_LOCK
```

Additionally, for each NightTune user on the target system, add the following line at the end of the file:

```
user username ntuneuser
```

where *username* is the login name of the user.

If the user requires capabilities not defined in the `ntuneuser` role, add a new role which contains `ntuneuser` and the additional capabilities needed, and substitute the new role name for `ntuneuser` in the text above.

In addition to registering your login name in `/etc/security/capability.conf`, certain files under the `/etc/pam.d` directory must also be configured to allow capabilities to be activated.

To activate capabilities, add the following line to the end of selected files in `/etc/pam.d` if it is not already present:

```
session required pam_capability.so
```

The list of files to modify is dependent on the list of methods that will be used to access the system. The following table presents a recommended configuration that will grant capabilities to users of the services most commonly employed in accessing a system.

Table 1-2. Recommended /etc/pam.d Configuration

/etc/pam.d File	Affected Services	Comment
remote	telnet rlogin rsh (when used <u>w/o</u> a command)	Depending on your system, the remote file may not exist. Do not create the remote file, but edit it if it is present.
login	local login (e.g. console) telnet* rlogin* rsh* (when used <u>w/o</u> a command)	*On some versions of Linux, the presence of the remote file limits the scope of the login file to local logins. In such cases, the other services listed here with login are then affected solely by the remote configuration file.
password-auth	Most services	This file is present on newer Linux systems; if it exists, add the entry to it as well.
rsh	rsh (when used <u>with</u> a command)	e.g. rsh system_name a.out
sshd	ssh	You must also edit /etc/ssh/sshd_config and ensure that the following line is present: UsePrivilegeSeparation no
gdm gdm-password	gnome sessions	Some systems require that gdm-password is modified as well.
kde	kde sessions	

If you modify **/etc/pam.d/sshd** or **/etc/ssh/sshd_config**, you must restart the **sshd** service for the changes to take effect:

```
service sshd restart
```

In order for the above changes to take effect, the user must log off and log back onto the target system.

NOTE

To verify that you have been granted capabilities, issue the following command:

```
/usr/sbin/getpcaps $$
```

The output from that command will list the roles currently assigned to you.

Command Line Options

Use the following command to start NightTune:

```
ntune [Options] [handle]
```

Options are described as follows:

```
--config=config-file  
-c config-file
```

specifies the configuration file that contains the start-up information for NightTune. In the absence of the **--config** option, NightTune searches for a configuration file using the following ordered criteria:

- a **.NightTunerc** file in the current working directory
- a **.NightTunerc** file in the user's home directory
- **/usr/lib/NightTune/lib/config**

```
--dry-run
```

When used with the **--restore-state** option, NightTune merely prints the actions which would be taken to restore a state. No actions are taken.

```
--help
```

causes NightTune to display its command line syntax (followed by a brief description of each option), then exit.

```
--no-gui
```

specifies that NightTune will be invoked in batch mode without the graphical user interface. Batch mode is useful if the specified (or implied) configuration file specifies that logging will occur (see "NightTune Logging" on page 4-1) or when restoring the state of the system (CPU shielding, interrupt CPU affinity, process and thread scheduling attributes and CPU affinity). The **--no-gui** option requires the *handle* parameter, which identifies the session, unless the **--wait** option is specified.

In batch mode, the *ntune* command immediately returns after launching a daemon that continues to run in the background (unless the **--wait** option is specified).

The *handle* argument, which is required with the **--no-gui** option, identifies the daemon process for subsequent **--quit** invocations.

Handle must be a simple file name and corresponds to a file under **/var/lib/NightTune**, which is created and managed by **ntune**. Since the namespace is flat, use of a unique handle name is recommended to avoid collisions (e.g. *ntune --no-gui \$\$*).

A typical sequence for batch-mode logging might be:


```
ntune --config=my-logging-config --no-gui hamster
./a.out
ntune --quit hamster
```

--quit

terminates a previously-invoked ntune daemon launched with the **--no-gui** option.

The *handle* parameter is required and identifies the daemon process.

--restore-state

implies the **--no-gui** option which causes NightTune to run in batch mode. See **--no-gui** for other required options. The required configuration file specified with the **--config** option must contain system state information saved from a previous NightTune session.

Restoring the system's state involves analyzing and adjusting CPU shielding, interrupt CPU affinity, process and thread scheduling attributes and CPU affinity.

NightTune can be run in daemon mode (the default with **--restore-state** unless **--wait** is specified) which will cause it to continually analyze the system and keep it in sync with the specified state. This can be useful in order to adjust applications or threads which come and go.

As long as the specified configuration contains system state information, a single-shot batch invocation for state restoration can be as following:

```
ntune -w 0 --config=state.config
```

You can restore the state of remote systems by using **--target** as well; multiple **--target** options can be specified and the state of all named systems will be restored.

See "Restore System State..." on page 2-4 for a more complete description of system state restoration.

--target=system-name

-t system-name

causes NightTune to operate on the specified remote target system. The NightTune GUI will run on the system where **ntune** was invoked and will communicate with a NightTune server process, which is automatically launched on the target system.

NightTune connects to the target system using the **nstar.d** daemon, supplied as part of NightStar RT tools, running on that target system.

Multiple **--target** options can be specified and can be used with the **--no-gui** option to do logging on remote systems.

--wait=seconds

-w seconds

instructs **ntune** to run in batch mode (implies **--no-gui**) for the specified number of seconds, and then exit. For example:

```
ntune --config=my-logging-config --wait=5
```

NightTune Windows

NightTune features are available through a number of functional panels that are placed in tabbed pages in NightTune windows.

Much of the time you will operate NightTune with a single window displaying several panels of information within one or more tabbed pages. However, NightTune allows you to have multiple windows active at once, customized to display any panels of interest.

Each window has:

- a menu bar (see “Menu Bar” on page 2-1)
- a toolbar that provides shortcuts and affects how data is refreshed (see “Toolbar” on page 2-18)
- a display and control area, which is the main body of the window where tabbed pages display selected panels (see “NightTune Panels” on page 3-1)

Menu Bar

The menu bar provides access to controlling which panels are displayed in the window as well as the appearance of items inside panels, preference settings, launching additional tools, and obtaining help.

The menu bar provides the following menus:

- File
- View
- Monitor
- Tools
- Help

Each menu is described in the sections that follow.

An icon appearing at the left of a menu item indicates that the corresponding icon on the toolbar can also be used to perform that function.

File

Mnemonic: Alt+F

The **File** menu allows you to set preferences, load or save configuration data, activate or deactivate logging, and exit NightTune.

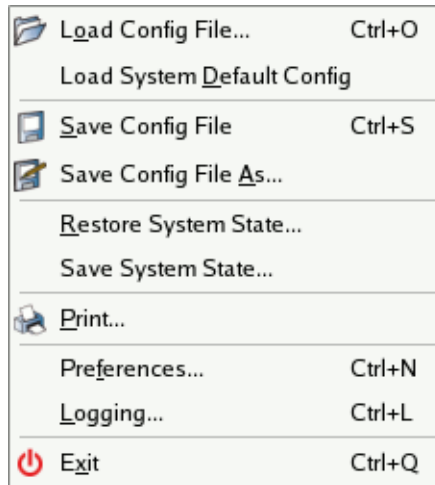


Figure 2-1. File Menu

The following paragraphs describe the options on the **File** menu in more detail.

Load Config File...

Mnemonic: O
Accelerator: Ctrl+O

This menu item displays a file selection dialog. By selecting a previously-saved configuration file, NightTune closes all current windows, tabbed pages, and panels and opens windows, tabbed pages, and panels as defined in the configuration file.

Load System Default Config

Mnemonic: D

This menu item loads the system default configuration, reverting the configuration of windows, tabs, panels and panes back to their original configuration before any user modifications were made.

Save Config File

Mnemonic: S
Accelerator: Ctrl+S

This menu item allows you to save the configuration from the current session to a configuration file. Configuration data includes the layout of panels in tabbed pages for windows in the current session as well as settings in the **Preferences** dialog (see “Preferences” on page 2-22).

When invoked without the **--config** or **-c** options, NightTune automatically searches for a configuration file. It first looks for **.NightTunerc** in the current working directory and then in your home directory, and finally in **/usr/lib/NightTune/lib/config**. The first file found is automatically loaded when NightTune launches.

The configuration will be saved to whatever configuration file was saved or loaded last. If no configuration file was ever saved or loaded during the current NightTune session, then the configuration file name will be **.NightTunerc** and will be placed in your home directory.

For configurations involving remote systems, the names of those systems can be saved in the configuration, if so desired. See “Bind Monitor Panels” on page 2-24 for more information.

Save Config File As...

Mnemonic: A

This menu item allows you to save the configuration from the current session to a configuration file with a name of your choice.

When you select this menu item, NightTune displays a file selection dialog. After making a selection, the configuration data from the current session is saved in the selected file. Configuration data includes the layout of panels in tabbed pages for windows in the current session as well as settings in the **Preferences** dialog (see “Preferences” on page 2-22).

Restore System State...

This menu item displays a dialog which allows you to apply a previously saved system state to the current system.

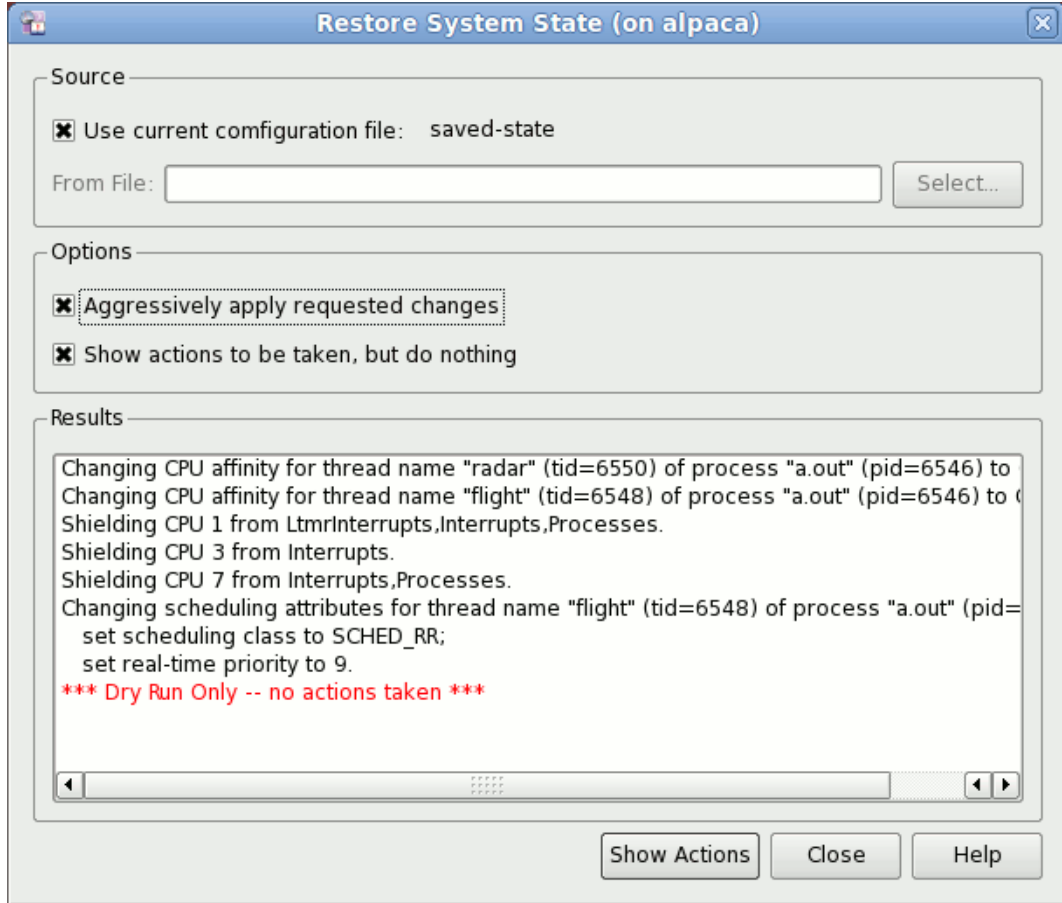


Figure 2-2. Restore System State Dialog

The “system state” in this context includes the shielding status of all CPUs, the scheduling attributes of selected processes or threads, and the CPU affinities of selected IRQs.

If the current configuration file contains saved system state data, then the **Use current configuration file** option will be checked by default. To load a saved system state from another file, clear the checkbox and specify the file name. System state is saved by using the **Save System State** option from the File menu. See “Save System State...” on page 2-5.

Options

This area provides you two options.

Aggressively apply requested changes allows NightTune to move processes or IRQs off of CPUs that are to be marked down.

Show actions to be taken, but do nothing allows you to see what NightTune would do to restore the state, as shown in the figure above. When this option is checked, the action button at the bottom says **Show Actions**. Otherwise, the action but says **Execute** and pressing it will cause NightTune to take actions to restore the system state.

Save System State...

Mnemonic: S

Shows a dialog that allows you to save tuning actions executed in this NightTune session for subsequent reuse to create the system state.

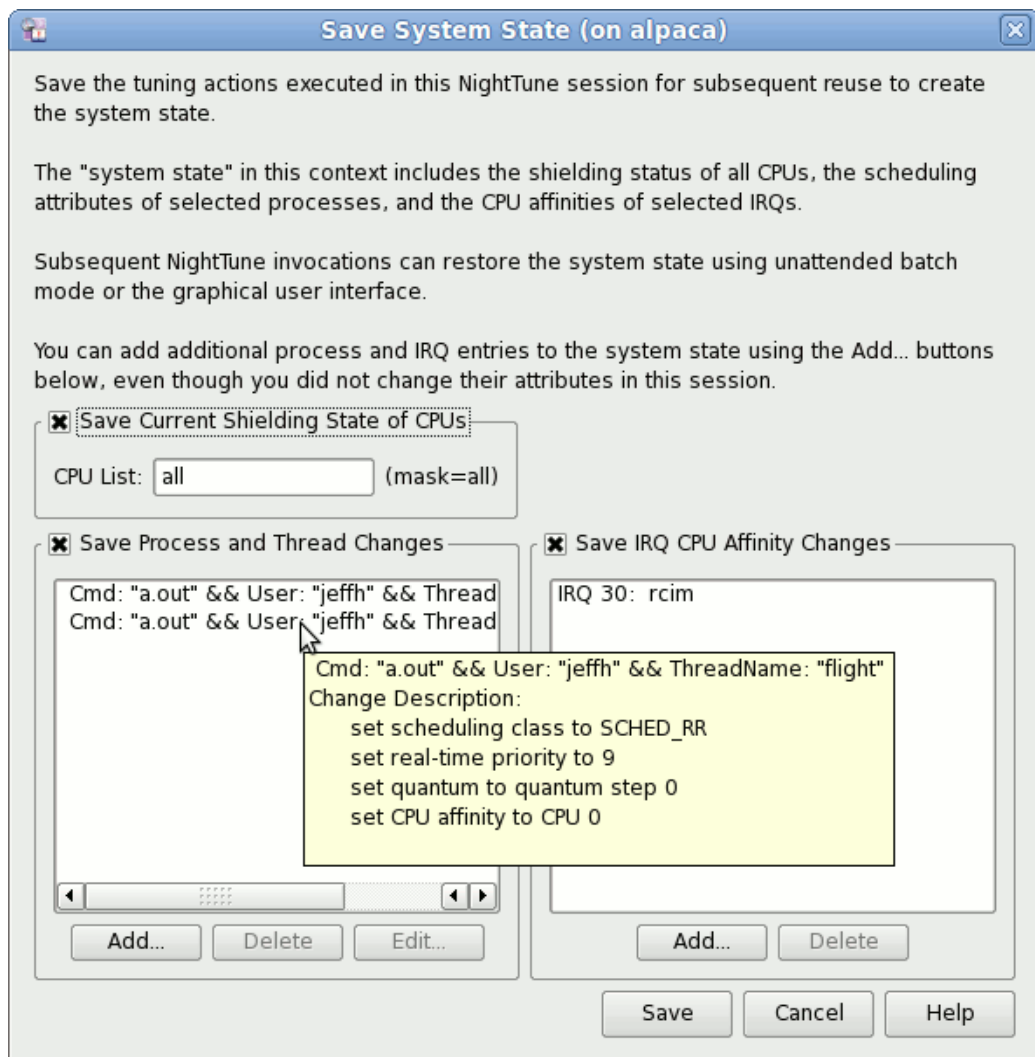


Figure 2-3. Save System State Dialog

The “system state” in this context includes the shielding status of all CPUs, the scheduling attributes of selected processes or threads, and the CPU affinities of selected IRQs.

Subsequent NightTune invocations can restore the system state using unattended batch mode or the graphical user interface.

The dialog also allows to you add additional process, thread, and IRQ entries to the system state, even though you did not change their attributes in this session.

Save Current Shielding State of CPUs

When checked, the shielding state of all CPUs in the specified list will be included in the saved state. The shielding state of the CPUs at the time you press the **Save** button will be used.

Save Process and Thread Changes

When checked, the scheduling attributes (policy, priority, and CPU affinity) for the processes or threads in the list is saved.

Hovering the mouse cursor over an item in the list will show you the attribute changes that will be saved.

You can delete items from the list or add new items -- even items you did not adjust in the current session.

By default, processes are identified using the user name and the command simple name of the process. If individual threads are in the list, their thread name or thread number is used as well. You can change the rules for process or thread identification by selecting an item and pressing **Edit**, which shows a process or thread identification dialog that allows you to change or add rules for identification. Identification of threads is rather tricky, since thread names aren't available unless you are using the NightTrace Logging API and have named your threads or the process is being debugged by NightView. If the threads have no names that NightTune can identify, it is best to identify them by their relative ordering within the process (thread number 1 is the first thread in the list of threads sorted by their TID (gettid(2)), as shown in NightTune).

Save IRQ CPU Affinity Changes

When checked, the CPU affinity of IRQs in the list are included in the state.

You can delete items from the list or add additional IRQs even though you did not change their affinity in the current NightTune session.

This menu item displays the Printer dialog which allows you to print the NightTune window.

The saved state can be restored by invoking NightTune in batch mode or by using the graphical user interface. See “**--restore-state**” on page 1-7 and “Restore System State...” on page 2-4.

Preferences...

Mnemonic: N
 Accelerator: Ctrl+N

This menu item displays the **Preferences** dialog which allows you to customize logging attributes and refresh intervals for various panels.

See “Preferences” on page 2-22 for more information.

Logging

This menu item launches the Logging Dialog which controls whether logging is enabled, the output format for logging, and which attributes and actions are to be logged.

See “Logging Dialog” on page 4-1 for more information.

Exit

Mnemonic: X
 Accelerator: Ctrl+Q

This menu item exits NightTune. NightTune will not prompt you to save any unsaved configuration data before exiting – select **Save Config File** prior to exiting to retain configuration data for subsequent invocations of NightTune.

View

Mnemonic: Alt+V

The **View** menu allows you to control tabbed pages and toolbars in the current NightTune session.

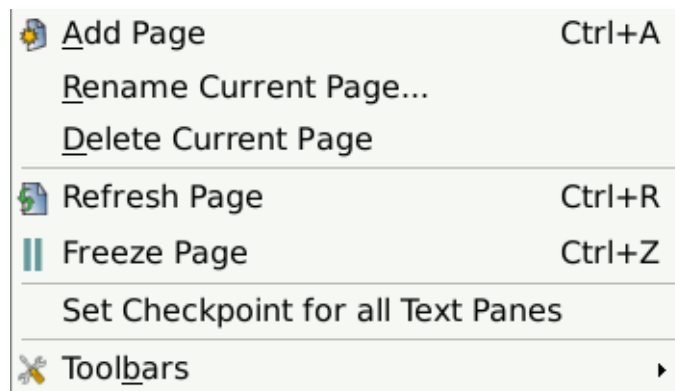


Figure 2-4. View Menu

The following paragraphs describe the options on the **View** menu in more detail.

Add Page

Mnemonic: A
Accelerator: Ctrl+A

This menu item creates a new tabbed page in the current window.

A dialog can be displayed by right-clicking on the tab that allows you to rename, delete or move the new page.

Rename Current Page...

Mnemonic: R

This menu item opens a dialog which allows you to change the name associated with the tabbed page currently displayed in the window.

Delete Current Page

Mnemonic: D

This menu item deletes the tabbed page currently displayed in the window.

Refresh Page

Accelerator: Ctrl+R

The **Refresh** menu item immediately refreshes the displayed data in all panels in the current tabbed page.

Freeze Page

Mnemonic: Z
Accelerator: Ctrl+Z

This menu item toggles the **Freeze** setting for all panels in the current tabbed page. When frozen, data values are not refreshed automatically; however, you can manually refresh the data on an otherwise frozen page by using the **Refresh Page** menu item.

Set Checkpoint for all Text Panes

This menu item causes every system status panel that supports display totals instead of rates to set its checkpoint to the current data. This is equivalent to using each of those panels' **Set Checkpoint** menu item simultaneously. It is intended to allow displaying totals from a consistent checkpoint. This menu item also affects **Single Process Counter** panels.

Note that, because checkpoints are done for each individual panel, this cannot affect panels created after setting the checkpoint.

Toolbars

Mnemonic: B

This menu item opens a sub-menu to control which toolbars are visible. If an item in this menu is checked, its corresponding toolbar will be visible; otherwise it will not. The toolbars are:

- File Toolbar
- View Toolbar
- Monitor Toolbar
- Connect Toolbar
- Drop Site

The first four contain icons for functions from their corresponding menus. The **Drop Site** toolbar contains two icons used as targets for drag and drop operations (see “CPU Shielding and Binding Drag and Drop Operations” on page 3-13 and “Process List Drag and Drop Operations” on page 3-85).

In addition, the sub-menu contains the **Show All Toolbars** and **Hide All Toolbars** menu items, which check all toolbars and uncheck them, respectively.

Monitor

Mnemonic: Alt+M

The Monitor menu allows you to select which panels are to be displayed in the tabbed page currently visible in the NightTune display area.

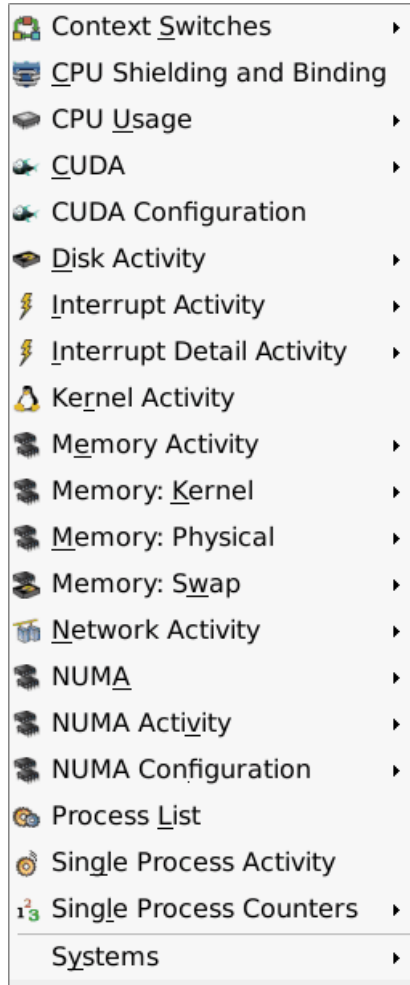


Figure 2-5. Monitor Menu

NOTE

The Monitor menu, like the other menus, can be “torn off” from the menu bar to reside in its own window separate from NightTune by clicking on the dashed line at the top of the menu. This is especially useful when making multiple changes to the panels to avoid having to re-select the menu after every choice.

Menu items with an arrow at the right display a sub-menu for controlling which panes for that panel are displayed in the NightTune display area. For example, many panels have a Text pane, a Bar Graph pane and a Line Graph pane. Also on such sub-menus, an additional choice will exist to display all of those panes.

The following paragraphs summarize individual panel activities. Detailed descriptions for each panel are provided in “NightTune Panels” on page 3-1.

Context Switches

Mnemonic: S

This panel displays context switches per CPU per second using textual and graphical displays.

CPU Shielding and Binding

Mnemonic: C

This panel describes each CPU in the system and includes information on the state of shielding, process binding, and interrupt binding. It allows you to change the shielding attributes of CPUs and to redefine the CPU binding of specific processes or interrupts.

CPU Usage

Mnemonic: U

This panel displays CPU utilization using textual and graphical displays. It provides User, System, Wait, and Idle time information per CPU.

CUDA

This panel displays usage, memory activity and usage, power usage, temperatures, and fan speeds about any nVidia CUDA-capable devices on the system.

CUDA Configuration

This panel displays configuration information about any nVidia CUDA-capable devices on the system.

Disk Activity

Mnemonic: D

This panel provides detailed disk activity for all disks in the system. It includes information about read and write operations and service times using textual and graphical displays.

Interrupt Activity

Mnemonic: I

This panel displays interrupt information for each interrupt across all CPUs using textual and graphical displays. It allows you to change the CPU affinity of individual interrupts.

Interrupt Detail Activity

Mnemonic: I

This panel displays interrupt information for each interrupt on each CPU using textual and graphical displays. It allows you to change the CPU affinity of individual interrupts.

Kernel Activity

Mnemonic: R

This panel displays an estimate of the amount of time spent in each function within the system kernel.

Memory Activity

Mnemonic: E

This panel displays physical memory page transfer rates on the system using textual and graphical displays.

Memory: Kernel

Mnemonic: V

This panel displays allocation of kernel virtual memory (vmalloc) on the system using textual and graphical displays.

Memory: Physical

Mnemonic: M

This panel displays allocation of physical memory on the system using textual and graphical displays.

Memory: Swap

Mnemonic: W

This panel displays allocation of swap space on the system using textual and graphical displays.

Network Activity

Mnemonic: N

This panel describes network activity in terms of packet I/O rates, errors, and collisions for each network device on the system using textual and graphical displays.

NUMA

Mnemonic: A

This panel describes physical memory allocation for each node on NUMA systems using textual and graphical displays.

NUMA Activity

Mnemonic: V

This panel describes physical memory activity for each node on NUMA systems using textual and graphical displays.

NUMA Configuration

Mnemonic: F

This panel describes the configuration of the NUMA systems, including the CPUs connected directly to each node and the relative distances from CPUs to memory on each node.

Process List

Mnemonic: L

This panel provides information about processes. Additional detailed information is available for individual processes via a dialog window. The ability to change scheduling attributes for processes is available via another dialog window.

Single Process Activity

Mnemonic: G

This panel displays an estimate of the amount of time spent in each function within a user-specified running process.

Single Process Counters

Mnemonic: L

This panel displays counts of various activities and events for a user-specified running process. Counters include instructions executed, branches, cache references, context switches, migrations, page faults, etc.

Systems

Mnemonic: Y

This menu item allows you to control connection to other systems on which NightTune is installed.

The following illustrates the **Systems** menu:

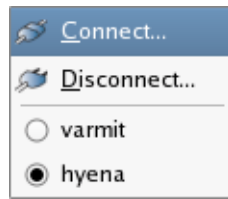


Figure 2-6. Systems Menu

The following paragraphs describe each of the selections in more detail:

Connect...

Mnemonic: C

This displays the Connect To System dialog allowing you to connect to another system.

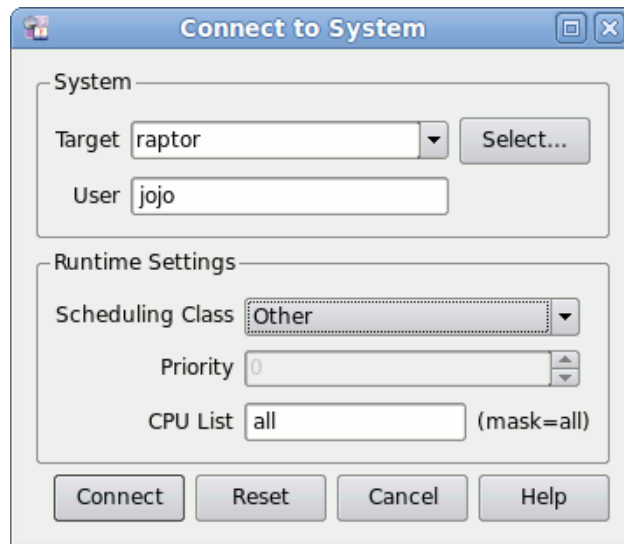


Figure 2-7. Connect To System Dialog

System Target/User

To connect to a system, specify its name or IP address in the **Target** field or select it from the list displayed by pressing the **Select...** button (which derives its information from the `/etc/hosts` file). You will only be able to connect to a system on which the same version of the NightTune server is installed. Provide the **User** name with which you wish to connect.

Runtime Settings

This area allows you to set the CPU affinity, scheduling class and priority for the NightTune server on the target system, if desired.

Connect

When you press the **Connect** button, NightTune attempts to authorize the specified **User** on the specified **Target** system using **ssh**. You may be prompted for ssh passphrases or a password, depending on how you have **sshd** configured on the target system.

If you have already added ssh keys to your current session, you may not need to be prompted for authentication information (see **ssh-add (1)** and **ssh-agent (1)** for more information on ssh authentication).

Disconnect...

Mnemonic: D

This displays a dialog that allows you to disconnect from any of the systems to which you currently are connected.

System Names

The names of the systems to which you are connected are listed. Selecting a system name activates it as the “current” system for creating new panels. When creating new panels, they always correspond to the current system. The current system name is indicated by the filled radio button.

Tools

Mnemonic: Alt+T

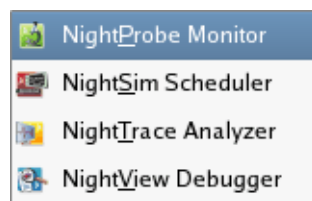


Figure 2-8. Tools Menu

The following describe the options on the **Tools** menu:

NightProbe Monitor

Mnemonic: P

Opens the NightProbe Data Monitoring tool. NightProbe is a real-time graphical tool for monitoring, recording, and altering program data within one or more executing programs without significant intrusion. NightProbe can be used in a development environment as a tool for debugging or in a production environment for data capture or to create a “control panel” for program input and output.

See also:

- *NightProbe RT User's Guide*

NightSim Scheduler

Mnemonic: S

Opens the NightSim Application Scheduler. NightSim is a tool for scheduling and monitoring real-time applications which require predictable, repetitive process execution. With NightSim, application builders can control and dynamically adjust the periodic execution of multiple coordinated processes, their priorities, and their CPU assignments.

See also:

- *NightSim RT User's Guide*

NightTrace Analyzer

Mnemonic: T

Opens the NightTrace Analyzer. The NightTrace Analyzer is a graphical tool for analyzing the dynamic behavior of multi-process and/or multi-processor user applications and operating system activity. NightTrace allows you to control user and kernel trace collection daemons and can graphically display the interplay between many real-time programs and processes across multiple processors and systems.

See also:

- *NightTrace RT User's Guide*

NightView Debugger

Mnemonic: V

Opens the NightView Source-Level Debugger. NightView is a graphical source-level debugging and monitoring tool specifically designed for real-time applications. NightView can monitor, debug, and patch multiple real-time processes running on multiple processors with minimal intrusion.

See also:

- *NightView RT User's Guide*

Help

Mnemonic: Alt+H

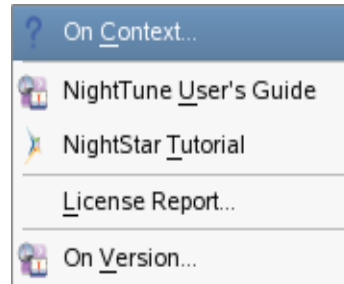


Figure 2-9. Help Menu

The following describe the options on the Help menu:

On Context...

Mnemonic: C

Gives context-sensitive help on the various menu items, dialogs, or other parts of the user interface.

Help for a particular item is obtained by first choosing this menu item, then clicking the mouse pointer on the object for which help is desired. The mouse pointer becomes a floating question mark when the **On Context** menu item is selected.

In addition, context-sensitive help may be obtained for the widget with the current focus by pressing the F1 key. NightTune's online help system will open with the appropriate topic displayed.

NightTune User's Guide

Mnemonic: G

Opens the online version of the *NightTune RT User's Guide* in the online help viewer.

NightStar RT Tutorial

Mnemonic: T

Opens the online version of the *NightStar RT Tutorial* in the online help viewer.

License Report...

Mnemonic: L

Provides licensing information.

On Version

Mnemonic: V

Displays a short description of the current version of NightTune.

Check for Updates...

Mnemonic: U





Opens NUU, the Concurrent Network Update Utility. This tool checks online repositories for updates to NightStar and other tools.














Toolbar


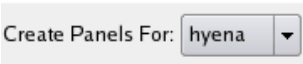


The NightTune tool bar provides icons for commonly used actions.



Figure 2-10. NightTune Main Tool Bar

	<p>Load Config File Loads the config file</p>
	<p>Save Config File Saves the config file</p>
	<p>Print Displays the Print dialog that allows you to print the NightTune Window</p>
	<p>Add Page Adds a tabbed page to the display</p>
	<p>Freeze Page Halts the automatic refresh of displayed data within the window. It affects all panels within the window; however, individual panels can subsequently override the window frozen state using context menus.</p>

	<p>Refresh Page</p> <p>Causes all data displayed in all panels associated with the window to be refreshed once, regardless of the state of freeze on the window or in individual panels.</p>
	<p>Process List</p> <p>Displays the Process List panel</p>
	<p>CPU Shielding and Binding</p> <p>Displays the CPU Shielding and Binding panel</p>
	<p>CPU Usage</p> <p>Displays the CPU Usage panel</p>
	<p>Memory: Physical</p> <p>Displays the Physical Memory panel</p>
	<p>Memory: Swap</p> <p>Displays the Swap Memory panel</p>
	<p>Interrupt Activity</p> <p>Displays the Interrupt Activity panel</p>
	<p>Network Activity</p> <p>Displays the Network Activity panel</p>
	<p>Disk Activity</p> <p>Displays the Disk Activity panel</p>
	<p>Context Switches</p> <p>Displays the Context Switches panel</p>
	<p>CUDA</p> <p>Displays the CUDA panel</p>
	<p>Kernel Activity</p> <p>Displays the Kernel Activity panel</p>
	<p>Single Process Activity</p> <p>Displays the Single Process Activity panel</p>
	<p>Single Process Counters</p> <p>Displays the Single Process Activity panel</p>

	<p>Kill dragged processes</p> <p>If a process or thread is dragged to this icon, that process or thread is killed with a SIGKILL.</p>
	<p>Unbind dragged processes/interrupts</p> <p>If a process, thread, or interrupt is dragged to this icon, that process, thread, or interrupt is unbound from any CPUs. That is, its CPU affinity is set to include all CPUs.</p> <p>Note that interrupts can be dragged to this icon only if NightTune is running as <code>root</code> or if the user running NightTune has been granted the proper capabilities (see “Capabilities” on page 1-3).</p>
	<p>Create panels for</p> <p>When a new NightTune panel is created, the system shown here is the one which will be monitored by that new panel. The dropdown lists all NightTune remote systems to which you are connected. To create a panel for a system other than the one shown here, select it in the dropdown first.</p>
	<p>Connect to a system</p> <p>Displays a dialog allowing connection to another system with NightTune installed</p>
	<p>Disconnect from a system</p> <p>Displays a dialog allowing disconnection from a system</p>

Pages

The remaining area of the main window is reserved for various tabbed pages which reflect and control the current configuration, provide browsing of programs, and viewing of live and recorded data.

Each page has a tab which contains the page title. When clicked or right-clicked, the page is raised to the top and becomes the current page.

Each tab has a context menu which allows you to manipulate the page position and title.

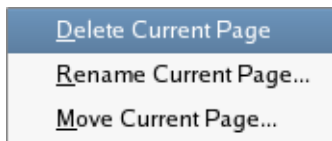


Figure 2-11. Tab Context Menu

Delete Current Page

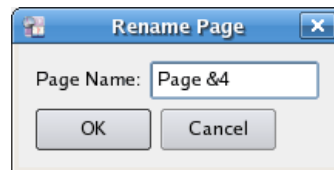
Mnemonic: D

This option deletes the current page.

Rename Current Page

Mnemonic: R

This option displays a dialog which allows you to rename the current page.

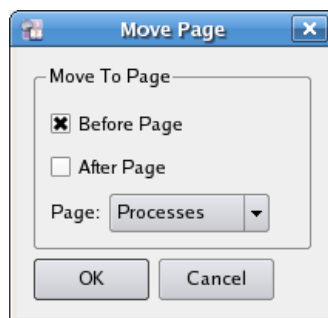
**Figure 2-12. Rename Page Dialog**

If the page title contains an ampersand character (&), it causes the next character to be underlined in the title, provides a keyboard shortcut for that page, and the ampersand becomes invisible in the title that is shown for the page. In the example above, the keyboard shortcut for this page will be **Alt+4** and the displayed title will become **Page 4**. Activating the shortcut for a page causes it to be raised to the top and it becomes the current page. Care should be taken when choosing shortcuts for pages so they do not conflict with other shortcuts. If you wish to have an ampersand displayed in the actual page title (as opposed to defining a shortcut), use two consecutive ampersand characters in the **Rename Page** dialog.

Move Current Page

Mnemonic: M

This option displays a dialog which allows you to reposition the current page among other pages. This option will be disabled unless at least two viewing pages exist.

**Figure 2-13. Move Page Dialog**

Preferences

The Preferences dialog, available from the File menu, allows you to tailor aspects of how NightTune operates.

These preferences are part of NightTune configuration data which can be saved to a configuration file so that subsequent invocations of NightTune can use the customized settings.

The following illustrates the Preferences dialog:

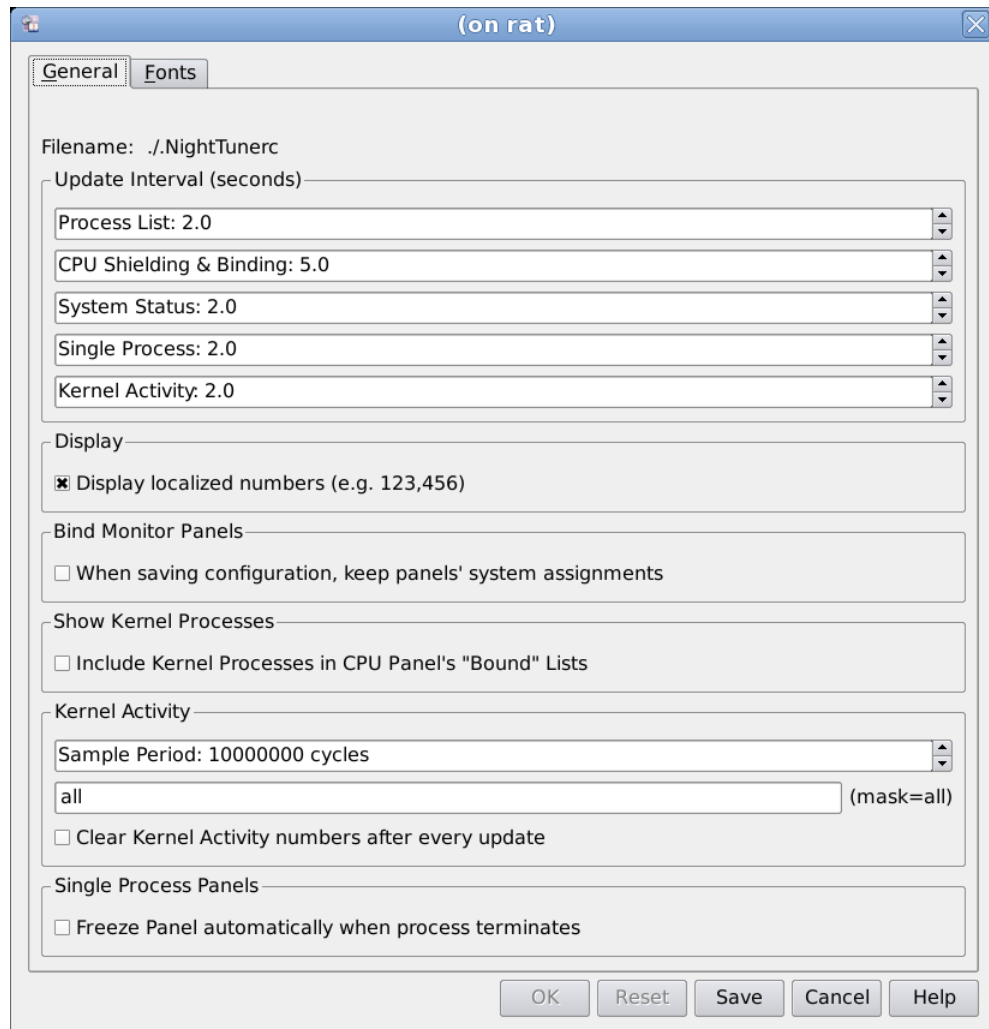


Figure 2-14. Preferences Dialog

General Preferences

The following paragraphs describe the labels and control features of the general page in the dialog:

Configuration File

This area shows the name of the current configuration file. The file name can be changed using the **Save Config File As...** menu item from the **File** menu, or a new configuration file can be loaded using the **Load Config File...** menu item, also from the **File** menu.

Update Intervals (secs)

This area allows you to control the update intervals used by NightTune to refresh displayed data. The units of all intervals are in seconds and can include fractional values.

An update interval of zero indicates that automatic updates will not occur. In such an instance, displays will be updated only when you click the **Refresh** tool icon or select a **Refresh** option from panel context menus.

Process List

The interval between which the **Process List** panel scans for new processes and updates the **Process List** panel display.

CPU Shielding & Binding

The interval between which the shielding attributes, bound processes and interrupts are updated in the **CPU Shielding and Binding** panel.

System Status

The interval between which data within all system status panels are updated (see “System Status Panel” on page 3-1).

Single Process

The interval between which data within all single process panels are updated (see “Single Process Panel” on page 3-6).

Kernel Activity

The interval between which data within the kernel activity panel are updated (see “Kernel Activity Panel” on page 3-53).

Display

Display Localized Numbers

The checkbox determines whether or not numbers will be displayed in a localized fashion (e.g. with commas and decimal points such as 123,456.78 in United States English). By default, this setting is checked.

Bind Monitor Panels

The checkbox here affects how panel information is saved when saving a configuration. If checked, when the configuration is used in the future, the panels will monitor those same systems. This would be useful if you wanted to create a configuration that always monitored the same system or set of systems.

If this item is not checked then, when saving a configuration, information about the particular systems being monitored by each panel is not saved as part of the configuration. If the configuration is used in the future, every panel would monitor the target system if specified, or the local system otherwise. This primarily is used for configurations designed to monitor a single system.

Show Kernel Processes

The checkbox here controls whether or not the CPU Shielding and Binding panel shows kernel processes bound to particular CPUs. The Linux kernel includes some processes which are visible to the user (e.g. `ksoftirqd/0`). Generally, users do not wish to see these processes, and so the default behavior is to exclude them from the CPU Shielding and Binding panel. But if this checkbox is checked, they will be displayed as appropriate for their CPU binding.

Kernel Activity

This area allows you to control collection and display of data in the Kernel Activity panel.

Sample Period

The sample period between snapshots of the kernel execution. It is measured in clock cycles. If this number is set too low, it may be throttled to a more reasonable value.

CPU Mask

The mask of CPUs on which to sample kernel execution. It may be a list of CPU numbers or ranges, or `all`.

Clear Kernel Activity

The checkbox here controls whether the counts for each kernel routine are cleared for every update. If checked, they are cleared. If unchecked, they accumulate throughout the life of the window.

Font Preferences

NightTune uses multiple fonts to present text in the most effective manner throughout the various display areas of the tool.

Variable-width fonts are most commonly used; these fonts most closely resemble how people write or print words.

Fixed-width fonts require that all characters and numbers have the same width (visual footprint). Fixed-width fonts are of benefit when source code is being displayed or manipulated or when columns of numbers are viewed.

NightTune further divides the use of fonts into the following categories; default and panel.

Default fonts are used for text associated with operational description and control, including: menus, buttons, selection devices, labels, tool tips, status bar messages, and generally descriptive verbiage.

Panel fonts are used in NightTune panels, which display the data of highest importance.

Fonts are selected by querying font preferences from the following sources until a preference is found:

- Your NightTune preference
- Your NightStar-wide preference
- The system's NightTune preference
- The system's NightStar-wide preference
- NightTune's ultimate default

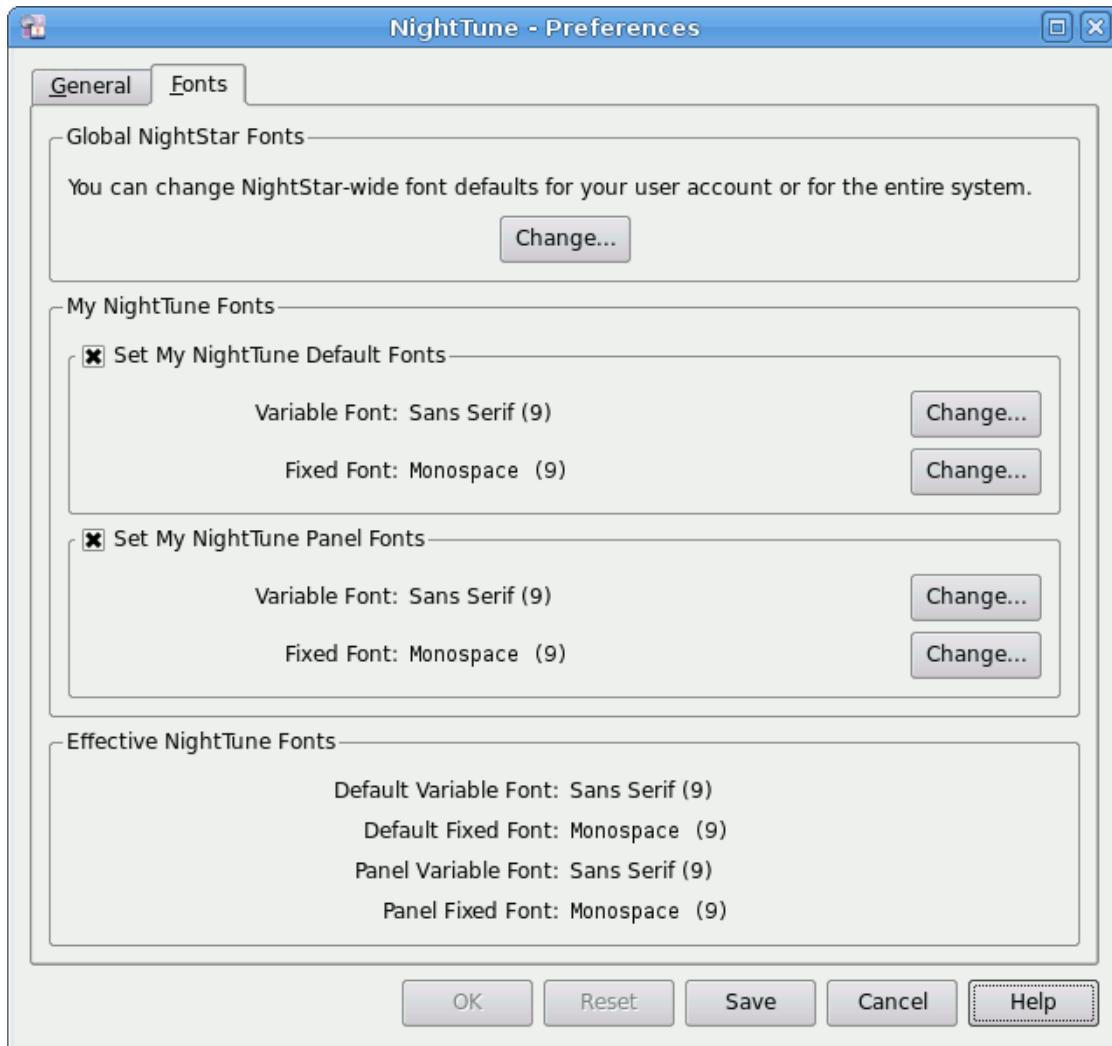


Figure 2-15. Font Preferences Page

This page is divided into three sections.

Global NightStar Fonts

The **Change...** button in this area launches the **NightStar Global Fonts** dialog which allows you to set your Nightstar-wide preferences, your preferences for another specific NightStar tool, or the system's tool or NightStar-wide preferences.

Note:

Setting a NightStar preference for the system typically requires root access.

Changes saved in the NightStar Global Fonts dialog are always saved to disk and apply to the current and subsequent NightTune invocations.

See “NightStar Global Fonts Dialog” on page 2-29 for more information.

My NightTune Fonts

This area allows you to set or clear your user’s preferences for NightTune.

Selection of the checkboxes for the individual font categories control whether or not your preferences are to be consulted. Clearing a checkbox effectively removes your user preference for that category. Setting a checkbox allows you to select specific fonts within the category.

Changes to any of the settings in this area, including individual fonts or category checkboxes, are immediately reflected in the **Effective NightTune Fonts** area at the bottom of the page so you can see the ultimate effect a change will have.

To change a specific font, ensure that the corresponding category’s checkbox is checked and then press the **Change...** button. This will launch a standard font selection dialog. When you select a font from the dialog and press **OK**, the name of the font family is displayed to the left of the **Change...** button and is displayed in the selected font as well.

Effective NightTune Fonts

This area shows you the effective fonts that will be used based on your user settings and consultation of global settings which aren’t shown in the page.

The values in this area immediately change to reflect the effective font whenever any change is made within the page.

Your changes in the **My NightTune Fonts** area are applied to the current invocation of NightTune when you press the **OK** button. However, your changes are not saved to disk and will not affect subsequent invocations of NightTune unless you press the **Save** button.

Separation of **apply** and **Save** operations make it easy to experiment with fonts in the current invocation without affecting long-term usage.

Note:

Changes to font preferences in the NightStar Global Fonts dialog are always saved to disk and apply to the current and subsequent NightTune invocations; i.e. there is no way to experiment with a global font preference without affecting subsequent NightTune invocations.

Control Buttons

The buttons at the bottom of the page control the application of your changes.

OK

Applies any changes made directly in the **Font Preferences** page to the current invocation of NightTune and closes the dialog. The changes will not be saved to disk or affect subsequent NightTune invocations unless you return to the **Preferences...** dialog and press **Save**.

Reset

Discards any changes you have made directly to the **Font Preferences** page since the dialog was launched and resets the dialog accordingly.

Note:

Changes made in the **NightTrace Global Fonts** dialog cannot be discarded via the **Reset** button.

Save

Applies the preferences from the dialog to the current invocation of NightTune, saves the preferences to disk thereby affecting subsequent NightTune invocations, and closes the dialog.

Cancel

Cancels any pending changes and closes the dialog.

Help

Opens the help system to display this section.

NightStar Global Fonts Dialog

The NightStar Global Fonts dialog allows you to set your Nightstar-wide preferences, your preferences for another specific NightStar tool, or the system's tool or NightStar-wide preferences.

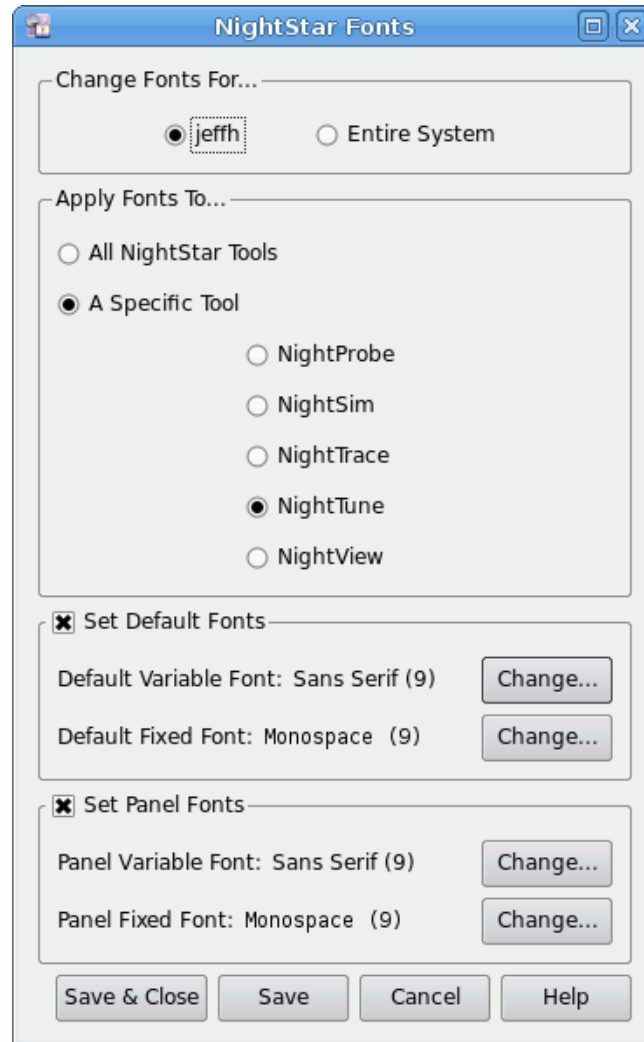


Figure 2-16. NightStar Global Fonts Dialog

Keep in mind that fonts are selected by querying font preferences from the following sources until a preference is found:

- Your NightTune preference
- Your NightStar-wide preference
- The system's NightTune preference
- The system's NightStar-wide preference
- NightTune's ultimate default

This dialog has two control areas which define the scope of font preference application.

Changes Fonts For...

By default, the dialog is set up to apply font preferences to your user account. Select the **Entire System** button if you wish to set the system's preferences.

Note:

Changing font preference for the system typically requires `root` access.

Apply Fonts To...

This area additionally controls the scope of font preference application. You can change a preference for a specific NightStar tool or change the NightStar-wide preference.

If you wish to change the font for more than one tool from this dialog, but not change the NightStar-wide preference, select the first tool of interest, make your preference change in the areas below, and then press the **Save** button. Then select the second tool of interest and repeat.

Set Default Fonts Set Panel Fonts

These areas contain the variable and fixed-width font preferences for each of the font categories, identified by the label next to each checkbox.

To remove the preferences in a category, clear its checkbox.

To change a specific font, ensure that the category's checkbox is checked and then press the **Change...** button. This will launch a standard font selection dialog. When you select a font from the dialog and press **OK**, the name of the font family is displayed to the left of the **Change...** button and is displayed in the selected font as well.

The buttons at the bottom of the page control the application of your changes.

Save & Close

Saves any changes made in this dialog to disk, thus affecting subsequent tool invocations, and closes the dialog.

These changes may affect the effective font preferences for the current invocation of NightTune. When the dialog is closed, the fonts shown in the **Effective NightTune Fonts** section of the **Preferences** dialog are updated. If you apply the changes in that dialog, they will take effect in the current invocation of NightTune.

Save

Applies the preferences from the dialog to the current invocation of NightTune, saves the preferences to disk thereby affecting subsequent NightTune invocations.

These changes may affect the effective font preferences for the current invocation of NightTune. When this dialog is subsequently closed, the fonts shown in the **Effective NightTune Fonts** section of the **Preferences** dialog are updated. If you apply the changes in that dialog, they will take effect in the current invocation of NightTune.

Cancel

Cancels any unsaved changes and closes the dialog.

Help

Opens the help system to display this section.

This chapter describes NightTune's panels, which provide functional units for displaying and modifying process and system activities. They are:

- Context Switches
- CPU Shielding and Binding
- CPU Usage
- CUDA
- CUDA Configuration
- Disk Activity
- Interrupt Activity
- Interrupt Detail Activity
- Kernel Activity
- Memory Activity
- Kernel Memory
- Physical Memory
- Swap Space
- Network Activity
- NUMA Memory
- NUMA Memory Activity
- NUMA Memory Configuration
- PCI Bus Configuration
- Process List
- Single Process Activity
- Single Process Counters

System Status Panel

Many panels are of a class called System Status Panels. These panels are divided into three panes: Text, Bar graph, and Line graph. NightTune allows the panes to be resized within the panel, but they cannot be moved or reordered in the same way that full

panels can. However, it is possible to create panels where one or two of the panes are hidden. When creating new system status panels via the **Monitor** menu, there is a sub-menu. If one of the **Text pane**, **Bar graph pane**, or **Line graph pane** items is selected, then a panel will be created with only that single pane visible. Alternately, after the creation of the panel by any means, the context menu can be used to control which panes are visible. The three menu options, **Show text pane**, **Show bar graph pane**, and **Show line graph pane** control the visibility of the **Text**, **Bar graph**, and **Line graph** panes, respectively.

In the **Text** pane, the current values for the last update are displayed numerically so that exact numbers can be seen. For values normally displayed as rates (count per second), it often is possible to reconfigure the text pane to display totals since boot, totals since a user-specified checkpoint, or raw differences in total since the last update.

In the **Bar graph** pane, the current values for the last update are displayed visually with horizontal bars. This makes it possible to compare relative values at a glance.

In the **Line graph** pane, the current values for the last update and a number of historical values are displayed. Each value is displayed along the vertical axis while time is represented by the horizontal axis. The rightmost value in the display represents the most recent value. A horizontal dotted line denotes the halfway mark of the range displayed. Vertical dotted lines denote every 20 updates. If you hover the mouse pointer over a line graph, a tooltip will display the location of the mouse pointer in terms of the value it points at and the age of the information, and also will display the value of the graph at that time.

The displayed information in system status panels is automatically refreshed periodically as controlled by the refresh interval, which can be adjusted using the **Preferences** dialog displayed from the **Preferences...** option of the **File** menu (see "Preferences" on page 2-22). The values in the **Line graph** pane automatically scroll after each refresh.

Bar and line graphs are scaled automatically by NightTune, usually based on peak values over the duration of the NightTune session. You can force NightTune to rescale the graphs based on current values using the **Rescale graphs** option from the System Status context menu. Further controls are available for rescaling line graphs using the **Line graph scale tracking** option from the System Status context menu.

System Status Context Menu

While positioned in a system status panel, right-clicking displays a context menu, which typically contains the following content:

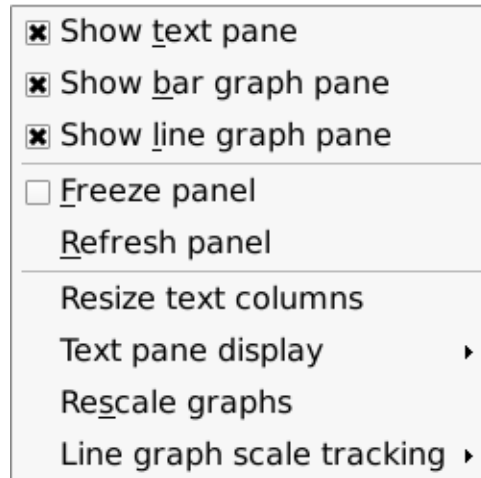


Figure 3-1. Typical Context Menu

The following paragraphs describe the menu items in detail:

Show legend

Mnemonic: G

This menu item toggles the visibility of the legend defining the colors used in the panel. Not every system status panel context menu has this item, but many do.

Show/Hide text pane

Mnemonic: T

This menu item toggles the visibility of the Text pane within the panel.

Show/Hide bar graph pane

Mnemonic: B

This menu item toggles the visibility of the Bar graph pane within the panel.

Show/Hide line graph pane

Mnemonic: L

This menu item toggles the visibility of the Line graph pane within the panel.

Freeze/Unfreeze panel

Mnemonic: F

This menu item toggles the **Freeze** setting for the panel. When frozen, data values are not refreshed automatically. This menu item overrides the **Freeze** setting for the window, but only applies to the particular panel.

Refresh panel

Mnemonic: R

This menu item causes all data within the panel to be refreshed once, regardless of the **Freeze** setting.

Resize text columns

This menu item causes NightTune to resize each text column such that it is wide enough to contain the widest value or text within that column.

Text pane display

Not every system status panel context menu has this item, but many do. Generally, they will have this item if they display any data normally shown as a rate (count per second).

This menu item displays a cascade menu with the following content:

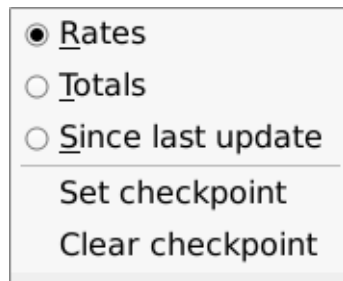


Figure 3-2. Text pane display cascade menu

The Rates, Totals, and Since last update radio buttons control how fields normally displayed as rates are displayed in the system status panel's text pane.

Rates

The information displayed in the text pane is shown as a rate (count per second). This is the default for any new panel.

Total

The information displayed in the text pane is the total count, either since the system was booted, or since the last manual checkpoint was taken. Fields which support this will be displayed with a tan background.

Since last update

The information displayed in the text pane is the difference in the count since the last update. Fields which support this will be displayed with a pink background.

Set checkpoint

This menu item causes the panel to set its checkpoint to the total values from the last update. Any totals displayed in this panel will be relative to that checkpoint.

Clear checkpoint

This menu item causes the panel to clear its checkpoint. Any totals displayed in this panel will be relative to system boot again.

Rescale graphs

Mnemonic: S

This menu item causes NightTune to rescale all graphs within the panel based on the current data for the panel.

Line graph scale tracking

This menu item displays a cascade menu with the following content:

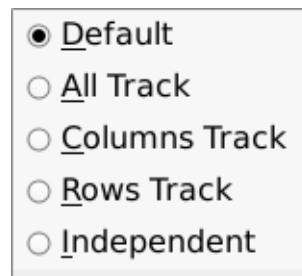


Figure 3-3. Line graph scale tracking cascade menu

The radio buttons in this menu determine how line graphs in the panel will rescale with respect to one another.

Default

The default tracking depends on the particular panel, but generally is designed for line graphs displaying the same types of information. For instance, in the Disk Usage Panel, the Usage graphs will track together, all the Ops/Sec graphs will track together, all the Sectors/Sec graphs will track together, and the two Average Time graphs will track together.

All Track

This setting causes all line graphs in the panel to rescale to a common maximum.

Columns Track

This setting causes all the rows in each column to rescale to a common maximum.

Rows Track

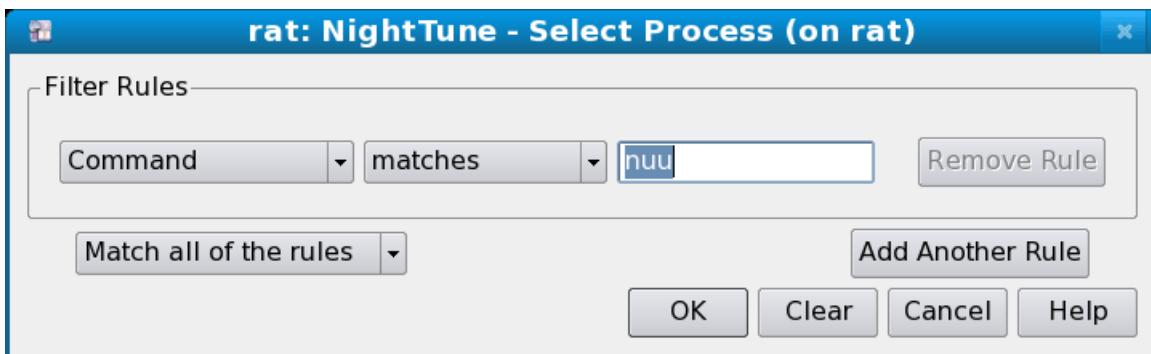
This setting causes all the columns in each row to rescale to a common maximum.

Independent

This setting causes each line graph in the panel to rescale independently of the others.

Single Process Panel

Many panels are of a class called Single Process Panels. They display information for a single user-specified process, as opposed to information about the entire system. Every single process panel has a context menu with a **Select Process** entry. Choosing this entry creates a **Select Process** dialog:



This dialog functions similarly to the **Filter Processes** dialog with some slight modifications. See “Filter Processes Dialog” on page 3-91 for more information on filtering.

The standard **Filter Processes** dialog includes some additional checkboxes which are not meaningful for selecting a single process. They are omitted from the **Select Process** dialog.

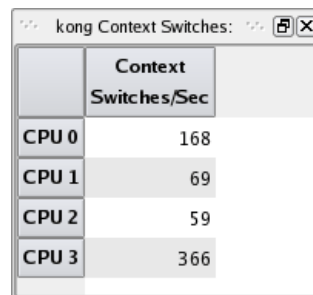
If the user-specified rules match multiple processes, then a single arbitrarily-chosen process matching the rules will be displayed in the panel.

Context Switches Panel

The **Context Switches** panel is a system status panel (see “System Status Panel” on page 3-1). Each pane displays the number of context switches per second per CPU that occurred over the period of time defined by the refresh interval.

Context Switches Text Pane

The following illustrates the **Context Switches Text** pane:



	Context Switches/Sec
CPU 0	168
CPU 1	69
CPU 2	59
CPU 3	366

Figure 3-4. Context Switches Text Pane

The information displayed in this area includes:

CPU Number

The logical CPU number is displayed in this column.

Context Switch Rate

The number of context switches per second is displayed in this column.

Context Switches Bar Graph Pane

The Context Switches Bar Graph pane provides individual bar graphs for each CPU.

The following illustrates the Context Switches Bar Graph pane:

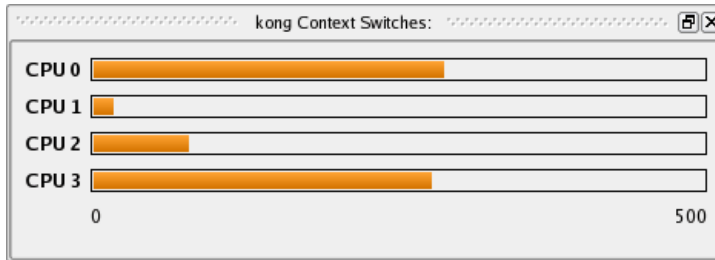


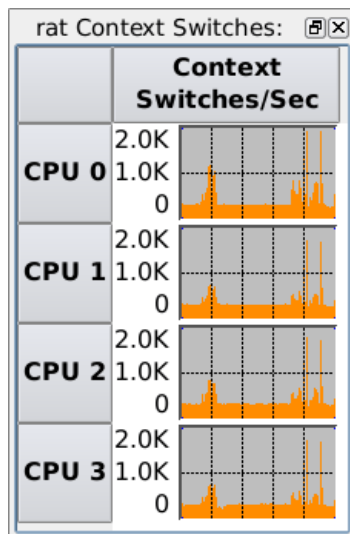
Figure 3-5. Context Switches Bar Graph Pane

The number of context switches per second is displayed as a horizontal bar for each CPU.

Context Switches Line Graph Pane

The Context Switches Line Graph pane provides individual line graphs detailing the context switch rate for each CPU.

The following illustrates the Context Switches Line Graph pane:



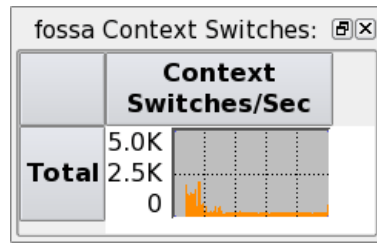


Figure 3-6. Context Switches Line Graph Pane

The number of context switches per second for a specific CPU is displayed vertically in each graph.

CPU Shielding and Binding Panel

The CPU Shielding and Binding panel displays a summary of the state of shielding, process binding, and interrupt binding on each CPU. It allows you to change the shielding attributes of CPUs and to redefine the CPU binding of specific processes or interrupts.

The following illustrates the CPU Shielding and Binding panel:

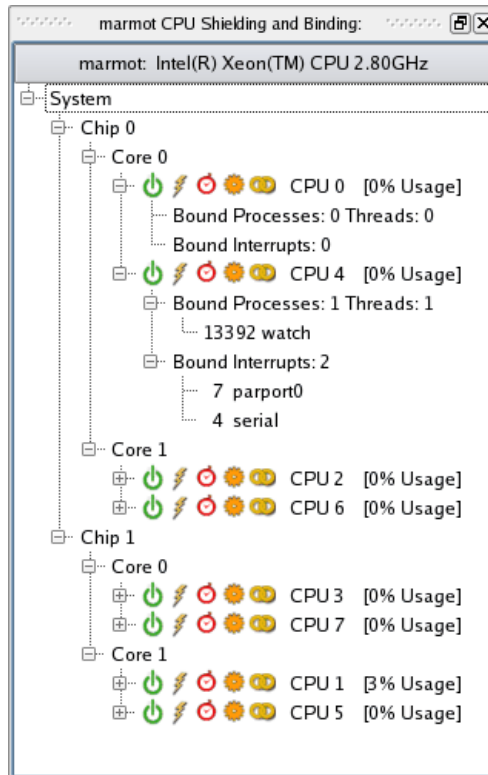


Figure 3-7. CPU Shielding and Binding Panel

All CPUs on the system are displayed in tree format. A chip typically represents a physical component within the system. On multi-core systems, each chip contains two or more cores which essentially act as independent processors. On hyper-threaded architectures, each core can be subdivided into two logical processors which behave similarly to independent processors in some ways, although they do share some logic and so execution on one processor can cause delays in execution in its sibling processor.

For each CPU, several icons represent attributes of that CPU. The symbols are described at "CPU Shielding Operations" on page 3-11. Changes to the shielding attributes of a CPU are made from the CPU Shielding dialog.

When the display is expanded, processes and interrupts that are bound to each CPU are shown. The display can be expanded by clicking on the plus sign (+) in the tree to expand the corresponding entry, or by selecting an entry, right-clicking to display the CPU Shielding and Binding panel context menu and selecting Expand All (see "CPU

Shielding and Binding Context Menu” on page 3-14). Bound kernel processes also are displayed when **Bound Kernel Processes** is activated from the Preferences dialog.

CPU Shielding Operations

The shielding attributes of a CPU can be changed with the CPU Shielding dialog.

To display the CPU Shielding dialog, right-click in the CPU Shielding and Binding panel and select the **Change Shielding...** menu item from the context menu (see “CPU Shielding and Binding Context Menu” on page 3-14).

The following illustrates the CPU Shielding dialog:

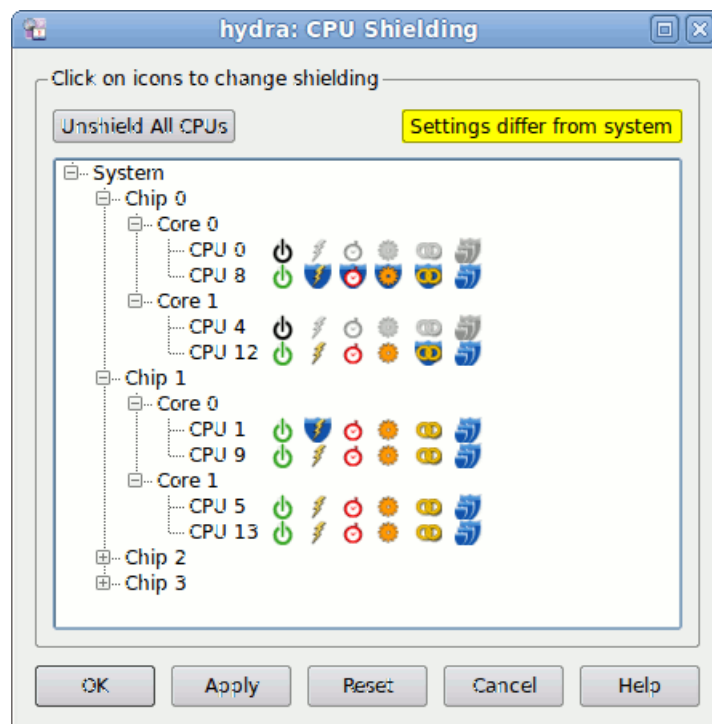


Figure 3-8. CPU Shielding Dialog

All CPUs on the system are displayed in tree format. CPUs which are hyper-threaded siblings (two hyper-threaded CPUs on the same core) are grouped together.

Each line shows the shielding attributes for one CPU.

By clicking on an icon for a CPU, the corresponding attribute is toggled and the icon display changes. If shielding is activated, a blue shield will appear for that attribute; if not activated, no shield appears.

Other changes that occur based on toggling an icon will be displayed, too. For example, if shielding a hyper-threaded CPU from hyper-threading, its sibling CPU is marked down.

To apply the changes, click on the **Apply** button at the bottom of the dialog. To apply the changes and dismiss the dialog, click on the **OK** button at the bottom of the dialog.

The following paragraphs describe the controls in detail:

Unshield All CPUs

This button changes the shielding state of all the icons so that all CPUs are completely unshielded. As with the icons in the tree, this button does not apply the changes, it merely sets the icons in the tree. You must use the **OK** or **Apply** buttons to apply the changes.

Active Toggle

The **Active** icon indicates whether the CPU is marked **Down** (black), meaning unavailable for processing; or marked **Active** (green), meaning available for processing. A common reason to mark a CPU **Down** is so that it does not interfere with processing on its sibling hyperthreaded CPU.

Interrupt Shielding Toggle

The **Interrupt Shielding** toggle indicates whether the CPU will be shielded from interrupts. Note that the **Local Timer** interrupt is not affected by this setting. When shielded, the only interrupts that will be handled by this CPU are those whose CPU affinities include only this CPU.

Local Timer Shielding Toggle

The **Local Timer Shielding** toggle indicates whether the CPU will be shielded from the **Local Timer** interrupt.

Process Shielding Toggle

The **Process Shielding** toggle indicates whether the CPU will be shielded from processes. When shielded, the only processes, including kernel daemons, allowed to execute on the CPU are those whose CPU affinities include only this CPU.

Hyper-Threading Shielding Toggle

The **Hyper-Threading Shielding** toggle indicates whether the hyper-threaded sibling CPU is marked **Active** or **Down**. When shielded, the **Active** toggle for the sibling is set to the **Down** state.

Shield All Button

The **Shield All** button enables each of the **Interrupt Shielding**, **Local Timer Shielding**, and **Process Shielding** toggles. This button is not a toggle.

The buttons at the bottom of the CPU Shielding dialog perform the following functions:

OK

Clicking **OK** attempts to apply the shielding changes to the system and closes the dialog when complete. This will fail if the user does not have sufficient capabilities; see “Capabilities” on page 1-3 for more information. Additionally, the shielding changes may conflict with current CPU usage. Attempting to set a CPU’s status to **Down** will be rejected if processes or interrupts are currently bound to the CPU. A diagnostic will be displayed if the shielding operation fails.

Apply

Clicking **Apply** attempts to apply the shielding changes to the system. This will fail if the user does not have sufficient capabilities; see “Capabilities” on page 1-3 for more information. Additionally, the shielding changes may conflict with current CPU usage. Attempting to set a CPU’s status to **Down** will be rejected if processes or interrupts are currently bound to the CPU. A diagnostic will be displayed if the shielding operation fails.

Reset

Clicking **Reset** restores the CPU Shielding dialog display with current information, discarding any changes that have not yet been applied.

Cancel

Clicking **Cancel** closes the dialog. Any changes that have not been applied are discarded.

Help

Clicking **Help** presents this section of the manual in the online help viewer.

CPU Shielding and Binding Drag and Drop Operations

To drag a process or interrupt from one CPU to another, select the items of interest under **Bound Processes** or **Bound Interrupts** using the mouse. Click and hold the mouse button, then drag the pointer to the destination area and release. Alternatively, if no processes or interrupts are selected in the area, click the mouse button anywhere on the row which describes the process or interrupt and drag to the destination area and release.

The CPU Shielding and Binding panel supports the following drag and drop operations:

- Dragging bound processes or interrupts from one CPU line to another CPU line in the CPU Shielding and Binding panel binds them to the CPU corresponding to the target line. When dragging a multi-threaded process (identified by a numeric pair enclosed in parentheses following the process name) from a CPU line to another CPU line, only the threads bound to the source CPU are rebound.

- Dragging bound processes or interrupts to the Unbind tool icon causes the processes or interrupts to be unbound from any CPUs.
- Dragging processes to the Kill tool icon causes the processes to be killed with a **SIGKILL** signal.
- Dragging a CPU to another CPU will rebind all processes and interrupts bound to the first CPU to the second.
- Dragging a CPU to the Unbind tool icon causes all processes and interrupts bound to the CPU to become unbound from any CPUs.
- Dragging a CPU to the Kill tool icon causes all processes bound to the CPU to be killed with a **SIGKILL** signal.

Note that interrupts can be dragged only if NightTune is running as `root` or if the user running NightTune has been granted the proper capabilities (see “Capabilities” on page 1-3).

CPU Shielding and Binding Context Menu

While positioned over a CPU line in the CPU Shielding and Binding panel, right-clicking displays a context menu for the associated CPU, as shown below. While positioned over a bound process or interrupt, right-clicking creates a context menu with the choices for unbinding the processes or interrupts enabled too.

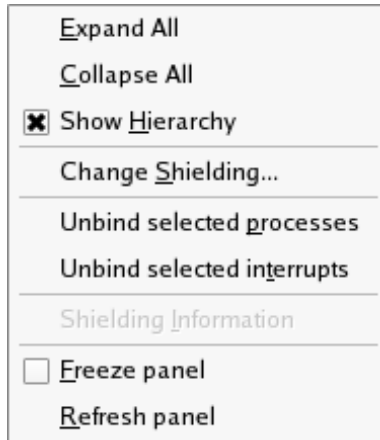


Figure 3-9. CPU Shielding and Binding Context Menu

All activities in the CPU Shielding and Binding context menu apply solely to the associated CPU, except where noted in the detailed descriptions below:

Expand All

Mnemonic: E

This menu item expands the selected item and all interior items that can be expanded.

Collapse All

Mnemonic: C

This menu item collapses the selected item and all interior items.

Show Hierarchy

Mnemonic: H

This menu item toggles the display mode of the panel. When this item is selected, CPUs are displayed in a hierarchy which indicates the relationship between CPUs and their parent chip. When not selected, the CPUs are shown in a flat list.

Clearing **Show Hierarchy** can be useful when you have a number of CPUs and need to save display space in the panel in order to see all CPUs.

Change Shielding

Mnemonic: S

This menu item displays the **CPU Shielding** dialog which allows you to change the shielding attributes of CPUs on the system. See “CPU Shielding Operations” on page 3-11 for details.

Unbind selected processes

Mnemonic: P

This menu item immediately changes the CPU affinity of the selected processes in the **Bounded Processes** list to include all CPUs; effectively unbinding them from the CPU. This item is not available on a system with only a single CPU.

Unbind selected interrupts

Mnemonic: T

This menu item immediately changes the CPU affinity of the selected interrupts in the **Bounded Interrupts** list to include all CPUs; effectively unbinding them from the CPU. This item is not available on a system with only a single CPU.

Note that interrupts can be unbound only if NightTune is running as `root` or if the user running NightTune has been granted the proper capabilities (see “Capabilities” on page 1-3).

Shielding Information

Mnemonic: I

This menu item pops up a dialog containing a textual description of the state of shielding for the selected CPU.

See the *RedHawk Linux User's Guide* for more information on CPU shielding and process and interrupt CPU affinity.

Freeze Panel

Mnemonic: F

This menu item toggles the **Freeze** setting for this panel. When frozen, this panel is not refreshed automatically.

Refresh Panel

Mnemonic: R

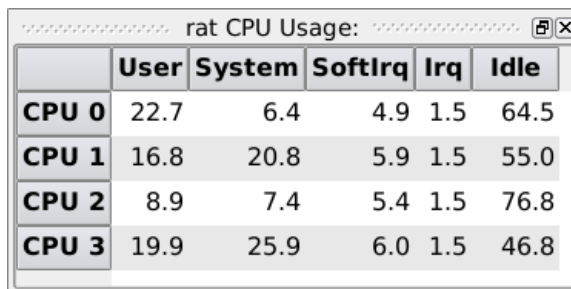
This menu item causes all data in all displays within the CPU Shielding and Binding panel to be refreshed once, regardless of the **Freeze** setting.

CPU Usage Panel

The CPU Usage panel is a system status panel (see “System Status Panel” on page 3-1). Each pane displays percentages that reflect CPU activity over the period of time defined by the refresh interval.

CPU Usage Text Pane

The following illustrates the CPU Usage Text pane:



	User	System	SoftIrq	Irq	Idle
CPU 0	22.7	6.4	4.9	1.5	64.5
CPU 1	16.8	20.8	5.9	1.5	55.0
CPU 2	8.9	7.4	5.4	1.5	76.8
CPU 3	19.9	25.9	6.0	1.5	46.8

Figure 3-10. CPU Usage Text Pane

The CPU Usage Text pane provides information on CPU usage for each CPU in the system.

The information displayed in this area includes:

CPU

The logical CPU number is displayed in this column.

User

The percentage of time the CPU was executing in user mode is displayed in this column. This excludes kernel execution but does include execution of kernel daemons which handle post-interrupt processing.

System

The percentage of time the CPU was executing in the operating system kernel is displayed in this column. This includes time spent executing system service calls on behalf of user processes as well as interrupt and machine exception processing.

SoftIrq

The percentage of time the CPU was executing IRQ utility routines by kernel threads after an interrupt handler exits.

Irq

The percentage of time the CPU was executing at interrupt level while handling an interrupt.

Idle

The percentage of time the CPU was executing the idle loop is displayed in this column. This excludes Wait time.

CPU Usage Bar Graph Pane

The CPU Usage Bar Graph pane shows the percentage of CPU time used for each CPU.

The following illustrates the CPU Usage Bar Graph pane:

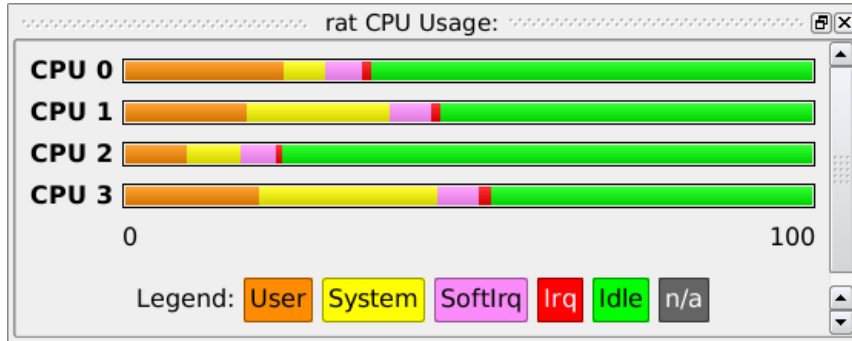


Figure 3-11. CPU Usage Bar Graph Pane

For each CPU, the User, System, Softirq, Irq, and Idle times are shown as percentages of CPU execution using color-coded, horizontal bars. See "CPU Usage Text Pane" on page 3-16 for the definitions of User, System, Wait, and Idle times.

CPU Usage Line Graph Pane

The CPU Usage Line Graph pane provides individual line graphs for User, System, Wait, and Idle times, expressed as a percentage of CPU execution for each CPU.

The following illustrates the CPU Usage Line Graph pane:

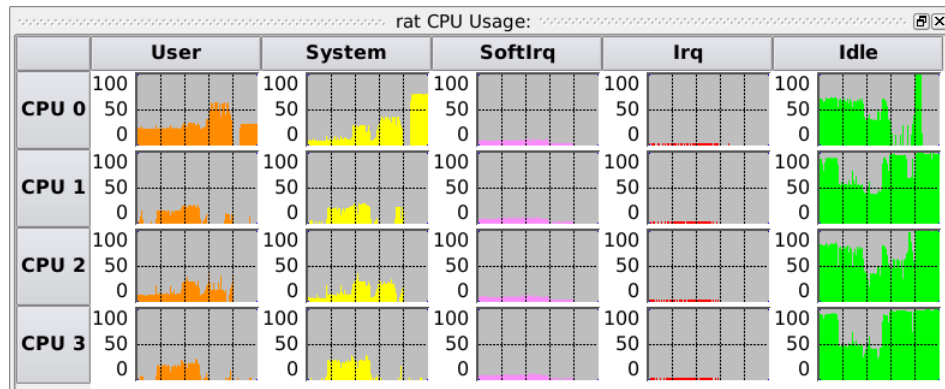


Figure 3-12. CPU Usage Line Graph Pane

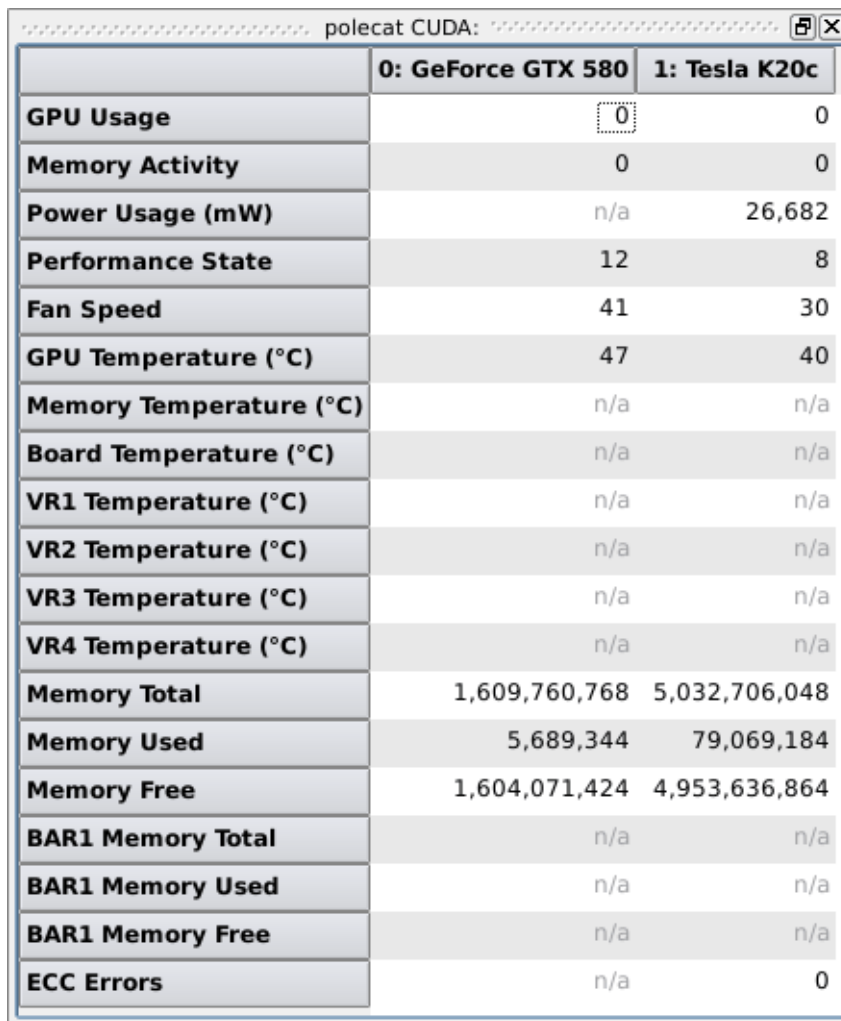
See “CPU Usage Text Pane” on page 3-16 for the definitions of User, System, SoftIrq, Irq, and Idle times. |

CUDA Panel

The CUDA panel is a system status panel (see “System Status Panel” on page 3-1). Each pane displays information about GPU usage, memory activity and usage, power usage, temperatures, and fan speeds for any nVidia CUDA devices on the system.

CUDA Text Pane

The following illustrates the CUDA Text pane:



The screenshot shows a window titled "polecat CUDA:" containing a table with the following data:

	0: GeForce GTX 580	1: Tesla K20c
GPU Usage	0	0
Memory Activity	0	0
Power Usage (mW)	n/a	26,682
Performance State	12	8
Fan Speed	41	30
GPU Temperature (°C)	47	40
Memory Temperature (°C)	n/a	n/a
Board Temperature (°C)	n/a	n/a
VR1 Temperature (°C)	n/a	n/a
VR2 Temperature (°C)	n/a	n/a
VR3 Temperature (°C)	n/a	n/a
VR4 Temperature (°C)	n/a	n/a
Memory Total	1,609,760,768	5,032,706,048
Memory Used	5,689,344	79,069,184
Memory Free	1,604,071,424	4,953,636,864
BAR1 Memory Total	n/a	n/a
BAR1 Memory Used	n/a	n/a
BAR1 Memory Free	n/a	n/a
ECC Errors	n/a	0

Figure 3-13. CUDA Text Pane

The CUDA Text pane provides information on various information for each nVidia CUDA device on the system.

The information displayed in this area includes:

GPU Usage

The percent of time in which any work has been performed by the GPU.

Memory Activity

The percent of time in which any CUDA framebuffer memory has been read or written.

Power Usage

The amount of power used by the device in mW.

Energy Consumption

The amount of energy used by the device since boot in mJ.

Performance State

The performance state of the device, where P0 represents the maximum performance state (highest clock rate), and P15 represents the minimum performance state (lowest clock rate).

Fan Speed

The speed of the GPU's fan as a percentage of full speed.

GPU Temperature

The temperature in Celsius of the CUDA GPU die.

Memory Temperature

The temperature in Celsius of the on-board memory.

Board Temperature

The temperature in Celsius of the CUDA device board.

VR1 Temperature

The temperature in Celsius of CUDA voltage regulator 1.

VR2 Temperature

The temperature in Celsius of CUDA voltage regulator 2.

VR3 Temperature

The temperature in Celsius of CUDA voltage regulator 3.

VR4 Temperature

The temperature in Celsius of CUDA voltage regulator 4.

Memory Total

The amount of total memory on the CUDA device, in bytes.

Memory Used

The amount of memory in use on the CUDA device, in bytes.

Memory Free

The amount of memory not in used on the CUDA device, in bytes.

BAR1 Memory Total

The amount of total BAR1 memory on the CUDA device, in bytes. BAR1 memory is used to map device memory so that it can be mapped directly by the host or other devices.

BAR1 Memory Used

The amount of BAR1 memory in use on the CUDA device, in bytes. BAR1 memory is used to map device memory so that it can be mapped directly by the host or other devices.

BAR1 Memory Free

The amount of BAR1 memory not in used on the CUDA device, in bytes. BAR1 memory is used to map device memory so that it can be mapped directly by the host or other devices.

ECC Errors

The total number of ECC single-bit and double-bit errors detected since boot.

PCIe Tx Throughput

The transmission (Tx) throughput on the PCIe bus for the CUDA device, in KB/s.

PCIe Rx Throughput

The receive (Rx) throughput on the PCIe bus for the CUDA device, in KB/s.

PCIe Replays

The total number of PCIe replays (retries) detected since boot.

Power Violation Time

The amount of time when the performance of the CUDA device was reduced below the configured application clock speed because of power violations, measured in ns/s.

Thermal Violation Time

The amount of time when the performance of the CUDA device was reduced below the configured application clock speed because of thermal violations, measured in ns/s.

Sync Boost Violation Time

The amount of time when the performance of the CUDA device was reduced below the configured application clock speed because of sync boost violations, measured in ns/s.

Board Limit Violation Time

The amount of time when the performance of the CUDA device was reduced below the configured application clock speed because of board limit violations, measured in ns/s.

Low Utilization Violation Time

The amount of time when the performance of the CUDA device was reduced below the configured application clock speed because of low utilization violations, measured in ns/s.

Reliability Violation Time

The amount of time when the performance of the CUDA device was reduced below the configured application clock speed because of board reliability violations, measured in ns/s.

Below Application Clock Time

The amount of time when the performance of the CUDA device was reduced below the configured application clock speed for any violation, measured in ns/s.

Below Base Clock Time

The amount of time when the performance of the CUDA device was reduced below the base clock speed for any violation, measured in ns/s.

Any data not provided by the CUDA device will be displayed as n/a.

CUDA Bar Graph Pane

The CUDA Bar Graph pane shows various information about each CUDA device on the system, arranged by type of information.

The following illustrates the CUDA Bar Graph pane:

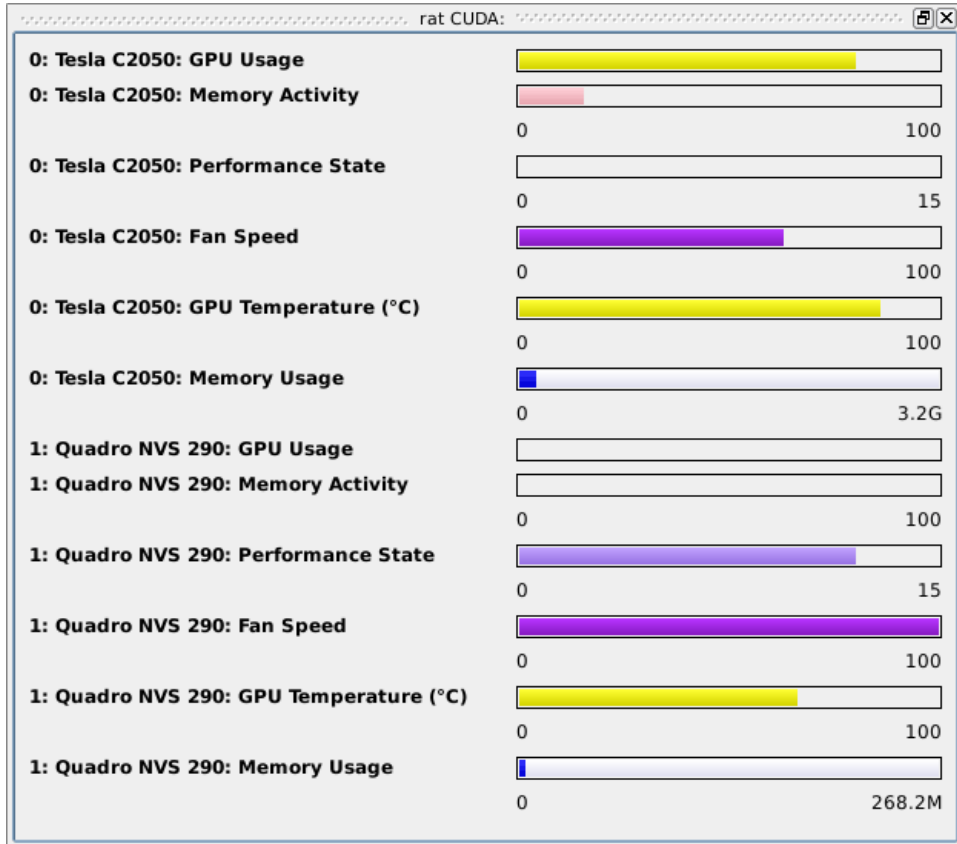


Figure 3-14. CUDA Bar Graph Pane

For each CUDA device, this pane has the capability of showing a number of bars. The exact set of bars is determined by the reporting capabilities of the nVidia device. A bar will be omitted if the device does not report any of the data displayed in the bar. The possible set of bars is:

GPU Usage

The percent of time in which any work has been performed by the GPU.

Memory Activity

The percent of time in which any CUDA framebuffer memory has been read or written.

Power Usage

The amount of power used by the device in mW.

Performance State

The performance state of the device, where 0 represents the maximum performance state (highest clock rate), and 15 represents the minimum performance state (lowest clock rate).

Fan Speed

The speed of the GPU's fan as a percentage of full speed.

GPU Temperature

The temperature in Celsius of the CUDA GPU die.

Memory Temperature

The temperature in Celsius of the on-board memory.

Board Temperature

The temperature in Celsius of the CUDA device board.

VR1 Temperature

The temperature in Celsius of CUDA voltage regulator 1.

VR2 Temperature

The temperature in Celsius of CUDA voltage regulator 2.

VR3 Temperature

The temperature in Celsius of CUDA voltage regulator 3.

VR4 Temperature

The temperature in Celsius of CUDA voltage regulator 4.

Memory Usage

The used memory and free memory as color-coded horizontal bars, each as a proportion of the memory total.

BAR1 Memory Usage

The used BAR1 memory and free BAR1 memory as color-coded horizontal bars, each as a proportion of the BAR1 memory total. BAR1 memory is used to map device memory so that it can be mapped directly by the host or other devices.

PCIe Tx Throughput

The transmission (Tx) throughput on the PCIe bus for the CUDA device, in KB/s.

PCIe Rx Throughput

The receive (Rx) throughput on the PCIe bus for the CUDA device, in KB/s.

CUDA Line Graph Pane

The CUDA Line Graph pane provides individual line graphs for each piece of information reported by each nVidia CUDA device. A bar will be omitted if the device does not report its information..

The following illustrates the CUDA Line Graph pane:

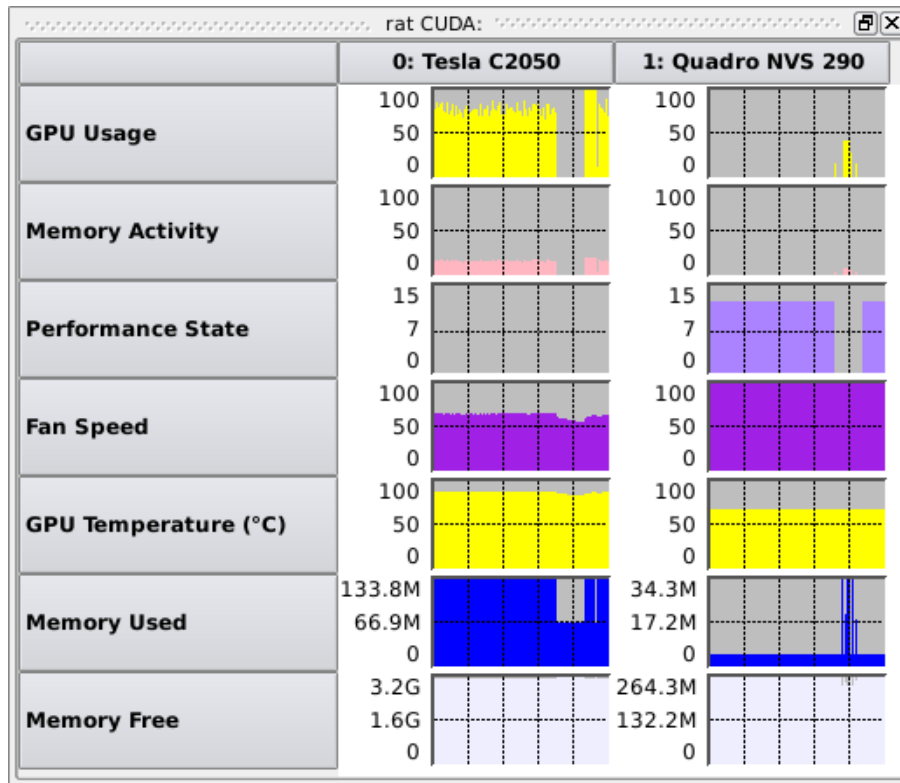


Figure 3-15. CUDA Line Graph Pane

See “CUDA Text Pane” on page 3-20 for the definitions of each piece of information available.

CUDA Configuration Panel

The CUDA Configuration panel provides configuration information for any nVidia CUDA devices on the system. Because this information is about the configuration of the system, it does not change with refresh intervals. Changes appear only when the system is reconfigured. This information is available only if a CUDA device is present on the system and if the `ccur-nvidia-cuda` package is installed.

The following illustrates the CUDA Configuration panel

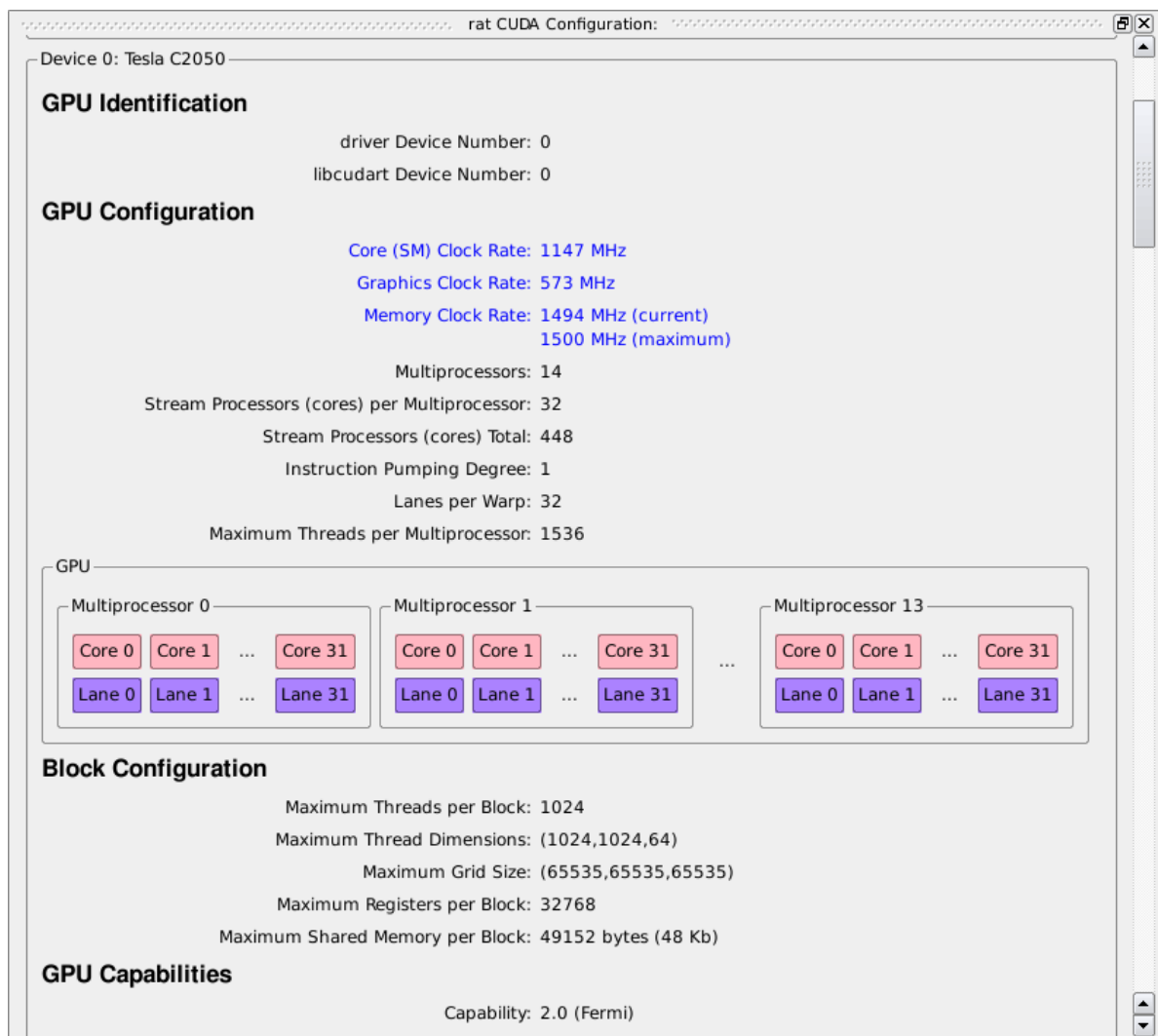


Figure 3-16. CUDA Configuration Panel

Driver & Library

The Driver & Library block describes overall information about the CUDA kernel and library provided in the **ccur-nvidia-cuda** package:

- **Kernel Driver CUDA Version** simply is the CUDA version (e.g. 5.5) of the nVidia CUDA kernel module.
- **Library Runtime CUDA Version** is the CUDA version (e.g. 5.5) of the **libcuda** library in the **ccur-nvidia-cuda** package.
- **Kernel Driver Module Version** is the driver version (e.g. 319.49) of the nVidia CUDA kernel module.
- **NVML Version** is the version (e.g. 5.319.49) of the NVML library, **libnvidia-ml**.

Devices

Subsequent blocks describe each CUDA device in turn. The information for each device is broken down into several sections.

Configurable Items

Items appearing in black are determined by the physical hardware or determined at boot time, and are not subject to change after that. Items appearing in blue can change after boot time. Some of these items (e.g. Compute Mode, Persistence Mode) can be changed by the user via the **Configure devices...** context menu item (see “Configure devices...” on page 3-39) or by command-line tools. Others may be changed internally by the hardware or driver (e.g. Power Management), and as such will not appear in the Configure Devices dialog but will still be shown in blue in the panel.

GPU Identification

There are two different number spaces used by various components that identify GPU devices. This section lists both values for the device. Sometimes they are the same values, but this is by chance and is not guaranteed.

The driver number is used by the **/proc/driver/nvidia** file system, IRQ reporting, **libnvidia-ml**, and **nvidia-smi**.

The **libcudart** number is used in CUDA API code when selecting a device (e.g. `cuda-SetDevice`).

GPU Configuration

This section describes the layout of the main physical processing units.

Logically, a CUDA workload is arranged into a *grid*. The grid is a 1- or 2-dimensional collection of *blocks* which execute either serially or in parallel with one another. Each block is arranged into a 1-, 2-, or 3-dimensional collection of individual threads. Threads in a block are considered to be executing in parallel, although some may execute serially, depending on the capabilities of the device. The following figure illustrates this hierarchy:

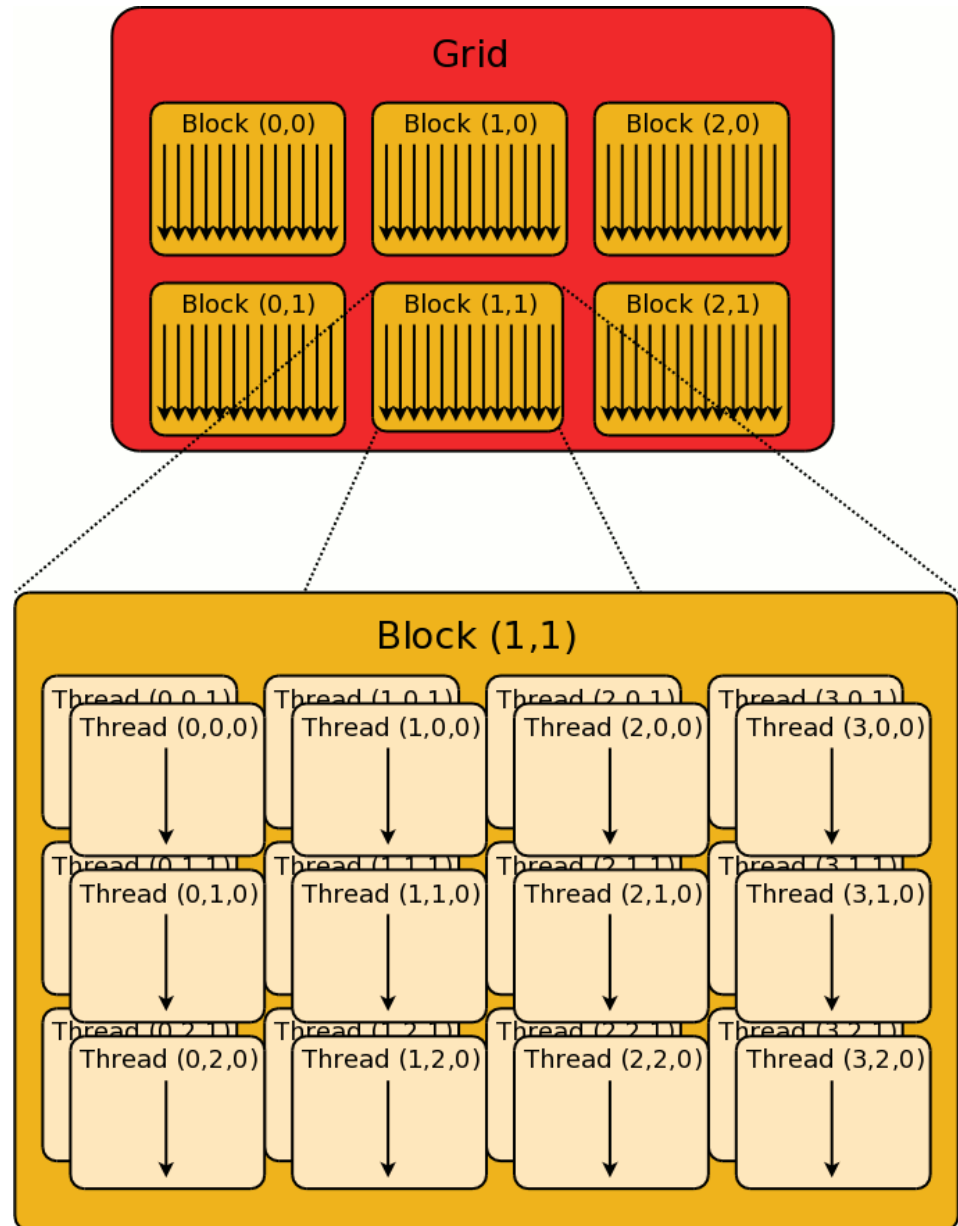


Figure 3-17. Grid, Block, and Thread Hierarchy for $\langle\langle(3,2),(4,3,2)\rangle\rangle$

Physically, a CUDA device is comprised of a number of *multiprocessors*, each of which has its own set of registers and some of its own memory. The multiprocessors operate largely independently of one another. Each multiprocessor has a number of *cores*, sometimes also called *stream processors*. A multiprocessor partitions the work of a single

block into collections of threads called *warps*. While a block may quite large, a warp has a fixed size determined by the GPU architecture (generally, a warp supports 32 threads). The multiprocessor executes the warp as a whole in a form of single-instruction-multiple-data (SIMD) execution, but with additional capabilities for handling divergent branches. The warp is considered to consist of a number of *lanes*, each of which handles one thread.

On some CUDA architectures (e.g. Kepler, Maxwell), each multiprocessor has a number of cores greater than the number of lanes in a warp, so it is capable of executing multiple warps simultaneously. On some CUDA architectures (e.g. Fermi), each multiprocessor has a number of cores equal to the number of lanes in a warp, and so executes one warp at a time. Finally, on other architectures (e.g. Tesla), the number of cores is less than the number of lanes in the warp. In that case, a single warp is handled by pumping the same instruction repeatedly through with the cores handling different lanes each time. This is done until all lanes have been executed.

In addition to the warps which are actively executing, a large number of additional threads may be resident on a multiprocessor. If a running warp stalls (e.g. due to a global memory load), it can be swapped out and a non-running warp can be swapped in. This swapping is performed by hardware and generally can be accomplished from one clock cycle to the next.

The meaning of a core varies depending on the CUDA architecture:

- On Tesla, Fermi, Kepler, Maxwell, and Pascal (compute capability 1.x - 6.x), a core is a combined FP32/INT computation unit. FP64 computations are handled by separate computation units.
- On Volta (compute capability 7.x), a core is an FP32 computation unit. INT32 and FP64 computations each are handled by separate computation units, and can be executed in parallel with the FP32 computations.

The GPU Configuration includes information on clock rates for the device. Depending on the device, this information may be complex. Three distinct clock rates are defined:

- **Core (SM) Clock Rate**, the clock rate for CUDA computations.
- **Graphics Clock Rate**, the clock rate for graphics operations.
- **Memory Clock Rate**, the clock rate for memory I/O.

For each of these clocks rates, a number of values may be reported, depending on the capabilities of the device. By default, only defined and non-redundant values are shown. But it is possible to show all values using the **Show All Clock Rates** context menu item (see “Show All Clock Rates” on page 3-39). The possible values that may be shown for each clock rate are:

- **current**, the current clock rate at this moment.
- **configured value if running an application**, the clock rate that would be adopted if an appropriate application (CUDA or graphics) were started on the device. On devices where the Applications Clock Rates are configurable, this may differ from the **default value if running an application**. This value is useful because it allows determination of the clock rate that would be used by an application even if the device currently is idle.

- **default value** if running an application, the clock rate that would have been adopted by default for an application running on the default. On devices where the Applications Clock Rates are configurable, this may differ from the configured value if running an application.
- **maximum**, the maximum clock rate allowed.

The additional information provided for the GPU Configuration includes:

- **Multiprocessors**, which is the number of multiprocessors on the CUDA device.
- **Cores per Multiprocessor**, which is the number of cores on each of the multiprocessors.
- **Cores Total**, which is the total number of cores on the whole CUDA device.
- **FP64 units per Multiprocessor**, which is the number of FP64 computation units on each of the multiprocessors.
- **FP64 units Total**, which is the total number of FP64 computation units on the whole CUDA device.
- **INT32 units per Multiprocessor**, which is the number of INT32 computation units on each of the multiprocessors. This information is present only if the INT32 computation units are separate from the cores.
- **INT32 units Total**, which is the total number of INT32 computation units on the whole CUDA device. This information is present only if the INT32 computation units are separate from the cores.
- **TensorCores per Multiprocessor**, which is the number of TensorCore computation units on each of the multiprocessors. This information is present only if the CUDA architecture supports TensorCores.
- **TensorCores Total**, which is the total number of TensorCore computation units on the whole CUDA device. This information is present only if the CUDA architecture supports TensorCores.
- **Instruction Pumping Degree**, which is the number of times each instruction must be executed serially by a multiprocessor to accommodate all the lanes in a warp. It is determined by the number of lanes in a warp and the number of cores in a multiprocessor.
- **Lanes per Warp**, which is the number of lanes in each warp.
- **Maximum Threads per Multiprocessor**, which is the maximum number of threads that can be *resident* on a multiprocessor at one time. This number may be larger than the number of threads that can *execute* on a multiprocessor at one time. Non-resident implies that resources for the thread/warp are still allocated to the multiprocessor, but the thread/warp is not executing currently (perhaps stalled due to a memory fetch in which case another warp can execute in its place).

Finally, this section in the CUDA Configuration panel shows pictorially the arrangement of multiprocessors, and the cores and lanes within each multiprocessor.

Block Configuration

This section describes limits on the sizes of blocks and grids supported by the CUDA device:

- **Maximum Threads per Block**, which is the maximum scalar number of threads per block. No matter the number of dimensions chosen, their product may not exceed this number.
- **Maximum Thread Dimensions**, which is a vector describing the maximum values allowed for each individual dimension when defining threads in a block. No dimension may exceed the value specified in the equivalent dimension of this vector.
- **Maximum Grid Size**, which is a vector describing the maximum values allowed for each individual dimension when defining blocks in a grid. No dimension may exceed the value specified in the equivalent dimension of this vector. A 1 in the z dimension indicates that no z dimension is supported.
- **Maximum Registers per Block**, which is the maximum number of 32-bit registers available to a block. The number of registers is shared by all blocks executing on a single multiprocessor.
- **Maximum Shared Memory per Block**, which is the maximum amount of shared memory available to a block. The memory is shared by all blocks executing on a single multiprocessor.
- **Maximum Shared Memory per Block Usable by Opt In**, which is the maximum amount of shared memory available to a block by special opt in. The memory is shared by all blocks executing on a single multiprocessor.

GPU Capabilities

The CUDA architecture is constantly evolving, and this section describes the capabilities of the particular CUDA device:

- **Capability**, which is a high-level indicator defined by nVidia for the set of features available on the device. In broad terms,
 - 1.x corresponds to the Tesla architecture
 - 2.x corresponds to the Fermi architecture
 - 3.x corresponds to the Kepler architecture
 - 5.x corresponds to the Maxwell architecture
 - 6.x corresponds to the Pascal architecture
 - 7.x corresponds to the Volta architecture
- **Tesla Compute Cluster (TCC) driver**, which indicates whether or not the kernel driver is a Tesla Compute Cluster (TCC) driver.

- **Supports Concurrent Memory Copy and Kernel Execution**, which indicates whether or not the device can concurrently copy memory between host and device while executing a CUDA kernel.
- **Supports Concurrent Kernel Execution**, which indicates whether or not the device supports executing multiple CUDA kernels within the same context simultaneously.
- **Stream Priorities**, which indicates whether or not the device supports specification of priorities for streams.
- **Kernel Execution Timeout Enabled**, which indicates whether or not there is a run time limit for CUDA kernels executed on the device.
- **Supports Compute Preemption**, which indicates whether or not the device supports preemption of running CUDA kernels in hardware. This differs from *software preemption* because it always is enabled, and is implemented in hardware and thus is far more efficient.
- **Supports Cooperative Launches**, which indicates whether or not the device supports launching of cooperative kernels via the `cudaLaunchCooperativeKernel` function.
- **Supports Multi-Device Cooperative Launches**, which indicates whether or not the device supports launching of cooperative kernels via the `cudaLaunchCooperativeKernelMultiDevice` function.
- **Compute Mode**, which indicates the compute mode that the device currently is in. Available modes are:
 - **Default**: Device is not restricted and multiple threads can use the device
 - **Exclusive**: Only one thread will be able to use the device
 - **Prohibited**: No threads can use the device.
- **GPU Operation Mode**, which indicates the GPU operation mode the device currently is in. Available modes are:
 - **All On**: Everything is enabled and running at full speed.
 - **Compute**: A mode designed for running only compute tasks. Graphics operations are disallowed.
 - **Low Double Precision**: A mode designed for running graphics applications without high bandwidth double precision.
- **Persistence Mode**, which indicates whether or not the driver software state is torn down (and forgotten) when the last client disconnects.
- **Clock Throttling Reasons**, which provides a list of reasons, if any, why the CUDA clock rate currently is being throttled. The reasons will be a subset of values listed in **Supported Clock Throttling Reasons**.
- **Supported Clock Throttling Reasons**, which provides a list of possible reasons why the CUDA clock rate could be throttled on this device. Possible reasons are:
 - **GPU idle**: Nothing is running on the GPU.

- `application_clocks`: A user-defined **Applications Clock Rates** setting is forcing the clocks to be throttled.
- `software_power_cap`: Software power scaling is throttling the CUDA clock rate..
- `hardware_slowdown`: The CUDA device hardware is throttling the clock. Reasons for this include the temperature being too high, the power draw being too high, or an external brake assertion triggered by the system power supply.
- **ECC Enabled**, which indicates whether or not the device has ECC (Memory Error Correcting Code) support turned on.
- **Single/Double Performance Ratio**, which is the ratio of single precision performance (in FLOPS) to double precision performance (in FLOPS).

Memory

This section describes the general Memory configuration of the CUDA device:

- **Global Memory**, which is the total amount of global memory available on the device.
- **Constant Memory**, which is total amount of constant memory available on the device.
- **Shared Memory per Multiprocessor**, which is the amount of shared memory available to each multiprocessor on the device.
- **Registers per Multiprocessor**, which is the the number of 32-bit registers available to each multiprocessor on the device.
- **Maximum Memory Pitch**, which is the maximum pitch allowed by the memory copy functions involving memory allocated by `cudaMallocPitch`.
- **Memory Bus Width**, which is the width of the memory bus in bits.
- **L1 Cache Support (global)**, which indicates whether or not the device supports L1 caching of global memory.
- **L1 Cache Support (local)**, which indicates whether or not the device supports L1 caching of local memory.
- **L2 Cache**, which is the total amount of level 2 cache on the device.

Texture Memory

This section describes limits and alignments for the special-purpose Texture Memory on the CUDA device:

- **Maximum Texture (1D)**, which is the maximum value for the one array dimension of a 1 dimensional texture.

- **Maximum Texture (2D)**, which is a vector describing the maximum values allowed for each individual dimension of a 2 dimensional texture.
- **Maximum Texture (3D)**, which is a vector describing the maximum values allowed for each individual dimension of a 3 dimensional texture.
- **Maximum Texture (Cubemap)**, which is the maximum value for all dimensions of a cubemap texture.
- **Maximum Layered Texture (1D)**, which is the maximum value for the one array dimension of a 1 dimensional layered texture, and the maximum number of layers.
- **Maximum Layered Texture (2D)**, which is a vector describing the maximum values allowed for each individual dimension of a 2 dimensional layered texture, and the maximum number of layers.
- **Maximum Layered Texture (Cubemap)**, which is the maximum value for all dimensions of a layered cubemap texture, and the maximum number of layers.
- **Maximum Linear Texture (1D)**, which is the maximum value for the one array dimension of a 1 dimensional texture bound to linear memory.
- **Maximum Linear Texture (2D)**, which is a vector describing the maximum values allowed for each individual dimension of a 2 dimensional texture bound to linear memory.
- **Maximum Texture, Gathered (2D)**, which is a vector describing the maximum values allowed for each individual dimension of a 2 dimensional texture if gather operations are performed on it.
- **Maximum Mipmapped Texture (1D)**, which is the maximum value for the one array dimension of a 1 dimensional mipmapped texture.
- **Maximum Mipmapped Texture (2D)**, which is a vector describing the maximum values allowed for each individual dimension of a 2 dimensional mipmapped texture.
- **Texture Alignment**, which is the alignment requirement such that texture base addresses need no offset applied to texture fetches.
- **Texture Pitch Alignment**, which is the alignment requirement for 2D texture references to pitched memory.
- **Maximum Surface (1D)**, which is the maximum value for the one array dimension of a 1 dimensional surface.
- **Maximum Surface (2D)**, which is a vector describing the maximum values allowed for each individual dimension of a 2 dimensional surface.
- **Maximum Surface (3D)**, which is a vector describing the maximum values allowed for each individual dimension of a 3 dimensional surface.
- **Maximum Surface (Cubemap)**, which is the maximum value for all dimensions of a cubemap surface.

- **Maximum Layered Surface (1D)**, which is the maximum value for the one array dimension of a 1 dimensional layered surface, and the maximum number of layers.
- **Maximum Layered Surface (2D)**, which is a vector describing the maximum values allowed for each individual dimension of a 2 dimensional layered surface, and the maximum number of layers.
- **Maximum Layered Surface (Cubemap)**, which is the maximum value for all dimensions of a layered cubemap surface, and the maximum number of layers.
- **Surface Alignment**, which is the alignment requirement for surfaces.

Host/Remote Memory Capabilities

- **Supports Mapping Host Memory**, which indicates whether or not the device can map host memory directly into the CUDA address space (thus avoiding memory copies).
- **Unified Addressing**, which indicates whether or not the device and host support sharing a common unified address space. Note that this does not imply that they share physical memory, just that the address spaces are synchronized, so that a pointer value in one corresponds to equivalent copied memory in the other.
- **Device can use Host Pointers for Registered Memory**, which indicates whether or not the device can use host pointers for memory registered via the `cudaHostRegister` function.
- **Managed Memory**, which indicates whether or not the device provides hardware support for allocation of managed memory. Managed memory allows automatic moving of memory between host and device without user interventions. Note that managed memory can be used even if this value is false, but then it is implemented by software.
- **Pageable Memory**, which indicates whether or not the device supports pageable memory, even when `cudaHostRegister` was not called for it. That is, the device can allocate virtual memory not yet backed by any physical memory. If the memory is accessed and has no physical memory backing it, a page fault occurs, and the device migrates the memory on a per-page basis from the host. This also means that it is possible to oversubscribe CUDA memory beyond the physical memory of the device.
- **Concurrent Managed Access**, which indicates whether or not the device and the CPU can maintain page-level coherency of memory shared between them.
- **Host/Device Link Supports Native Atomics**, which indicates whether or not the link between host and device (e.g. NVLINK, PCIe, etc.) supports native atomic operations. If this is not supported, atomics using host memory must be implemented with software assistance.
- **RDMA Peer Access to**, which indicates whether or not this device can access directly another device's memory via RDMA. If more than two devices are present, an indication is provided for each other device.

- **RDMA Peer Access from**, which indicates whether or not another device can access directly this device's memory via RDMA. If more than two devices are present, an indication is provided for each other device.

Hardware Information

This section describes hardware information about the CUDA device:

- **PCI Domain:Bus:Device ID**, which is the PCI domain:bus:device ID (in the same sense as `lspci (1)`) of the device.
- **PCIe Link**, which is PCIe link generation and bit width (e.g. PCIe 2x16) of the device. If this is less than the maximum supported by the device because of how it is installed or configured, then the maximum generation and bit width also is displayed.
- **Serial Number**, which is the serial number of the device.
- **Globally Unique Identifier (UUID)**, which is the globally unique identifier of the device.
- **Board Id**, which is the board ID number for a multi-board device. Multiple devices with the same board ID are on the same board. These numbers are assigned arbitrarily by the driver, and so are not guaranteed to be consistent across reboots.
- **VBIOS Version**, which is the version number of the VBIOS (Video BIOS).
- **infoROM Board Part Number**, which is the Board Part Number from the infoROM flashed on the device.
- **infoROM Image Version**, which is the version number of the infoROM flashed on the device.
- **infoROM OEM Version**, which is the version number of the infoROM OEM data structures on the device.
- **infoROM ECC Version**, which is the version number of the infoROM ECC data structures on the device.
- **infoROM Power Management Version**, which is the version number of the infoROM Power Management data structures on the device.
- **Bridge Chips**, which is a description of each bridge chip on the GPU board, and some additional information about them.
- **Multi-GPU Board**, which indicates whether or not the device is on a multi-GPU board. If so, it provides an identifier which is common to all devices which share the same board.
- **Integrated on Motherboard**, which indicates whether or not the device is an integrated (motherboard) GPU.
- **Display Active**, which indicates whether or not a display is initialized using the device. Generally, this means that an X Server application is

using the device. This does not necessarily mean that a monitor is physically attached to the display.

- **Display Connected**, which indicates whether or not a physical device (e.g. a monitor) is connected to any of the devices connectors.
- **Power Management Mode**, which indicates whether or not a power management algorithm is enabled for the device.
- **Power Management Limit**, which is the power limit at which a power management algorithm would take effect. It also displays the default limit and valid range for the limit. This is only meaningful if **Power Management Mode** is true.
- **Enforced Power Limit**, which is the power limit that the driver enforces after taking into account all limiters. Limiters include the **Power Management Limit**, and the out of band limit interface.
- **ECC Mode**, which indicates whether or not error correcting codes (ECC) are enabled.
- **Temperature Threshold: shutdown**, which is the temperature which, if exceeded, causes the GPU to be shut down to protect the hardware.
- **Temperature Threshold: slowdown**, which is the temperature which, if exceeded, causes the GPU to be slowed down.
- **Temperature Threshold: memory slowdown**, which is the memory temperature which, if exceeded, causes the GPU to be slowed down.
- **Temperature Threshold: GPU slowdown**, which is the GPU temperature which, if exceeded, causes the GPU to be reduced below its base clock speed.

CUDA Configuration Panel Context Menu

While positioned in the CUDA Configuration panel, right-clicking displays the CUDA Configuration context menu:

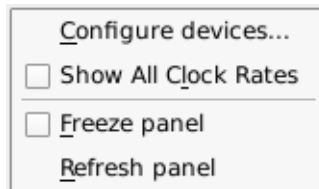


Figure 3-18. CUDA Configuration Context Menu

Many items are common to all context menus (see “System Status Context Menu” on page 3-3), but some new menu items are present in this menu:

Configure devices...

Mnemonic: C

This menu item displays the **CUDA Configure Devices** dialog which allows you to change some configuration settings on CUDA devices (see “CUDA Devices Configuration Dialog” on page 3-39).

Show All Clock Rates

Mnemonic: L

This menu item toggles the **Show All Clock Rates** setting for the panel. Normally, the **Core (SM) Clock Rate**, **Graphics Clock Rate**, and **Memory Clock Rate** items (see 3-30) show only known and non-redundant values. However, if this is checked, they will show all values.

Freeze/Unfreeze panel

Mnemonic: F

This menu item toggles the **Freeze** setting for the panel. When frozen, data values are not refreshed automatically. This menu item overrides the **Freeze** setting for the window, but only applies to the particular panel.

Refresh panel

Mnemonic: R

This menu item causes all data within the panel to be refreshed once, regardless of the **Freeze** setting.

CUDA Devices Configuration Dialog

The **CUDA Devices Configuration** dialog allows changing some configuration settings for CUDA devices on the system. Different CUDA devices support different sets of

configurable settings. For any setting not supported by a device, it will be disabled (greyed out).

The following illustrates the CUDA Devices Configuration dialog:

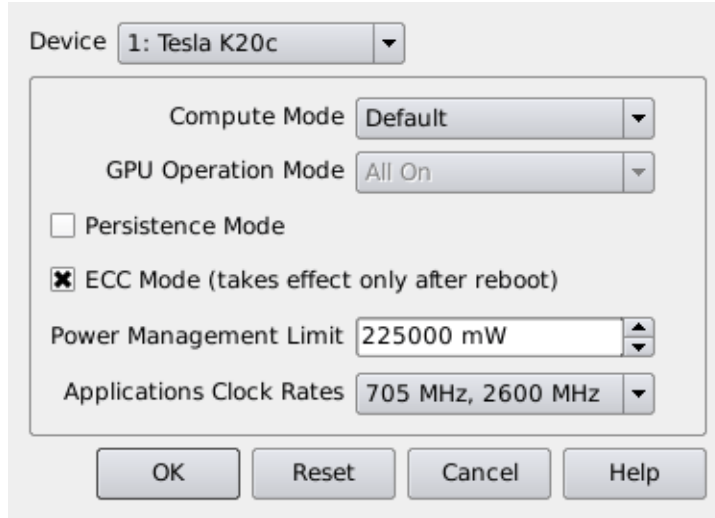


Figure 3-19. CUDA Devices Configuration Dialog

The dialog consists of the following functional labels and controls:

Device

This combo box allows the user to select which device is to be configured.

Compute Mode

This allows changing the Compute Mode (see 3-33).

GPU Operation Mode

This allows changing the GPU Operation Mode (see 3-33).

Persistence Mode

This allows changing the Persistence Mode (see 3-33).

ECC Mode

This allows changing the ECC Mode (see 3-38). Note that changing this value does not have immediate effect. It will only take effect after a reboot.

Power Management Limit

This allows changing the Power Management Limit (see 3-38).

Applications Clock Rates

This allows changing the Applications Clock Rates. These are displayed as values for each of the **Core (SM) Clock Rate**, **Graphics Clock Rate**, and **Memory Clock Rate** items (see 3-30). The **Applications Clock Rates** determine the clock rates that will be used when running an application. They will not be enforced at other times, such as when the GPU is idle. The selectable items show all allowed combinations of clock rates. For each selectable item, the first clock rate affects both the **Core (SM) Clock Rate** and the **Graphics Clock Rate**; and the second clock rate affects the **Memory Clock Rate**.

OK

Pressing the **OK** button applies the CUDA configuration changes in the dialog, and then closes the dialog.

Reset

Pressing the **Reset** button discards any changes not yet applied and refreshes the displayed values to the current configuration of the devices.

Cancel

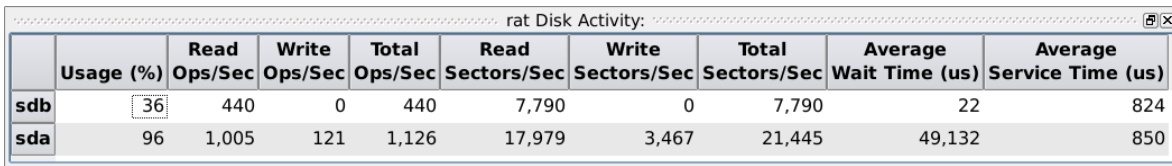
Pressing the **Cancel** button discards any changes not yet applied, and closes the dialog.

Disk Activity Panel

The Disk Activity panel is a system status panel (see “System Status Panel” on page 3-1). Each pane displays statistics related to disk operations that occurred over the period of time defined by the refresh interval. Statistics for individual disks are displayed.

Disk Activity Text Pane

The following illustrates the Disk Activity Text pane:



	Usage (%)	Read Ops/Sec	Write Ops/Sec	Total Ops/Sec	Read Sectors/Sec	Write Sectors/Sec	Total Sectors/Sec	Average Wait Time (us)	Average Service Time (us)
sdb	36	440	0	440	7,790	0	7,790	22	824
sda	96	1,005	121	1,126	17,979	3,467	21,445	49,132	850

Figure 3-20. Disk Activity Text Pane

The information displayed in this area includes:

Disk ID

The disk device name is displayed in this column.

Usage

The percentage of time the disk was busy servicing a transfer request is displayed in this column.

Read Operations per Second

The number of read transfer requests serviced by the disk per second is displayed in this column.

Write Operations per Second

The number of write transfer requests serviced by the disk per second is displayed in this column.

Total Operations per Second

The number of read and write transfer requests serviced by the disk per second is displayed in this column.

Read Sectors per Second

The number of sectors transferred from the disk per second is displayed in this column.

Write Sectors per Second

The number of sectors transferred to the disk per second is displayed in this column.

Total Sectors per Second

The number of sectors transferred to or from the disk per second is displayed in this column.

Average Wait Time

The average wait time, in microseconds, that transfer requests waited on the queue before being serviced is displayed in this column.

Average Service Time

The average service transfer time, in microseconds, is displayed in this column. The time includes seek time, rotational latency, and data transfer time.

Disk Activity Bar Graph Pane

The Disk Activity Bar Graph pane provides individual bar graphs detailing metrics related to disk operations.

The following illustrates the Disk Activity Bar Graph pane:

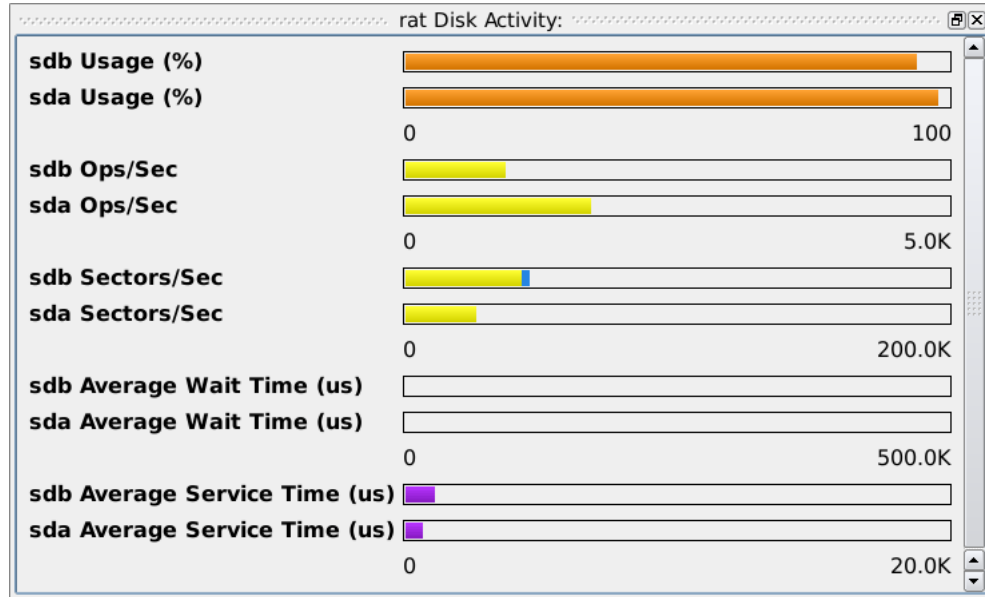


Figure 3-21. Disk Activity Bar Graph Pane

Individual bar graphs are shown for each disk for each metric. See “Disk Activity Text Pane” on page 3-42 for definitions of these metrics. The values of the metrics are displayed horizontally in each graph. In the case of Operations per Second and Sector per Second, the reads and writes are shown on the same graph as each other, with the reads appearing in yellow and the writes appearing in blue.

Disk Activity Line Graph Pane

The Disk Activity Line Graph pane provides individual line graphs detailing metrics related to disk operations.

The following illustrates the Disk Activity Line Graph pane:

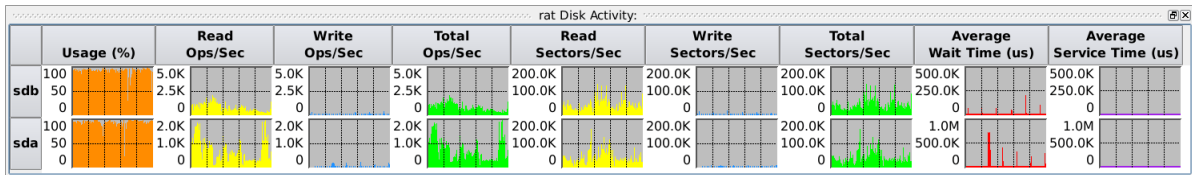


Figure 3-22. Disk Activity Line Graph Pane

Individual line graphs are shown for each disk for each metric. The value of each metric is displayed vertically in each graph. See “Disk Activity Text Pane” on page 3-42 for definitions of the metrics.

Interrupt Activity Panel and Interrupt Activity Detail Panel

The Interrupt Activity and Interrupt Detail Activity panels are system status panels (see “System Status Panel” on page 3-1). Each pane displays interrupt activity on the system. They differ from each other in whether or not they display interrupt activity individually for each CPU, or as a total across all CPUs. The Interrupt Detail Activity panel may be more desirable because of its distinction between CPUs, but this can grow cumbersome and significantly reduce NightTune’s interactive response on systems with large

numbers of CPUs and large numbers of interrupts. In such cases, the Interrupt Activity panel may be preferable.

Interrupt (Detail) Activity Text Pane

The following illustrates the Interrupt Detail Activity Text pane:

rat Interrupt Detail Activity (Interrupts/Second):					
CPUS					
	0	1	2	3	Description
1 i8042	0	0	0	0	i8042
8 rtc0	0	0	0	0	rtc0
9 acpi	0	0	0	0	acpi
12 i8042	0	0	0	0	i8042
16 usb1	0	0	0	0	ehci_hcd:usb1
17 rcim, ...	0	0	0	0	rcim, hda_intel
21 ata_piix, ...	5	5	0	0	ata_piix, ata_piix
23 usb2	0	0	0	0	ehci_hcd:usb2
40 eth0	105	105	0	0	eth0
41 hda_intel	0	0	0	0	hda_intel
NMI	0	0	0	0	Non-maskable interrupts
LOC	309	320	0	0	Local timer interrupts
SPU	0	0	0	0	Spurious interrupts
PMI	0	0	0	0	Performance monitoring interrupts
PND	0	0	0	0	Performance pending work
RES	0	0	0	0	Rescheduling interrupts
CAL	0	0	0	0	Function call interrupts
TLB	0	0	0	0	TLB shutdowns
KTLB	0	0	0	0	kernel space TLB shutdowns
MOV	0	0	0	0	IRQ-move cleanup interrupts
TRM	0	0	0	0	Thermal event interrupts
THR	0	0	0	0	Threshold APIC interrupts
MCE	0	0	0	0	Machine check exceptions
MCP	0	0	0	0	Machine check polls
SMI	0	0	0	0	system management interrupts while idle
ERR	0				
MIS	0				

Legend: Unshielded Shielded Inactive Bound

Figure 3-23. Interrupt Detail Activity Text Pane


The Interrupt Detail Activity Text pane provides information on individual interrupts and allows the user to change an interrupt's CPU affinity.

The information displayed in this area includes:

Interrupt Value

The IRQ value or mnemonic for the interrupt is displayed.

Interrupts Per Second

For each CPU, the number of interrupts per second is displayed. For interrupts which have a CPU affinity which does not include all CPUs, a shield icon  is displayed for that interrupt in columns associated with CPUs in its CPU affinity mask. Thus, at a glance you can determine which interrupts have an affinity with which CPUs.

For CPUs which are shielded from interrupts, the column header for that CPU will have a blue shield.

The Interrupt Activity Text pane looks similar but has only a single numeric column for all CPUs.

Interrupt Description

For interrupts associated with devices, the device name is displayed. For other interrupts, a short functional description of their purpose is displayed.

Interrupt Control Drag and Drop Operations

Individual interrupts can be dragged onto various destination panels and drop targets.

To drag an interrupt, click anywhere on the interrupt row and drag the pointer to the destination area and release.

- Dragging an interrupt onto the Unbind tool icon causes the interrupt to be unbound from any CPUs.
- Dragging an interrupt onto a CPU in the CPU Shielding and Binding panel (see “CPU Shielding and Binding Panel” on page 3-10) will bind it to that CPU.

Interrupt (Detail) Activity Bar Graph Pane

The Interrupt Detail Activity Bar Graph pane provides individual bar graphs showing the number of interrupts per second per CPU.

The following illustrates the Interrupt Detail Activity Bar Graph pane:

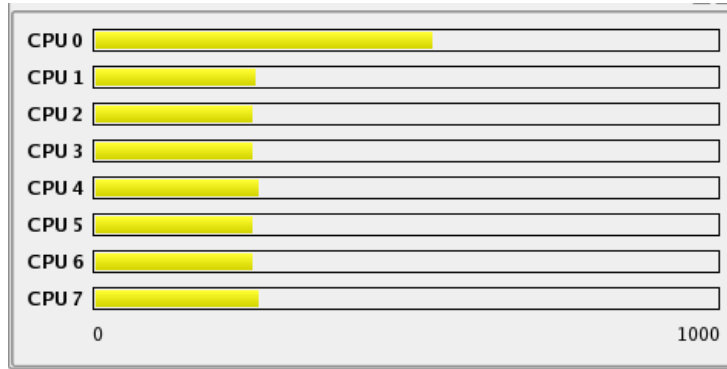


Figure 3-24. Interrupt Detail Activity Bar Graph Pane

Regardless of which interrupts are displayed in the Interrupt Activity Text and Interrupt Activity Line Graph panes, these bar graphs represent **all** interrupt activity on the system.

The Interrupt Activity Text pane looks similar but has only a single bar for all CPUs.

Interrupt (Detail) Activity Line Graph Pane

The Interrupt Detail Activity Line Graph pane provides individual line graphs for each interrupt for each CPU. The Interrupt Activity Line Graph pane looks similar but has only a single column for all CPUs.

The following illustrates the Interrupt Detail Activity Line Graph pane:

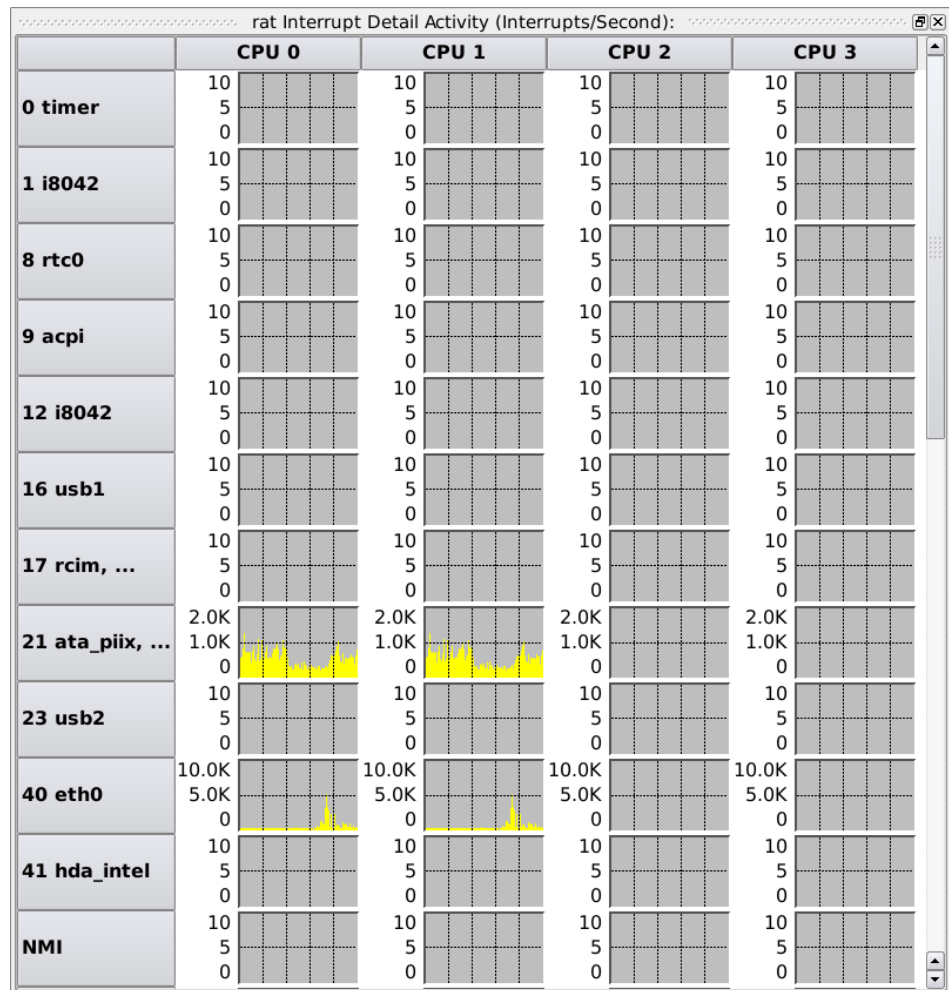


Figure 3-25. Interrupt Activity Line Graph Pane

The number of interrupts per second is displayed in each graph.

Interrupt (Detail) Activity Context Menu

While positioned in the Interrupt Activity panel, right-clicking displays the Interrupt Activity context menu:

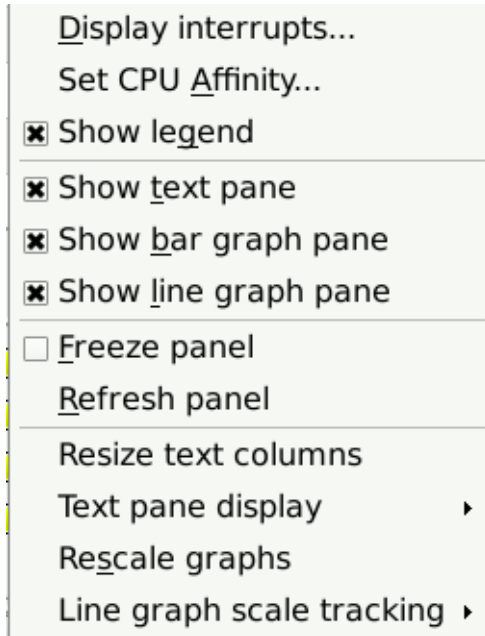


Figure 3-26. Interrupt Activity Context Menu

Many items are common to all system status context menus (see “System Status Context Menu” on page 3-3), but some new menu items are present in this menu:

Display Interrupts...

Mnemonic: D

This menu item displays the Interrupt Selection dialog which allows you to select which interrupts will be displayed in the Interrupt Activity Text and Interrupt Activity Line Graph panes in the panel. This menu item does not affect the interrupt activity as shown in the Interrupt Activity Bar Graph pane.

Set CPU Affinity...

Mnemonic: A

This menu item displays the Interrupt Affinity dialog which allows you to view and change the affinity of the selected interrupt. See “Interrupt Affinity Dialog” on page 3-51.

Interrupt Affinity Dialog

The Interrupt Affinity dialog displays the current CPU affinity of a specific interrupt and allows you to change it. It is accessed by right-clicking on the Interrupt Activity Text pane and selecting Set CPU Affinity... from the context menu.

The following illustrates the Interrupt Affinity dialog:

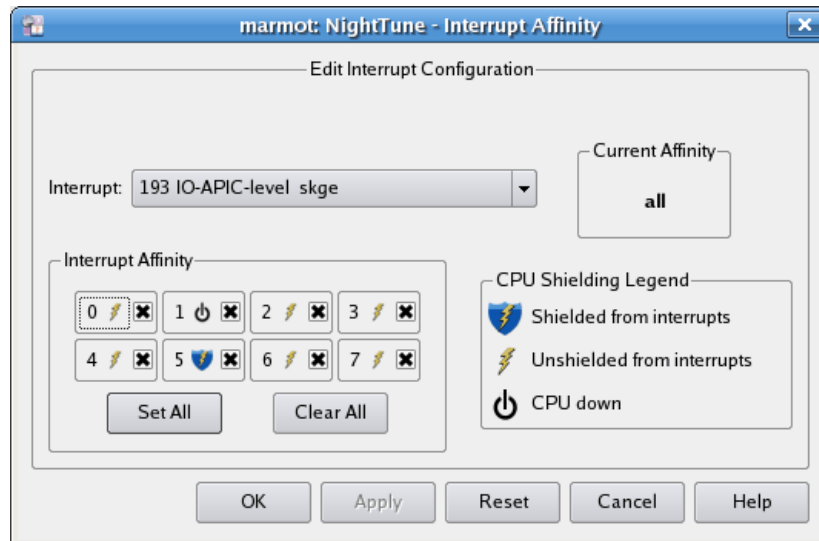


Figure 3-27. Interrupt Affinity Dialog


The dialog consists of the following functional labels and controls:

Interrupt

The IRQ value, name, and description currently associated with the dialog is displayed. A drop-down list allows you to select a different interrupt.

Interrupt Affinity

The Interrupt Affinity area allows you to specify individual CPUs where the interrupt can be processed. If a single CPU checkbox is checked, the interrupt is bound to that CPU and the interrupt is displayed in the Bound Interrupts list for the associated CPU in any CPU Shielding and Binding panel.

For each CPU that has been shielded from interrupts, the ‘shielded from interrupts’ icon  is displayed. If a CPU has been shielded from interrupts, no interrupt will be processed on that CPU unless its affinity includes that CPU and includes no non-shielded CPUs.

The Set All and Clear All buttons make it possible to set or clear all CPU checkboxes quickly without having to click on each one individually.

Current Affinity

This area displays the affinity currently in effect for the interrupt as a hexadecimal mask or as the word **all**.

CPU Shielding Legend

This area displays the symbols used to indicate the status of the CPUs on the system: shielded from interrupts, unshielded from interrupts, and CPU down.

OK

Clicking **OK** applies the CPU affinity selected in the dialog to the interrupt, and then closes the dialog. If the user lacks the capabilities to change the CPU affinity, the **OK** button will be desensitized. See “Capabilities” on page 1-3 for more information.

Apply

Clicking the **Apply** button applies the CPU affinity selected in the dialog to the interrupt. If the user lacks the capabilities to change the CPU affinity, the **Apply** button will be desensitized. See “Capabilities” on page 1-3 for more information.

Reset

Clicking the **Reset** button discards any changes not yet applied and refreshes the displayed values to the current affinity of the interrupt.

Cancel

Clicking the **Cancel** button closes the dialog; any changes made to the dialog that have not been applied are discarded.

Help

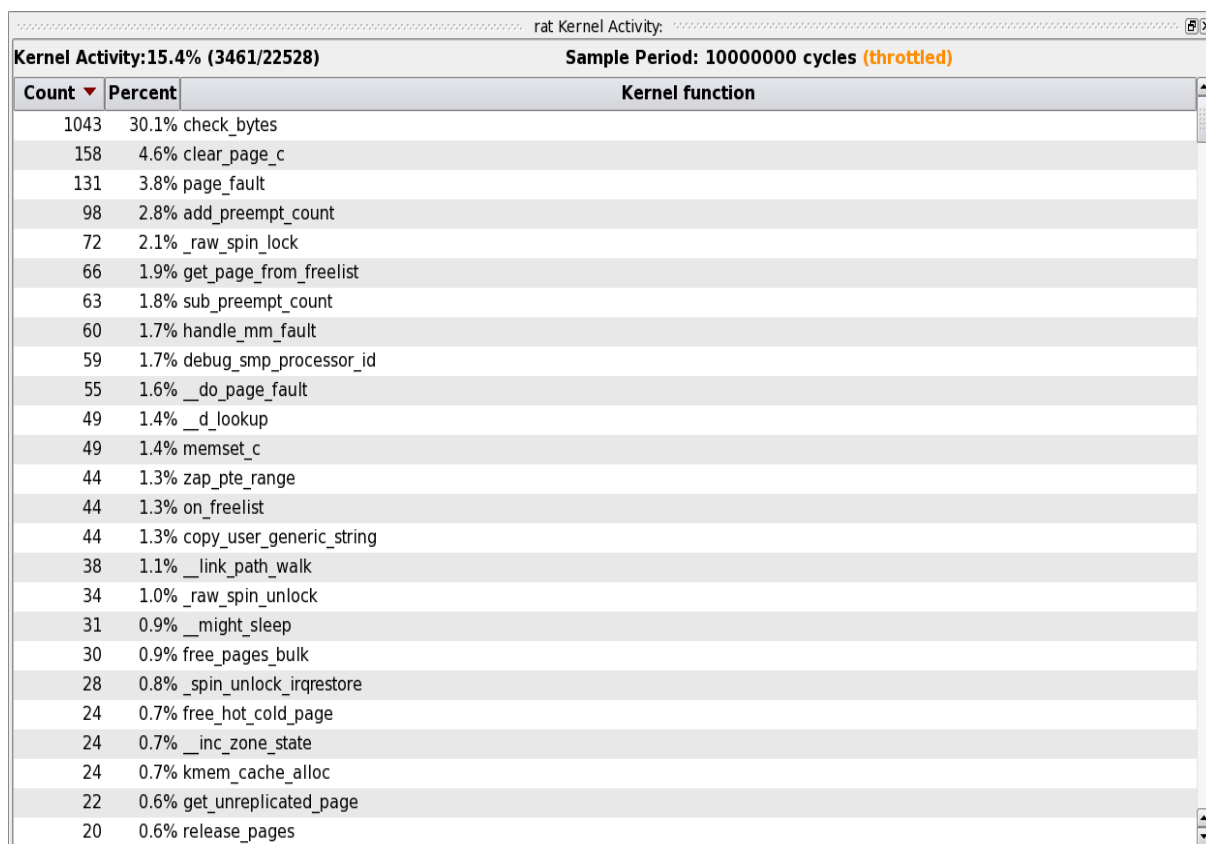
Clicking the **Help** button displays this section in the online help viewer.

Kernel Activity Panel

The Kernel Activity panel displays information about the amount of time the system kernel spends within each kernel function. It determines this information by taking a snapshot of the location where the kernel is executing periodically with a user-specified period (see “Sample Period” on page 2-24) measured in CPU cycles. It displays this information sorted by number of hits for each function which is found to be executing.

This panel depends on performance counter support on both the CPU and the Linux kernel. Read the sub-section on “Required Privileges for the Kernel Activity Panel” on page 3-55 for important information.

The following illustrates the panel:



The screenshot shows a window titled "rat Kernel Activity:" with a header bar. The header bar contains "Kernel Activity:15.4% (3461/22528)" and "Sample Period: 10000000 cycles (throttled)". Below the header is a table with three columns: "Count", "Percent", and "Kernel function". The table lists 20 kernel functions, sorted by count in descending order.

Count	Percent	Kernel function
1043	30.1%	check_bytes
158	4.6%	clear_page_c
131	3.8%	page_fault
98	2.8%	add_preempt_count
72	2.1%	_raw_spin_lock
66	1.9%	get_page_from_freelist
63	1.8%	sub_preempt_count
60	1.7%	handle_mm_fault
59	1.7%	debug_smp_processor_id
55	1.6%	_do_page_fault
49	1.4%	_d_lookup
49	1.4%	memset_c
44	1.3%	zap_pte_range
44	1.3%	on_freelist
44	1.3%	copy_user_generic_string
38	1.1%	_link_path_walk
34	1.0%	_raw_spin_unlock
31	0.9%	_might_sleep
30	0.9%	free_pages_bulk
28	0.8%	_spin_unlock_irqrestore
24	0.7%	free_hot_cold_page
24	0.7%	_inc_zone_state
24	0.7%	kmem_cache_alloc
22	0.6%	get_unreplicated_page
20	0.6%	release_pages

Figure 3-28. Kernel Activity Panel

The header indicates the Kernel Activity, which is the percentage of time spent in the system kernel, as opposed to a user application. It also displays the user-specified Sample Period, measured in CPU cycles. If the sample period is set too low, the kernel may throttle the number of snapshots taken, artificially increasing the sample period beyond the user-specified number. If this happens, the header will display (throttled). Finally, if any errors occurred, the most recent will be displayed in the header. Any previous errors can be viewed in a tooltip by hovering the mouse over the most recent error.

The table displays the list of kernel functions. For each function, it displays **Count**, the number of times that the function was encountered by a snapshot. It also displays **Percent**, the percentage of times that the snapshot encountered the function out of the total number of times it encountered any kernel function. Note that **Percent** ignores any snapshots that encountered a user application, so it is just a percentage of time in the kernel function out of time in the kernel.

Kernel Activity Context Menu

While positioned in a kernel activity panel, right-clicking displays a context menu, which contains the following content:

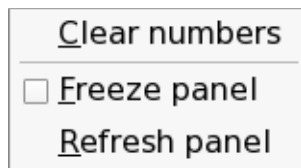


Figure 3-29. Kernel Activity Context Menu

The following paragraphs describe the menu items in detail:

Clear numbers

Mnemonic: C

This menu item manually resets all the counts for every kernel function to zero. Use this to clear away all existing counts and start fresh, for instance right before starting a user application to test.

Freeze/Unfreeze panel

Mnemonic: F

This menu item toggles the **Freeze** setting for the panel. When frozen, data values are not refreshed automatically. This menu item overrides the **Freeze** setting for the window, but only applies to the particular panel.

Refresh panel

Mnemonic: R

This menu item causes all data within the panel to be refreshed once, regardless of the **Freeze** setting.

Required Privileges for the Kernel Activity Panel

The Kernel Activity panel utilizes CPU performance counters.

CPU performance counter support is only available on modern Intel-based systems and requires a recent RedHawk kernel from Concurrent Real-Time. Additionally, by default, you must either be the **root** user or have the `CAP_SYS_ADMIN` capability. However, you can allow wide-open use of this panel for all users via the following command (executed as the **root** user):

```
sysctl -w kernel.perf_event_paranoid=-1
```

You should modify `/etc/sysctl.conf` if you want this setting to persist after a reboot.

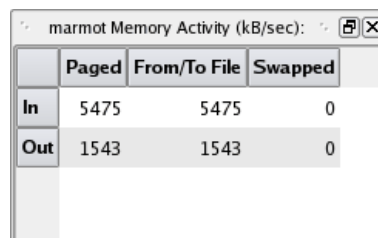
You can read more about the privileges/capabilities required when using CPU performance counters in the `ccur_per_event_open(3)` man page. See “Capabilities” on page 1-3 for more information on capabilities and how to set them up for your user account.

Memory Activity Panel

The Memory Activity panel is a system status panel (see “System Status Panel” on page 3-1). Each pane displays interrupt activity on the system. Each pane displays the number of physical memory page transfers per second that occurred over the period of time defined by the refresh interval.

Memory Activity Text Pane

The following illustrates the Memory Activity Text pane:



The screenshot shows a window titled "marmot Memory Activity (kB/sec):" with a table containing the following data:

	Paged	From/To File	Swapped
In	5475	5475	0
Out	1543	1543	0

Figure 3-30. Memory Activity Text Pane

The information displayed in this area includes:

Paged

The number of KB read in or written out per second due to paging is displayed in this column.

From/To File

The number of KB read in from disk files to memory and read out from memory to disk files is displayed in this column.

Swapped

The number of KB read in from swap space to memory and read out from memory to swap space is displayed in this column.

Memory Activity Bar Graph Pane

The Memory Activity Bar Graph pane provides individual bar graphs detailing the paging rates.

The following illustrates the Memory Activity Bar Graph pane:

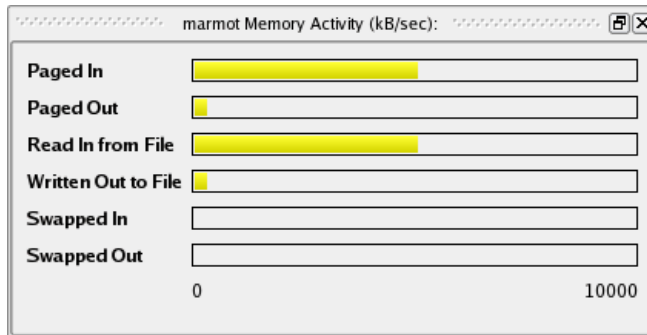


Figure 3-31. Memory Activity Bar Graph Pane

The number of KB in physical memory page transfers per second for each page operation type is displayed as a horizontal line in each graph. See "Memory Activity Text Pane" on page 3-55 for definitions of the metrics.

Memory Activity Line Graph Pane

The Memory Activity Line Graph pane provides individual line graphs detailing the paging rates.

The following illustrates the Memory Activity Line Graph pane:

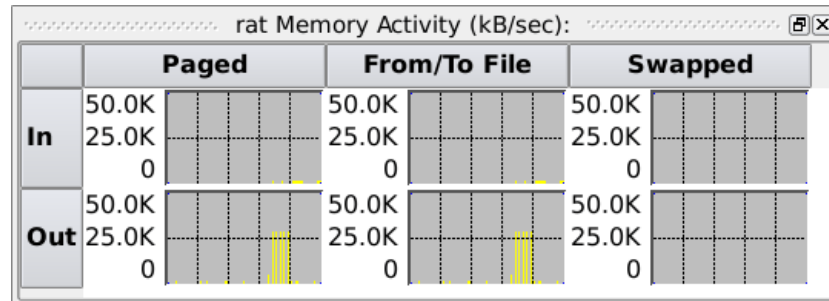


Figure 3-32. Memory Activity Line Graph Pane

The number of KB in physical memory page transfers per second for each operation type is displayed as a vertical line in each graph. See “Memory Activity Text Pane” on page 3-55 for definitions of the metrics.

Kernel Virtual Memory Panel

The Memory Activity panel is a system status panel (see “System Status Panel” on page 3-1). Each pane displays the allocation of kernel virtual memory that exists at the time of the last refresh.

Kernel Virtual Memory Text Pane

The following illustrates the Kernel Virtual Memory Text pane:

Total	Used	Free	Largest Chunk
114680	20300	94380	49132

Figure 3-33. Kernel Virtual Memory Text Pane

The information displayed in this area includes:

Total

The total number of KB reserved for kernel virtual memory (vmalloc) on the system is displayed in this column.

Used

The number of KB of kernel virtual memory (vmalloc) allocated by the kernel or user processes is displayed in this column.

Free

The number of KB of kernel virtual memory (vmalloc) not being used is displayed in this column.

Largest Chunk

The number of KB in the largest contiguous block of free kernel virtual memory (vmalloc) is displayed in this column.

Kernel Virtual Memory Bar Graph Pane

The Kernel Virtual Memory Bar Graph pane provides individual bar graphs detailing the allocation of kernel virtual memory on the system.

The following illustrates the Kernel Virtual Memory Bar Graph pane:

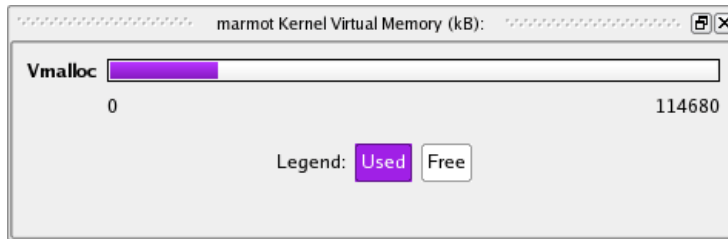


Figure 3-34. Kernel Virtual Memory Bar Graph Pane

The Total kernel virtual memory for the system is represented by the full width of the graph, and is expressed in KB. The amount that currently is Used is displayed in purple, and the amount that is Free is displayed in white. See “Kernel Virtual Memory Text Pane” on page 3-57 for definitions of the metrics.

Kernel Virtual Memory Line Graph Pane

The Kernel Virtual Memory Line Graph pane provides individual line graphs detailing the allocation of kernel virtual memory (vmalloc).

The following illustrates the Kernel Virtual Memory Line Graph pane:

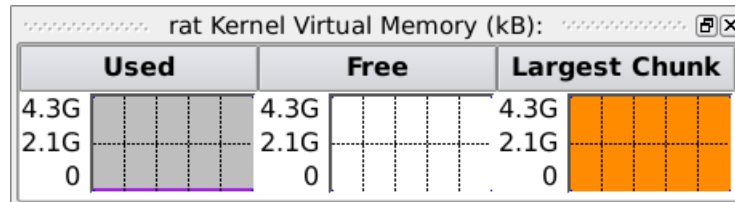


Figure 3-35. Kernel Virtual Memory Line Graph Pane

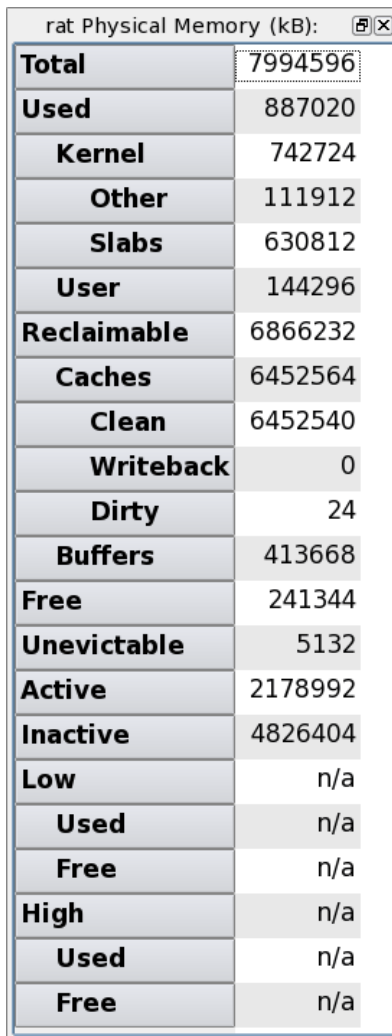
The allocation of kernel virtual memory is displayed as a vertical line in each graph. See “Kernel Virtual Memory Text Pane” on page 3-57 for definitions of the metrics

Physical Memory Panel

The Physical Memory panel is a system status panel (see “System Status Panel” on page 3-1). Each pane displays the allocation of physical memory that exists at the time of the last refresh.

Physical Memory Text Pane

The following illustrates the Physical Memory Text pane:



The screenshot shows a window titled "rat Physical Memory (kB):" with a table of memory usage statistics. The table has two columns: a category name and a numerical value in kB. The categories include Total, Used, Kernel, Other, Slabs, User, Reclaimable, Caches, Clean, Writeback, Dirty, Buffers, Free, Unevictable, Active, Inactive, and Low/High memory usage (Used and Free).

Category	Value (kB)
Total	7994596
Used	887020
Kernel	742724
Other	111912
Slabs	630812
User	144296
Reclaimable	6866232
Caches	6452564
Clean	6452540
Writeback	0
Dirty	24
Buffers	413668
Free	241344
Unevictable	5132
Active	2178992
Inactive	4826404
Low	n/a
Used	n/a
Free	n/a
High	n/a
Used	n/a
Free	n/a

Figure 3-36. Physical Memory Text Pane

The information displayed in this area is as follows. All values are in KB.

The total memory of the system is subdivided in three different, orthogonal, ways:

- Usage
- Recency
- Low vs. High Memory

Subdivision by Usage

Total

The total memory on the system in KB.

Used

The total memory in KB allocated by the kernel or user processes but not including reclaimable memory. This category is further subdivided as follows:

Kernel shows the total amount of memory allocated by the kernel. **Other** indicates memory allocated by the kernel for page tables and other internal data stored in whole pages. **Slabs** indicates memory allocated by the kernel for internal data structures.

User shows the total amount of memory allocated by user processes, but not including reclaimable memory.

Reclaimable

The total memory in KB allocated by the kernel but easily discardable and therefore available to the kernel or user processes if necessary:

Caches indicates total memory used for cached copies of pages from disk files: **Clean** shows memory used for cached pages which remain unmodified, **Write-back** shows memory used for cached pages which have been modified and are in the process of being written back to disk, and **Dirty** shows the memory used for cached pages which have been modified but have not yet been written back to disk.

Buffers indicates memory for pages intended to be written to disk but which have not yet been written. This differs from Dirty in that the pages were not originally cached. For example, calls to **write(2)** may create such pages.

Free

The total memory in KB that is not being used.

Subdivision by Recency

This subdivision includes **Total**, **Kernel** (and its further subdivisions), **Free**, and also:

Unevictable

User, cache or buffer memory that currently cannot be to be reclaimed if memory is needed (e.g. locked memory).

Active

User, cache or buffer memory that has been used recently and therefore is unlikely to be reclaimed when memory is needed.

Inactive

User, cache or buffer memory that has been used less recently and therefore is likely to be reclaimed when memory is needed.

Subdivision by Low vs. High Memory

Low

Total low memory, which is usable for any purpose by either user processes or the kernel. **Used** indicates low memory including reclaimable memory that is allocated. **Free** indicates low memory not allocated.

High

Total high memory, which is usable for user processes or the kernel's page cache but not for most kernel data structures. **Used** indicates high memory that is allocated for user processes, page tables or buffers, including reclaimable memory. **Free** indicates high memory not allocated for any purpose.

Physical Memory Bar Graph Pane

The Physical Memory Bar Graph pane provides individual bar graphs detailing the allocation of physical memory on the system.

The following illustrates the Physical Memory Bar Graph pane:

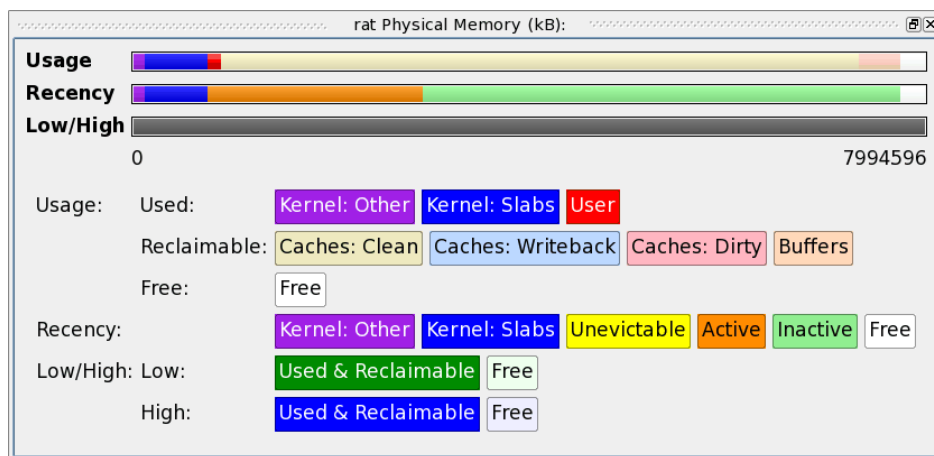


Figure 3-37. Physical Memory Bar Graph Pane

The allocation of physical memory is displayed as horizontal lines. See “Physical Memory Text Pane” on page 3-60 for definitions of the metrics. One bar is displayed for each method of subdivision. In the Usage bar, dark colors are used to represent **Used** memory, while light colors are used to represent **Reclaimable** memory, and white is used for **Free** memory.

Physical Memory Line Graph Pane

The Physical Memory Line Graph pane provides individual line graphs detailing the allocation of physical memory.

The following illustrates the Physical Memory Line Graph pane:

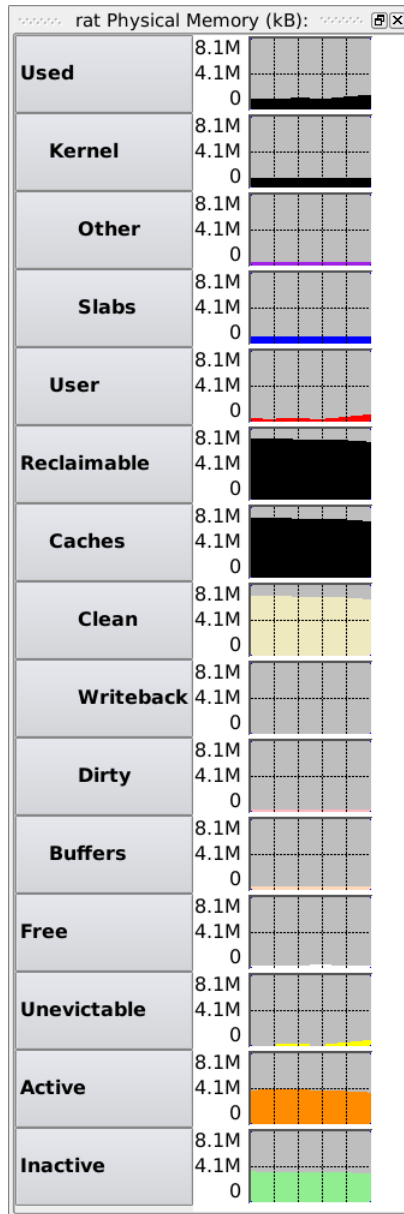


Figure 3-38. Physical Memory Line Graph Pane

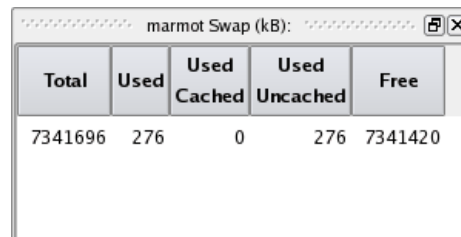
The allocation of physical memory is displayed as a vertical line in each graph. See “Physical Memory Text Pane” on page 3-60 for definitions of the metrics.

Swap Panel

The Swap panel is a system status panel (see “System Status Panel” on page 3-1). Each pane displays the swap space allocation at the time of the last refresh.

Swap Text Pane

The following illustrates the Swap Memory Text pane:



Total	Used	Used Cached	Used Uncached	Free
7341696	276	0	276	7341420

Figure 3-39. Swap Memory Text Pane

The information displayed in this area includes:

Total

The total swap space on the system in KB.

Used

The total swap space in use on the system. This is further subdivided as Cached and Uncached.

Used: Cached

The total swap space that, after having been swapped out, has been swapped back in and therefore is in both physical memory and swap space. If additional swapping becomes necessary in the future, the memory characterized in this way can be swapped for nearly no cost, because the swap area already contains a copy of them memory. Conversely, if swap space becomes low, this swap space can be reclaimed because its content already has been swapped back into physical memory.

Used: Uncached

The total swap space that has not been swapped back in and therefore exists only in swap space.

Free

The total swap space that is not being used.

Swap Bar Graph Pane

The Swap Memory Bar Graph pane provides a bar graph illustrating the used and free swap space on the system. The total width of the bar represents the total amount of swap space on the system. The metrics are described in “Swap Text Pane” on page 3-65.

The following illustrates the Swap Memory Bar Graph pane:

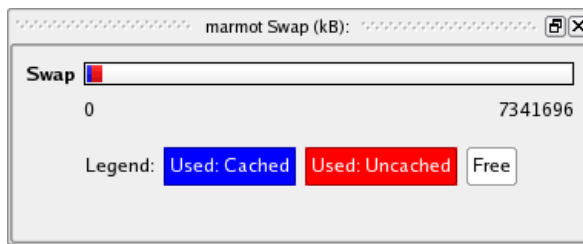


Figure 3-40. Swap Memory Bar Graph Pane

Swap Line Graph Pane

The Swap Memory Line Graph pane provides individual line graphs detailing the swap space.

The following illustrates the Swap Memory Line Graph pane:

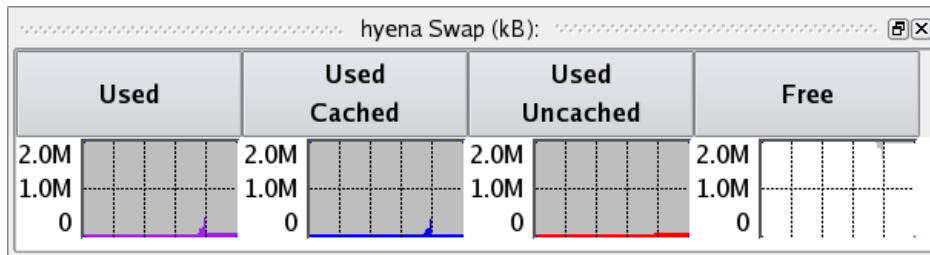


Figure 3-41. Swap Memory Line Graph Pane

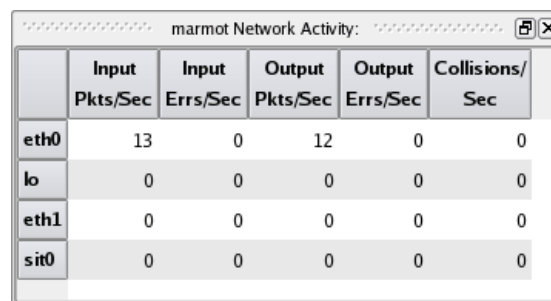
The allocation of swap memory is displayed as a vertical line in each graph. The metrics are described in “Swap Text Pane” on page 3-65.

Network Activity Panel

The Network Activity panel is a system status panel (see “System Status Panel” on page 3-1). Each pane displays statistics related to network operations that occurred over the period of time defined by the refresh interval. Statistics for individual network devices are displayed.

Network Activity Text Pane

The following illustrates the Network Activity Text pane:



	Input Pkts/Sec	Input Errs/Sec	Output Pkts/Sec	Output Errs/Sec	Collisions/ Sec
eth0	13	0	12	0	0
lo	0	0	0	0	0
eth1	0	0	0	0	0
sit0	0	0	0	0	0

Figure 3-42. Network Activity Text Pane

The information displayed in this area includes:

Interface

The name of the network interface is displayed in the left-most column.

Input Pkts/Sec

The number of packets per second transferred on behalf of input operations is displayed in this column.

Input Errs/Sec

The number of errors per second that occurred during input operations is displayed in this column.

Output Pkts/Sec

The number of packets per second transferred on behalf of output operations is displayed in this column.

Output Errs/Sec

The number of errors per second that occurred during output operations is displayed in this column.

Collisions/Sec

The number of network collisions per second that occurred is displayed in this column.

Network Activity Bar Graph Pane

The Network Activity Bar Graph pane provides individual bar graphs detailing metrics related to network operations.

The following illustrates the Network Activity Bar Graph pane:

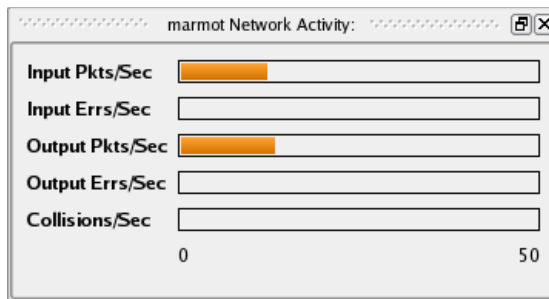


Figure 3-43. Network Activity Bar Graph Pane

Individual bar graphs are shown for each metric. The value of the metric is displayed horizontally in each graph. See "Network Activity Text Pane" on page 3-67 for definitions of these metrics.

Network Activity Line Graphs

The Network Activity Line Graph pane provides individual line graphs detailing metrics related to network operations.

The following illustrates the Network Activity Line Graph pane:

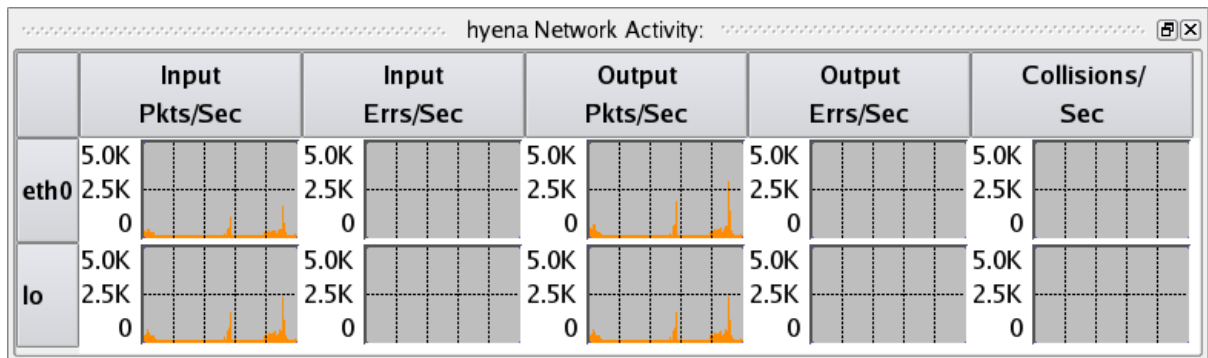


Figure 3-44. Network Activity Line Graph Pane

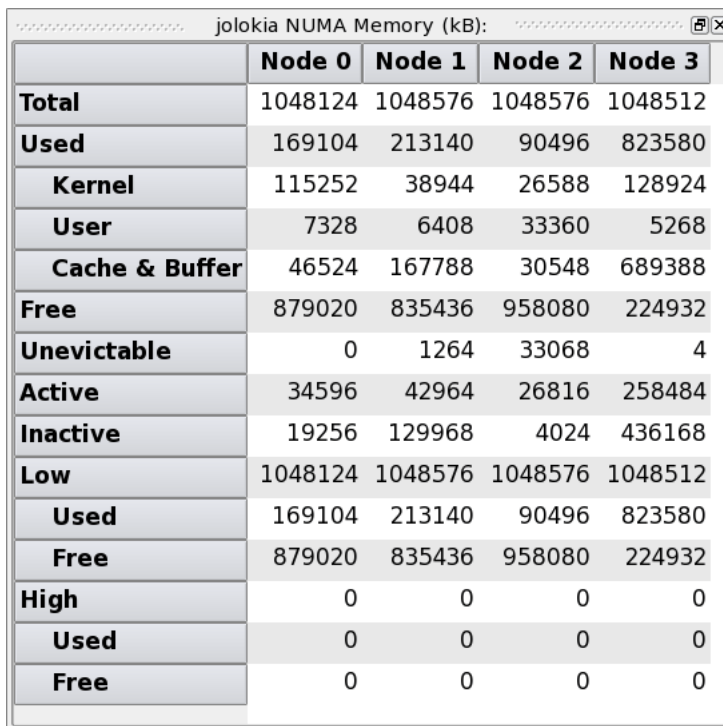
Individual line graphs are shown for each network device for each metric. The value of the metric is displayed vertically in each graph. See “Network Activity Text Pane” on page 3-67 for definitions of these metrics.

NUMA Panel

The NUMA panel is a system status panel (see “System Status Panel” on page 3-1). Each pane displays the allocation of memory on each NUMA node at the time of the last refresh.

NUMA Text Pane

The following illustrates the NUMA Text pane:



	Node 0	Node 1	Node 2	Node 3
Total	1048124	1048576	1048576	1048512
Used	169104	213140	90496	823580
Kernel	115252	38944	26588	128924
User	7328	6408	33360	5268
Cache & Buffer	46524	167788	30548	689388
Free	879020	835436	958080	224932
Unevictable	0	1264	33068	4
Active	34596	42964	26816	258484
Inactive	19256	129968	4024	436168
Low	1048124	1048576	1048576	1048512
Used	169104	213140	90496	823580
Free	879020	835436	958080	224932
High	0	0	0	0
Used	0	0	0	0
Free	0	0	0	0

Figure 3-45. NUMA Text Pane

The total memory for each NUMA node is subdivided in two different, orthogonal, ways:

- Usage
- Recency
- Low vs. High Memory

Subdivision by Usage

Total

The total memory in the NUMA node.

Used

The total memory in KB from the NUMA node allocated by the kernel or user processes, including reclaimable memory. This is further subdivided into **Kernel**, **User** and **Cache & Buffer** memory.

Kernel

The total memory in KB from the NUMA node allocated by the kernel.

User

The total memory in KB from the NUMA node allocated as user memory.

Cache & Buffer

The total memory in KB allocated by the kernel but easily discardable and therefore available to the kernel or user processes if necessary. Cache memory is for cached copies of pages from disk files. Buffer memory is for pages intended to be written to disk but which have not yet been written. For example, calls to **write(2)** may create buffer memory.

Free

Total unused memory in KB available on the NUMA node.

Subdivision by Recency

This subdivision includes Total, Kernel, Free and the following:

Unevictable

User, cache or buffer memory that currently cannot be reclaimed if memory is needed (e.g. locked memory).

Active

User, cache or buffer memory that has been used recently and therefore is unlikely to be reclaimed when memory is needed.

Inactive

User, cache or buffer memory that has been used less recently and therefore is likely to be reclaimed when memory is needed.

Subdivision by Low vs. High Memory

Total

The total memory in the NUMA node.

Low

Total low memory, which is usable for any purpose by either user processes or the kernel. **Used** indicates low memory allocated, including reclaimable memory. **Free** indicates low memory that is not allocated for any purpose.

High

Total high memory, usable for user processes or the kernel's page cache but not for most kernel data structures. **Used** indicates high memory allocated, including reclaimable memory. **Free** indicates high memory that is not allocated for any purpose.

NUMA Bar Graph Pane

The NUMA Bar Graph pane provides individual bar graphs detailing metrics related to memory allocation on NUMA systems.

The following illustrates the NUMA Bar Graph pane:

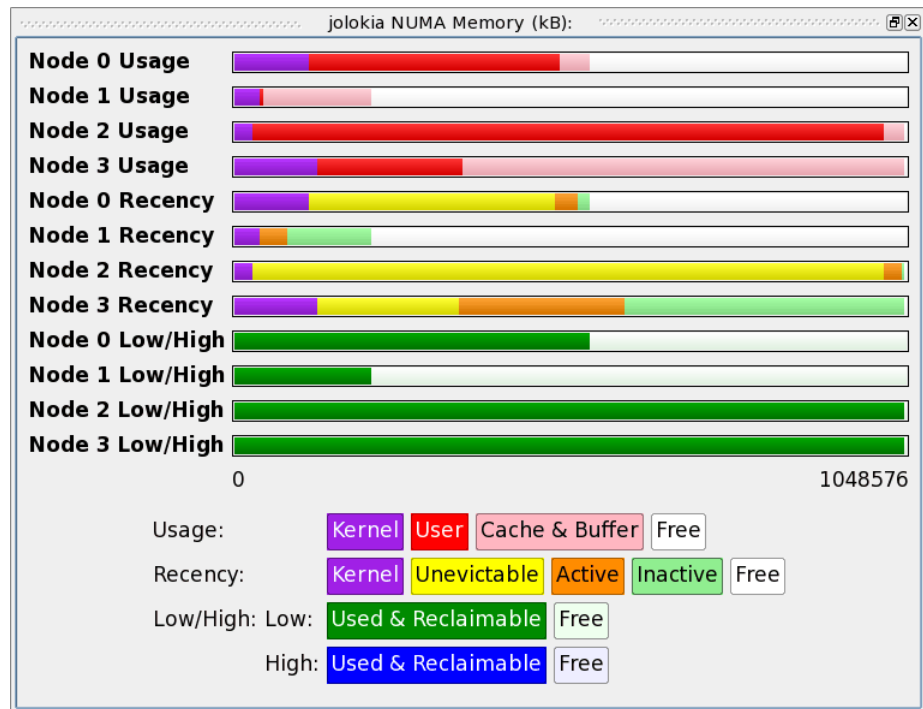


Figure 3-46. NUMA Bar Graph Pane

The allocation of memory for each NUMA node is displayed as horizontal lines. One bar is displayed for each method of subdivision for each NUMA node. See “NUMA Text Pane” on page 3-70 for definitions of the subdivisions and metrics.

NUMA Line Graph Pane

The NUMA Line Graph pane provides individual line graphs detailing metrics related to memory allocation on NUMA systems.

The following illustrates the NUMA Line Graph pane:

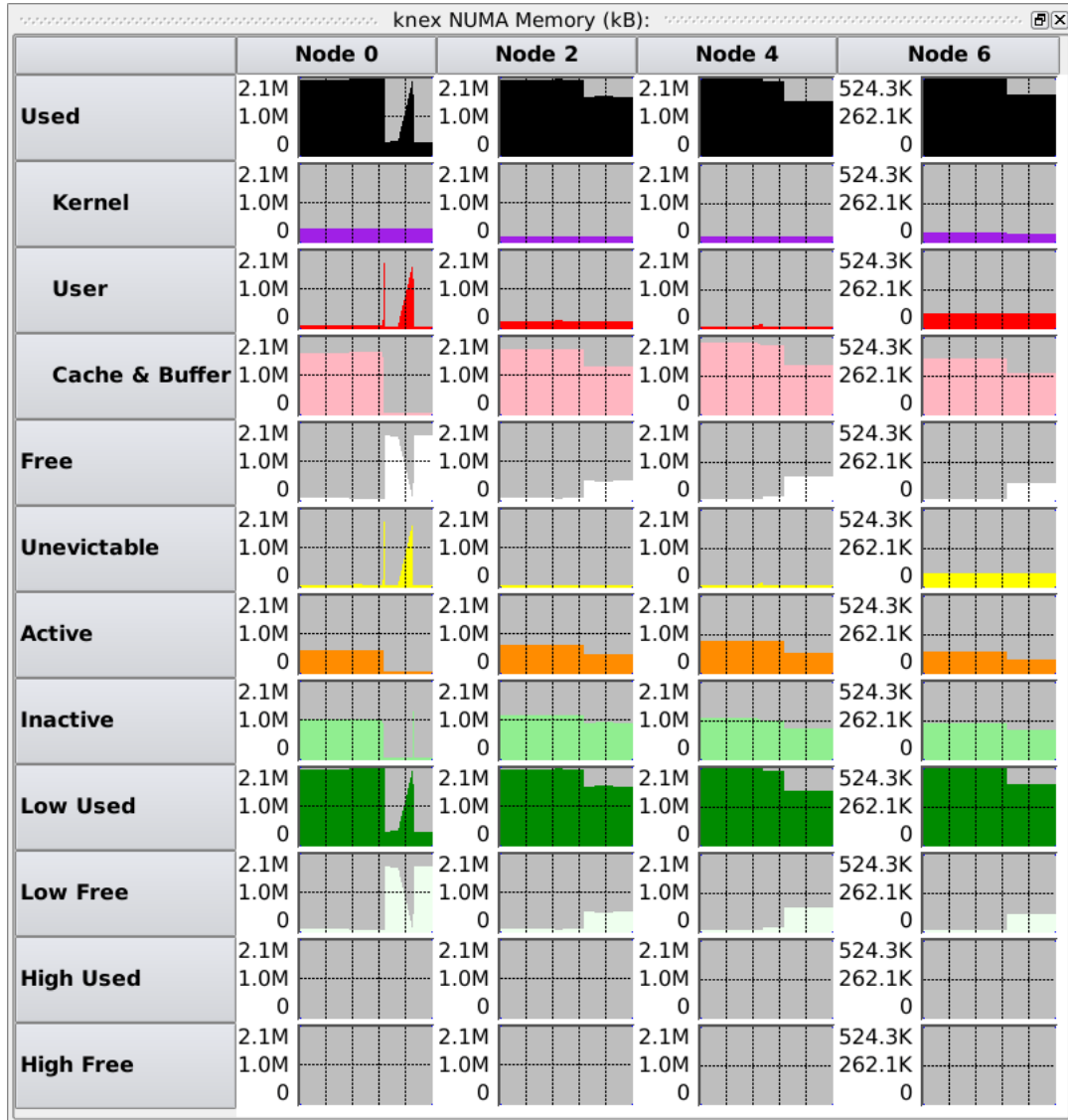


Figure 3-47. NUMA Line Graph Pane

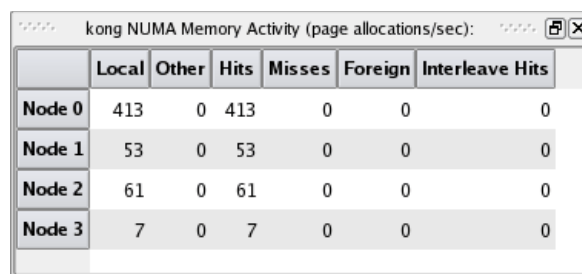
Individual line graphs are shown for each node's memory allocation and for each metric. See "NUMA Text Pane" on page 3-70 for definitions of these metrics.

NUMA Activity Panel

The NUMA Activity panel is a system status panel (see “System Status Panel” on page 3-1). Each pane displays statistics related to memory activity on NUMA systems that occurred over the period of time defined by the refresh interval. Statistics for individual nodes are displayed.

NUMA Activity Text Pane

The following illustrates the NUMA Activity Text pane:



	Local	Other	Hits	Misses	Foreign	Interleave Hits
Node 0	413	0	413	0	0	0
Node 1	53	0	53	0	0	0
Node 2	61	0	61	0	0	0
Node 3	7	0	7	0	0	0

Figure 3-48. NUMA Activity Text Pane

The information displayed in this area includes:

Local

Pages allocated from this node by a CPU associated with this node.

Other

Pages allocated from this node by a CPU not associated with this node.

Hits

Pages allocated from this node when this was the preferred node in the allocation’s memory policy.

Misses

Pages allocated from this node when this was not the preferred node in the allocation’s memory policy.

Foreign

Pages allocated from another node when this was the preferred node in the allocation’s memory policy.

Interleaved Hits

Pages allocated from this node with interleaved memory policy when this was the appropriate interleaved node.

NUMA Activity Bar Graph Pane

The NUMA Activity Bar Graph pane provides individual bar graphs detailing metrics related to memory activity on NUMA systems.

The following illustrates the NUMA Activity Bar Graph pane:

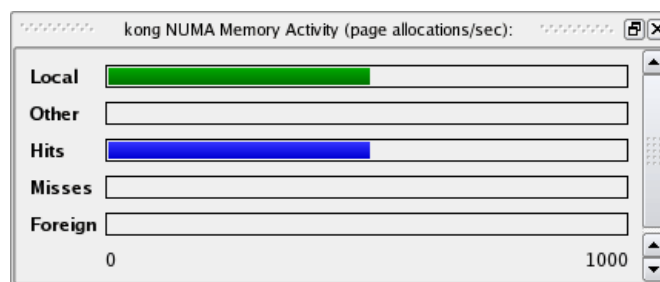


Figure 3-49. NUMA Activity Bar Graph Pane

Individual bar graphs are shown for each node's memory activity. See "NUMA Activity Text Pane" on page 3-75 for definitions of these metrics.

NUMA Activity Line Graph Pane

The NUMA Activity Line Graph pane provides individual line graphs detailing metrics related to memory activity on NUMA systems.

The following illustrates the NUMA Activity Line Graph pane:

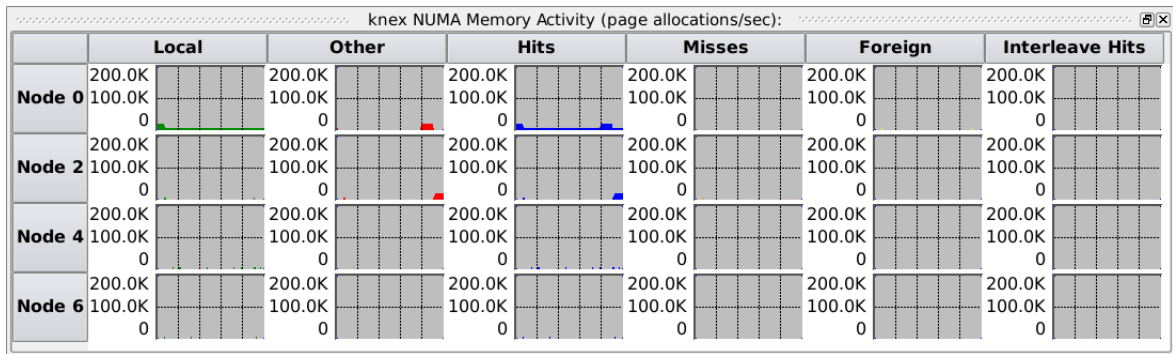


Figure 3-50. NUMA Activity Line Graph Pane

Individual line graphs are shown for each node’s memory activity. See “NUMA Text Pane” on page 3-70 for definitions of these metrics.

NUMA Configuration Panel

The NUMA Configuration panel provides two panes that display configuration statistics for each node on NUMA systems: the CPUs pane and the Distance pane.

Because this panel provides statistics related to the NUMA configuration of the system, the statistics do not change with refresh intervals. Changes appear only when the system is reconfigured.

NUMA Configuration CPUs Pane

This pane shows the CPUs connected to each NUMA node.

The following illustrates the NUMA Configuration CPUs pane:

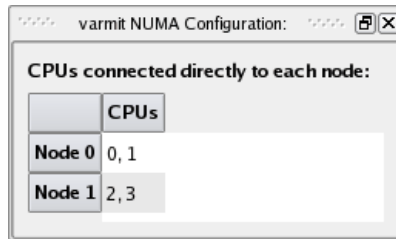


Figure 3-51. NUMA Configuration Text Pane

NUMA Configuration Distance Pane

The NUMA Configuration Distance pane provides the relative distances from CPUs to memory on NUMA systems.

The following illustrates the NUMA Configuration Distance pane:

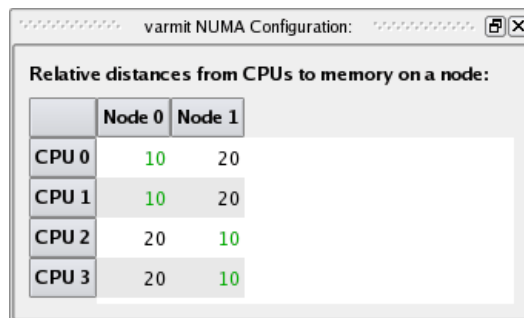


Figure 3-52. NUMA Configuration Distance Pane

Values displayed in green show minimum distances for each CPU. Generally, that means that the a CPU is connected directly to the node if the cell in the table for that pair is green.

NUMA Configuration Context Menu

While positioned in the NUMA Configuration panel, right-clicking displays the NUMA Configuration context menu.

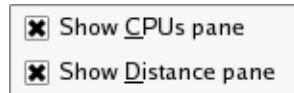


Figure 3-53. NUMA Configuration Context Menu

The following paragraphs describe the menu items in detail:

Show CPUs pane

Mnemonic: C

This menu item toggles the visibility of the NUMA Configuration CPUs pane within the panel.

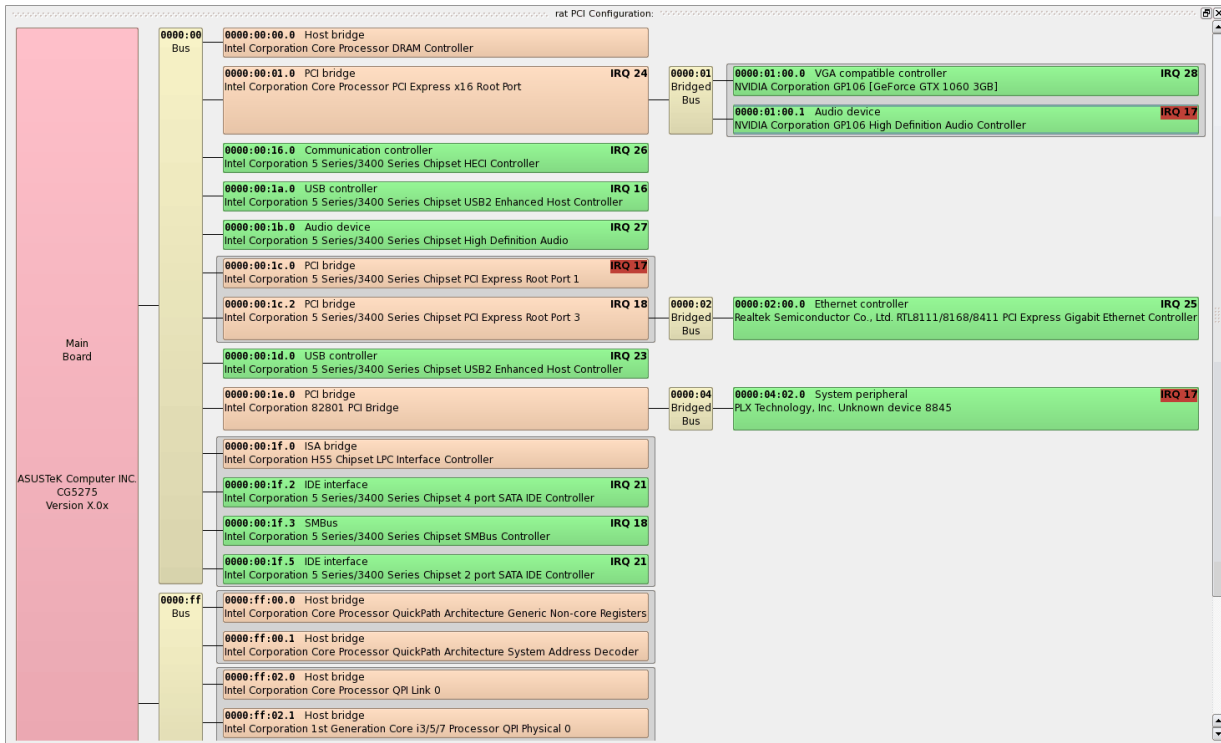
Show Distance pane

Mnemonic: D

This menu item toggles the visibility of the NUMA Configuration Distance pane within the panel.

PCI Configuration Panel

The PCI Configuration panel provides a graphical representation of the organization of PCI buses and devices.



Within this panel, the *mainboard*, *buses*, and *devices* are displayed as colored boxes. Connections between them are displayed with horizontal lines. Connections between devices (and the mainboard) always occur via buses. The mainboard usually has at least one bus, and may have more. In addition, some devices called *bridges* may provide additional buses attached to the mainboard indirectly. The colors of the items are as follows:

- Pink: Mainboard (always leftmost)
- Yellow: Bus
- Orange: Bridge device capable of providing an additional bus
- Green: Normal device

In addition, PCI devices can provide multiple *functions*. Each function is presented as an independent device. But it is useful to see the relationship graphically. So, multiple functions within a single device are shown grouped within a gray rectangle.

Each item can be in one of three states, determining how much information is displayed for that device:

- Minimal
- Medium
- Full

The amount of information display for each item is determined by its state. Clicking in a device causes its state to cycle to the next state in the list. If the state is Full, then it reverts to the Minimal state.

For all states, the following information is displayed:

Upper left: This information depends on whether the item is a device or a bus. For a device, it is the extended geographical ID of the device (a.k.a. its BDF). It is of the form *dddd:BB:DD.f*, where:

dddd is the Domain number in hexadecimal.

BB is the Bus number in hexadecimal.

DD is the Device number in hexadecimal.

f is the Function number in decimal.

For a bus, the form is similar, but it contains only the geographical ID of the form *dddd:BB*.

Upper center: The PCI class of the device.

Upper right: Any IRQs used by the device. If the mouse is hovered over a particular device which uses an IRQ, it will cause the IRQ numbers for any *other* devices which share the IRQ to be displayed highlighted with a red background.

For items whose state is at least Medium, the following additional information is displayed:

Second line: A description of the device, or simply **Bus** or **Bridge Bus** for buses. This description is for the *chip* manufacturer and model.

An item must be in the Full state to see the remaining information:

Subsystem: A description of the *card* manufacturer and model.

Vendor: The vendor ID used to determine the device (chip) description.

Device: The device ID used to determine the device (chip) description.

Subsystem Vendor: The vendor ID used to determine the subsystem (card) description.

Subsystem Device: The device ID used to determine the subsystem (card) description.

NOTE

The global registry of vendor IDs is maintained by PCI-SIG. Each vendor maintains its own list of device IDs.

IRQ: An expanded description of the IRQ assigned to the device, if any. It reiterates the IRQ number, but also follows it with a list of device chips which use the same IRQ. The names are assigned by the kernel drivers which control them.

Kernel Driver: The name of the kernel driver which controls this device.

I/O ports: Device I/O ports mapped into host memory.

Memory: Device memory mapped into host memory.

Register offsets: Device registers or register ranges mapped into host memory.

IRQs: IRQs assigned to the device.

DMA: Device DMA memory mapped into host memory.

Buses: Secondary bus numbers bridged by the device.

Expansion ROM: Device expansion ROM mapped into host memory.

Addresses shown are of the memory mapping on the host. Mappings may have any or none of these attributes:

- **64-bit:** whether or not the region supports 64-bit addressing.
- **readonly:** whether or not the memory is write-protected.
- **prefetchable:** memory may be prefetched (disabled if reads or writes would have side effects).
- **cacheable:** memory reads may be cached (disabled if reads or writes would have side effects).

PCI Configuration Context Menu

While positioned in the PCI Configuration panel, right-clicking displays the PCI Configuration context menu.



The following paragraphs describe the menu items in detail:

All Devices: Minimal

This menu item changes the state of all items in the panel to **Minimal**.

All Devices: Medium

This menu item changes the state of all items in the panel to **Medium**.

All Devices: Full

This menu item changes the state of all items in the panel to Full.

Find...

This menu item activates a

Accelerator: Ctrl+F

This menu item displays the find bar (see “Find Bar” on page 3-86) at the bottom of the panel. Enter a regular expression in the text entry area to initiate a search.

Find Again

Accelerator: Ctrl+G

This menu item searches for the next match of the regular expression previously entered in the find bar (see “Find Bar” on page 3-86).

Refresh Panel

Mnemonic: R

This menu item causes the panel to refresh in case the PCI configuration has changed.

Process List Panel

The purpose of the **Process List** panel is to provide detailed descriptions of individual processes and threads.

The process list can be organized with a parent-child hierarchy or can be a flat list of processes. In addition, either the hierarchical or flat structures can be organized on a per-user-ID basis, if desired. These selections can be made from the **Process List** context menu (see “Process List Context Menu” on page 3-87).

The following illustrates the Process List panel:

PID	State	Size	Data	%CPU	CPU Time	User	System	CPU	Affinity	Nice	RPri	CL	Command
Users													
├── ldap													
├── lp													
├── mail													
├── messageb													
├── nobody													
├── ntp													
├── root													
├── sms													
└── todd													
4397	Waiting	20020	10516	0.0	14.96	14.44	0.52	0	all	0	0	OT	emacs
6326	Waiting	4716	668	0.0	0.24	0.20	0.04	0	all	0	0	OT	ksh
4279	Waiting	1368	108	0.0	0.18	0.13	0.05	0	all	0	0	OT	prefix
4434	Waiting	1728	248	0.0	0.00	0.00	0.00	0	all	0	0	OT	be-polite
4278	Waiting	1512	252	0.0	0.19	0.15	0.04	0	all	0	0	OT	watch
4329	Waiting	3884	556	0.0	0.05	0.02	0.03	0	all	0	0	OT	xpmxload
4330	Waiting	6180	1372	0.0	50.34	30.92	19.42	0	all	0	0	OT	xpmxterm
4707	Waiting	9364	4556	0.0	0.11	0.09	0.02	0	all	0	0	OT	xpmxterm
4720	Waiting	4880	832	0.0	0.10	0.08	0.02	0	all	0	0	OT	ksh
4726	Waiting	9360	4552	0.0	0.09	0.06	0.03	0	all	0	0	OT	xpmxterm
4734	Waiting	13648	8840	0.0	0.15	0.08	0.07	0	all	0	0	OT	xpmxterm
4773	Waiting	12156	7348	0.0	0.10	0.07	0.03	0	all	0	0	OT	xpmxterm

Figure 3-54. Process List Panel

The columns of fields that are displayed are controlled from the **Process Fields** menu from the **Display Fields** item of the **Process List** context menu (see “Process Fields Menu” on page 3-117). The command name of the process is displayed in short form by default, although this also can be changed from the context menu. The long form of the command displays when the mouse hovers over the short form of the command name.

Information in the process line is automatically updated at a selectable interval. The interval can be changed using the **Preferences** dialog from the **Preferences...** item from the **File** menu. Immediate refresh of information is available using the **Refresh** tool icon or through the **Process List** context menu.

Some operations in the panel operate on processes that are selected in the process list. Selection of individual processes is done using the left mouse button, which toggles whether the associated process is selected or deselected. Multiple selection is done by selecting individual processes with the shift and/or control keys. Selecting or deselecting a multi-threaded process selects or deselects all its threads.

You can use the **Process List** context menu (see “Process List Context Menu” on page 3-87) to filter the processes shown. When the panel is filtered, a status bar appears at the bottom of the panel to describe the filter, shown in Figure 3-55.



Figure 3-55. Process List Filter Bar

The filter status bar also contains a **Change Filter** button to bring up the **Filter Processes** dialog (see “Filter Processes Dialog” on page 3-91) and a **Remove Filter** button to remove the filter.

You can search for a process with the **Find...** item in the **Process List** context menu (see “Process List Context Menu” on page 3-87) or by using **Ctrl+F**. The find bar appears at the bottom of the panel. See “Find Bar” on page 3-86. If a matching line is found, the line is highlighted.

Note that hidden fields are searched. For example, if you search for your username, processes you own will match even though the **Username** field is not displayed.

You can search for another process with the **Find Again** item in the **Process List** context menu or by using **Ctrl+G**.

For threads, NightTune attempts to determine individual thread names using three techniques:

1. If the thread uses the **prctl(2)** service and names itself using the **PR_SET_NAME** form of that service, NightTune will display that name.
2. If the process is being debugged by NightView, NightTune consults NightView and retrieves a list of thread names. NightView automatically names threads by using the simple name of their start routine, as passed to **pthread_create(3)**. You can also override a thread's name in NightView using the **set-thread-name** command.
3. If the process uses the NightTrace Logging API and names its threads using **trace_set_thread_name(3)**, then NightTune can determine individual thread names. This is true even if the process isn't actively logging trace data.

If NightTune cannot determine a thread's name, the field is left blank in the process list. NightTune will not be able to determine thread names using the latter two techniques above if the user running NightTune lacks appropriate access to the processes in question.

Process List Drag and Drop Operations

Individual processes or groups of processes can be dragged onto various destination panels and drop targets.

To drag an individual process, click anywhere on the row describing the process and drag the pointer to the destination area and release the mouse button. Similarly, you can drag all the processes currently associated with a user by clicking on the user line and dragging the pointer to the destination area.

The Process List panel supports the following drag and drop operations:

- Dragging a process or user line onto a CPU line in the CPU Shielding and Binding panel binds the processes to the corresponding CPU.
- Dragging a process onto the Process Scheduler dialog changes the dialog to refer to that process.
- Dragging a process or user line onto the Kill tool icon causes the processes to be killed with a **SIGKILL** signal.
- Dragging a process or user line onto the Unbind tool icon causes the processes to be unbound from any CPUs.

Find Bar

A find bar, shown in Figure 3-56, appears at the bottom of the Process List panel when you ask to search for a process. A find bar also appears at the bottom of the Trace Output window when you ask to search the output of those traces.



Figure 3-56. Process List Find Bar

Enter a regular expression in the text entry field; for example, a program name. As you type more characters, the search progresses. If no match is found, the background color of the text entry field changes to a salmon color. A label at the end of the find bar shows the status of the search.

To go to the next match, click on **Next** or type **Ctrl+G**. Click on **Previous** to go to the previous match. The **Match case** check box indicates whether the search should be case-sensitive.

Press **Esc** or click on the close button **X** to hide the find bar.

The find bar in the Trace Output window has an additional control, as shown in Figure 3-57.

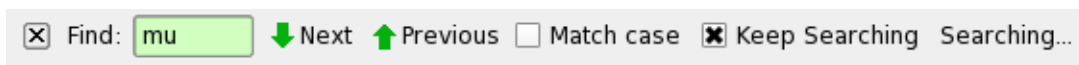


Figure 3-57. Find Bar with Keep Searching Option

If the **Keep Searching** box is checked, and the search reaches the end of the output without matching, the search waits for more trace output until a match is found. During this time the background of the text entry field changes to a light green and the label at the end of the find bar shows the status **Searching....** To stop the search from waiting for more output, uncheck the **Keep Searching** box.

When the **Keep Searching** box is not checked, a forward search wraps around to the beginning of the output. A backward search always wraps around to the end of the output.

Process List Context Menu

To display the Process List context menu, right-click while positioned in the Process List panel.

The following illustrates the Process List context menu:

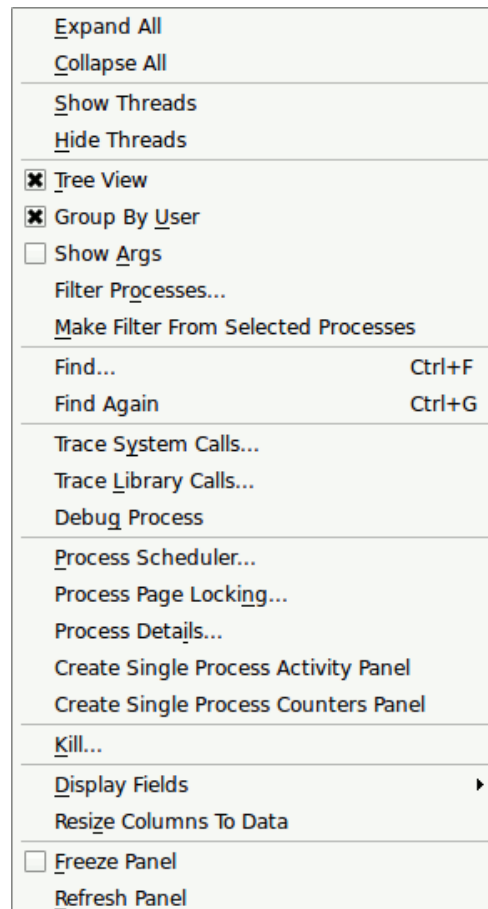


Figure 3-58. Process List Context Menu

The following paragraphs describe the options on the menu in more detail:

Expand All

Mnemonic: E

This menu item expands the selected item and all its interior items.

Collapse All

Mnemonic: C

This menu item collapses the selected item and all its interior items.

Show Threads

Mnemonic: S

This menu item causes the list of threads to be expanded for the selected processes.

Hide Threads

Mnemonic: H

This menu item causes the list of threads for the selected processes to be hidden, leaving a single process line that summarizes all the threads.

Tree View

Mnemonic: T

This menu item, when selected, organizes the processes into a hierarchy according to their parent-child relationships. When not selected, the processes are organized as a flat list.

Group by User

Mnemonic: U

This menu item, when selected, organizes processes by user, with top-level items for each user with running processes and all processes placed beneath their user, with the organization under each user controlled by the **Tree View** setting. When not selected, processes are organized merely according to the **Tree View** setting.

Show Args

Mnemonic: A

This menu item, when selected, causes each process to be described in a long form including arguments in the Command column. If not selected, the short form is displayed in the Command column. In either case, the long form is displayed in a tooltip if the mouse hovers over the Command column.

Filter Processes...

Mnemonic: O

This menu item displays the **Filter Processes** dialog, which allows you to select the processes shown in the **Process List** panel. See “Filter Processes Dialog” on page 3-91.

Make Filter from Selected Processes

Mnemonic: M

This menu item makes a filter that causes the panel to show only the processes you currently have selected. If you selected user lines, the display includes only the processes owned by those users. If you selected process lines, the display includes only those processes. If you selected both user lines and process lines, the display includes the processes owned by those users plus the selected processes.

Find...

Accelerator: Ctrl+F

This menu item displays the find bar (see “Find Bar” on page 3-86) at the bottom of the panel. Enter a regular expression in the text entry area to initiate a search.

Find Again

Accelerator: Ctrl+G

This menu item searches for the next match of the regular expression previously entered in the find bar (see “Find Bar” on page 3-86).

Trace System Calls...

Mnemonic: Y

This menu item displays **strace (1)** output for the selected process. See “Trace Output Window” on page 3-96.

Trace Library Calls...

Mnemonic: L

This menu item displays **ltrace (1)** output for the selected process. See “Trace Output Window” on page 3-96.

Debug Process

Mnemonic: G

This menu item launches the NightView debugger to debug the selected process.

Process Scheduler...

Mnemonic: P

This menu item displays the **Process Scheduler** dialog where the scheduling properties of a selected process can be viewed or modified. See “Process Scheduler Dialog” on page 3-97 for details.

Process Page Locking...

Mnemonic: N

This menu item displays the **Process Page Locking** dialog which allows you to lock or unlock a process's pages into memory. See “Process Page Locking Dialog” on page 3-101 for details.

Process Details...

Mnemonic: I

This menu item displays the **Process Details** dialog which shows memory usage, detailed information about memory pages, file descriptors, signal status, capabilities and environment variables for the selected process. See “Process Details Window” on page 3-101 for details.

Kill...

Mnemonic: K

This menu item displays the **Kill Process** dialog where you may select a signal (e.g. **SIGKILL**) to be sent to the processes.

Display Fields

Mnemonic: D

This menu item displays the **Process Fields** menu that allows you to select the fields (columns) that are visible in the **Process List** panel. See “Process Fields Menu” on page 3-117 for definitions of these metrics.

Resize Columns to Data

Mnemonic: Z

This menu item resizes the columns to fit the data visible.

Freeze Panel

Mnemonic: F

This menu item toggles the **Freeze** setting for this panel. When frozen, this panel is not refreshed automatically.

Refresh Panel

Mnemonic: R

This menu item causes the panel to refresh with the latest list of users and processes, regardless of the **Freeze** setting.

Filter Processes Dialog

This dialog pops up when you click on **Filter Processes...** in the **Process List** context menu, or when you click on the **Change Filter** button in the **Process List** panel. (The button is visible only when processes are filtered.)

The following illustrates the **Filter Processes** dialog.

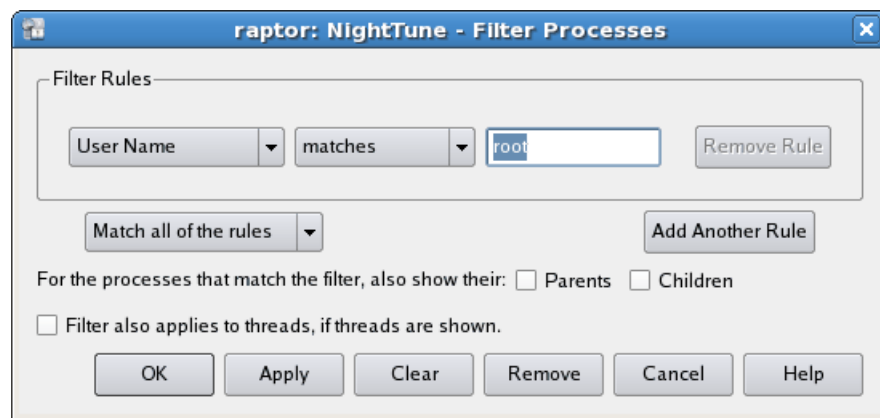


Figure 3-59. Filter Processes Dialog

The **Filter Processes Dialog** lets you describe which processes you want to show in the **Process List** panel. You do this by creating a set of rules for the processes to match and indicating whether each process must match all the rules or may match any of the rules.

Filter Rules

Each filter rule is depicted graphically by a row of controls. The first control is a combo box containing a list of process attributes. Select an attribute on which to filter. The next control is a combo box that describes how the process' value is to be compared to the rule's value. Next is one or more controls to enter the filter value.

For example, you might select “%CPU Time” and “>”, then enter “1.00” to show only processes using more than 1% of the CPU time.

Click the last button in a row, **Remove Rule**, to remove that row of controls and the corresponding filter rule.

For single-threaded processes, all the attributes match based on the process.

For multi-threaded processes, some attributes match based on the process, some match based on the threads, and the rest match if either the process matches or its threads match. The following attributes match based on the process:

- User Name
- Command
- Command Line
- Process ID
- Parent PID
- User ID
- Number of Threads
- Size
- Data Size
- Resident Size
- CUDA Memory Size

The following attributes match based on the threads:

- Thread ID
- State
- Recent CPU
- Affinity
- Nice
- Priority
- Real-Time Priority
- Scheduling Class
- Thread Name
- Thread Number

The following attributes match if either the process matches or its threads match:

- % CPU Time
- CPU Time
- User Time
- System Time

The attributes are described below.

User Name

Enter a regular expression to describe the user names. For example, enter your user name.

Command

Enter a regular expression to describe the program names, not including the program path and the arguments. For example, enter “sh\$”.

Command Line

Enter a regular expression to match the commands, including the program path and the arguments.

Process ID Parent PID

Enter a regular expression to describe the process IDs. For example, enter a single process ID.

User ID

Enter a regular expression to describe the user IDs. For example, enter your user ID.

Number of Threads

Enter a number of threads.

Size Data Size Resident Size CUDA Memory Size

Enter a memory size and select the desired units.

Thread ID

Enter a regular expression to describe the thread IDs. For example, enter a single thread ID.

State

Check the boxes to select the states you want to match.

Recent CPU Affinity

Select whether you want to enter a hexadecimal mask such as “0x17”, or a CPU list such as “0-2,4”, then enter the value.

**Nice
Priority
Real-Time Priority**

Enter a number.

Scheduling Class

Check the boxes to select the scheduling classes you want to match.

Thread Name

Enter the thread name to match. NightTune can determine the name of a thread if the thread uses the `prctl(PR_SET_NAME, ...)` call (see `prctl(2)`), is being debugged by NightView, or is using the NightTrace Logging API and calls `trace_set_thread_name(3x)`.

Thread Number

Enter the ordinal number (the order in which the thread is created, with the main thread being thread number 1) of the thread of interest. This is less useful than Thread Name, as the order in which threads are created may vary, depending on the application; however the Thread Name is often not available. See "Thread Name" on page 3-94.

% CPU Time

Enter the percent CPU time you want to match.

**CPU Time
User Time
System Time**

Enter the CPU time you want to match.

**Match all of the rules
Match any of the rules**

If there are multiple filter rules, this controls whether a process must match all the rules to be accepted by the filter, or whether matching any rule is sufficient.

Add Another Rule

Clicking this button adds another row of controls to represent another filter rule.

**For the processes that match the filter, also show their:
Parents
Children**

In addition to displaying the processes that match the filter, you can show the parents and children of those processes.

Filter also applies to threads, if threads are shown

In addition to processes, the **Process List** panel can show the threads of a multi-threaded process. See **Show Threads** in the **Process List Context Menu**. Check this box to indicate that the filter should control not only which processes are displayed, but also which threads are displayed. Uncheck this box to show all threads.

Note that threads can always contribute to the decision of whether a *process* is displayed, regardless of the state of this check box.

The following buttons control the dialog.

OK

Clicking **OK** applies the filter to the **Process List** panel and closes the dialog.

Apply

Clicking **Apply** applies the filter to the **Process List** panel and leaves the dialog open.

Clear

Clicking **Clear** sets the filter controls to a default state, as they appear the first time you show the dialog.

Remove

Clicking **Remove** removes the filter, showing all processes, and closes the dialog.

Cancel

Clicking **Cancel** closes the dialog without making any changes.

Help

Clicking **Help** presents this section of the manual in the online help viewer.

Trace Output Window

The Trace Output window is displayed from the Trace System Calls... or Trace Library Calls... option of the Process List context menu.

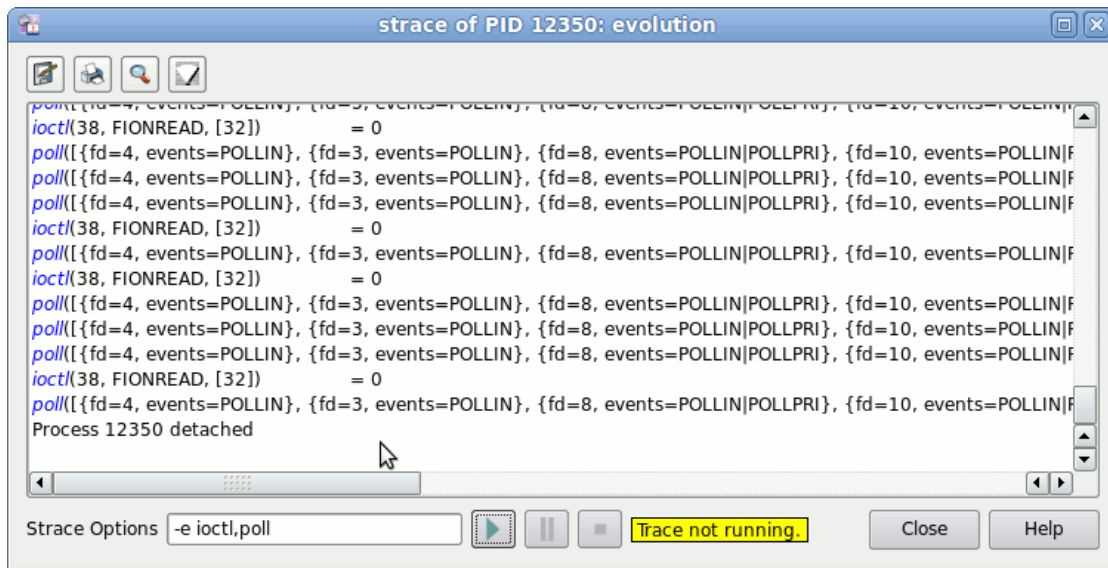










Figure 3-60. Trace Output Window

This window shows the output of an **strace(1)** or **ltrace(1)** on the selected process. Buttons provide the ability to save trace output to a file  or print the window .

You can search the trace output by clicking on the search button  or by typing Ctrl+F. This causes the find bar to appear. See “Find Bar” on page 3-86. Search for the same text again by pressing Enter in the find bar, by clicking on  Next or by typing Ctrl+G.

You can clear the trace text by clicking on the  icon. This does not pause or stop the trace, it merely removes the results already collected from the scrollable text area.

The output may be paused  and resumed . NightTune continues to gather data while the output is paused. The stored data is added to the window when the output is resumed.

The options text area at the bottom of the dialog allows you to specify options as you would to the corresponding **strace(1)** or **ltrace(1)** commands. Once the trace is running, you cannot change the options unless you stop the trace using the  icon.

The output is automatically scrolled so that new output is visible. To stop automatic scrolling, move the scrollbar away from the bottom. Automatic scrolling is also stopped after a successful search. Move the scrollbar to the bottom to enable automatic scrolling again.

Note that tracing a process can make a significant impact on its performance.

Click on the **Close** button to terminate the trace and close the dialog. Click on the **Help** button to see this section in the online help viewer.

Debug Process

Opens the NightView Source-Level Debugger and arranges for it to attach to the selected process. NightView is a graphical source-level debugging and monitoring tool specifically designed for real-time applications. NightView can monitor, debug, and patch multiple real-time processes running on multiple processors with minimal intrusion.

See also:

- *NightView RT User's Guide*

Process Scheduler Dialog

You can use the Process Scheduler dialog to alter the scheduling attributes and CPU affinity for any process for which you have appropriate privileges.

The Process Scheduler dialog is displayed when the process is selected in the Process List panel and Processor Scheduler is selected from the Process List context menu.

The following illustrates the Process Scheduler dialog:

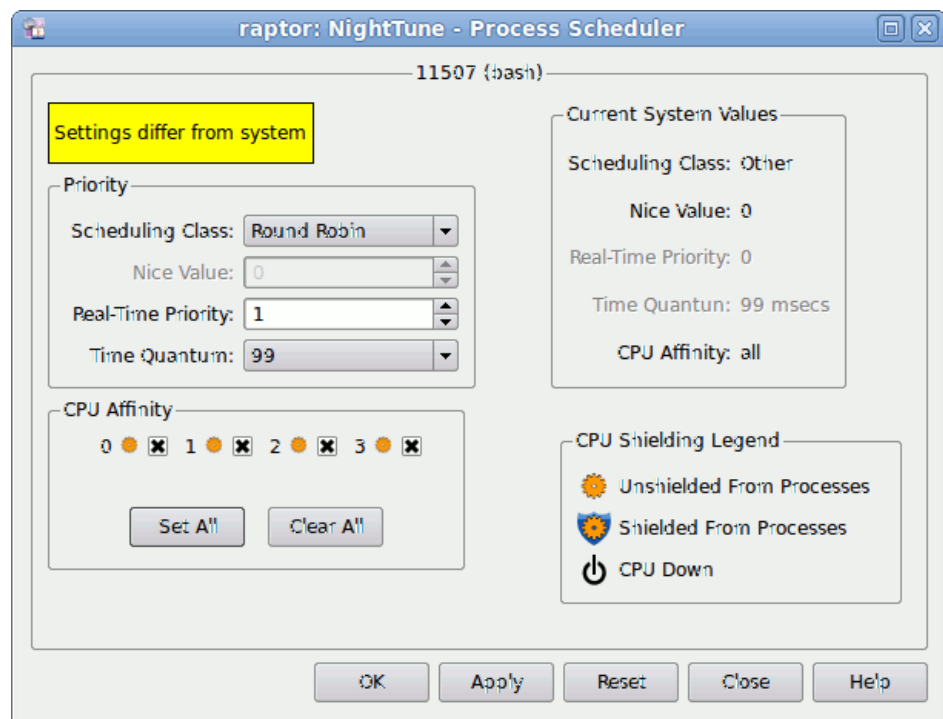


Figure 3-61. Process Scheduler Dialog

Descriptions of the text fields and controls contained in the Process Scheduler dialog follow:

Process

The process ID (PID) and simple name of the process currently referenced by the dialog is displayed at the top of the dialog. You can use this window as a drop target for processes; dropping a process onto the window changes the dialog to refer to that process. You cannot drop multiple processes or a process with more than one thread on the dialog.

Scheduling Class

This menu item allows you to select one of the three scheduling classes supported by the operating system:

Other

Selects the **SCHED_OTHER** scheduling policy which is the default universal time-sharing policy used by most processes. The priority of the process is first determined by the **Nice Value**, but then is adjusted by the operating system based on CPU utilization. This is a non-real-time policy, and processes using it will always have a less favorable priority than the real-time policies.

Batch

Selects the **SCHED_BATCH** scheduling policy, which is a variation on the **SCHED_OTHER** policy designed for CPU-bound “batch” processes. In addition to the **Nice Value**, a further small scheduling penalty is applied to wakeup behavior. This is a non-real-time policy, and processes using it will always have a less favorable priority than the real-time policies.

Idle

Selects the **SCHED_IDLE** scheduling policy, which is an extremely low priority policy. It is treated as even lower than **SCHED_OTHER** or **SCHED_BATCH** with a **Nice Value** of +19. This is a non-real-time policy.

First In-First Out

Selects the **SCHED_FIFO** scheduling policy. The priority of processes within this policy are static — they are not adjusted by the operating system. Processes retain use of the CPU until they block, voluntarily yield the CPU, or are preempted by higher priority processes or interrupts. This is a real-time policy.

Round Robin

Selects the **SCHED_RR** scheduling policy. It is very similar to **SCHED_FIFO**, but every process with this scheduling policy has an associated *time quantum*. In addition to the cases listed for **SCHED_FIFO**, if a process with policy **SCHED_RR** has been running for longer than its time quantum, it will be preempted and any other process with the same priority will be allowed to run. This is a real-time policy.

The current scheduling policy for the process is displayed within the **Current System Values** box.

See `sched_setscheduler(2)` for more information on the scheduling classes.

Nice Value

This text field allows you to specify a nice value for processes scheduled under the **Other** scheduling class. Nice values are inverted: lower numbered values have a more favorable priority than higher numbered values.

The current nice value for the process is displayed within the **Current System Values** box.

The **Nice Value** is only sensitized for processes using the **Other** scheduling class.

Real-Time Priority

This text field allows you to specify the priority within an associated real-time scheduling policy, either **Round Robin** or **First In-First Out**.

Values for the priority must be in the range 1..99. Higher numbered values have a more favorable priority than lower numbered values.

The current real-time priority for the process is displayed within the **Current System Values** box.

The **Real-Time Priority** is desensitized for processes using the **Other** scheduling class.

Time Quantum

This text field allows you to specify the duration in milliseconds of the execution time-slice for processes using the **Round Robin** scheduling class. It is not applicable to any other scheduling class.


The current time quantum for the process is displayed within the **Current System Values** box.

The **Time Quantum** is only sensitized for processes using the **Round Robin** scheduling class.

The operating system only supports certain values for the **Time Quantum**, and these vary between kernel versions. The combo-box shows the list of actual quanta available on your system.

CPU Affinity

The **CPU Affinity** area allows you to specify CPUs on which the process is allowed to execute. If a single CPU checkbox is checked, the process is bound to that CPU and the process is displayed in the **Processes** list in the associated CPU box in the **CPU Status** panel.

For each CPU that has been shielded from processes, the 'shielded from processors' icon  is displayed. If a CPU has been shielded from processes, no processes will execute on that CPU unless its affinity includes that CPU and no non-shielded CPUs.

The **Set All** and **Clear All** buttons eliminate having to click on each individual interrupt to set or clear all interrupts quickly.

For more information about CPU affinity and scheduling policies and priorities, refer to the `mpadvise(2)`, `sched_setaffinity(2)`, `sched_setscheduler(2)`, and `sched_setparam(2)` manual pages. Additional information is available in the *RedHawk Linux User's Guide*.

Current System Values

This area displays the values currently in effect for the process.

CPU Shielding Legend

This area displays the symbols used to indicate the status of the CPUs on the system: shielded from processes, unshielded from processes, and CPU down.

Process Scheduling Operations

The process scheduling operations are controlled using the buttons at the bottom of the dialog:

OK

Clicking **OK** applies any scheduling changes made in the dialog. You may not see a change reflected immediately since the operating system may defer certain operations until the next time the process becomes active. This is particularly likely if a process is being starved of CPU time; in this case try unbinding the process, or binding it to a different CPU, and making the change again. The dialog is closed when the actions are complete.

Apply

Clicking **Apply** applies any scheduling changes made in the dialog. You may not see a change reflected immediately since the operating system may defer certain operations until the next time the process becomes active. This is particularly likely if a process is being starved of CPU time; in this case try unbinding the process, or binding it to a different CPU, and making the change again.

Reset

Clicking **Reset** causes the current process scheduling attributes to be reflected in the dialog, discarding any changes that have not yet been applied.

Cancel

Clicking **Cancel** closes the dialog. Any changes that have not been applied are discarded.

Help

Clicking **Help** presents this section of the manual in the online help viewer.

Process Page Locking Dialog

The **Process Page Locking** dialog is displayed from the **Process Page Locking** option of the **Process List** context menu.

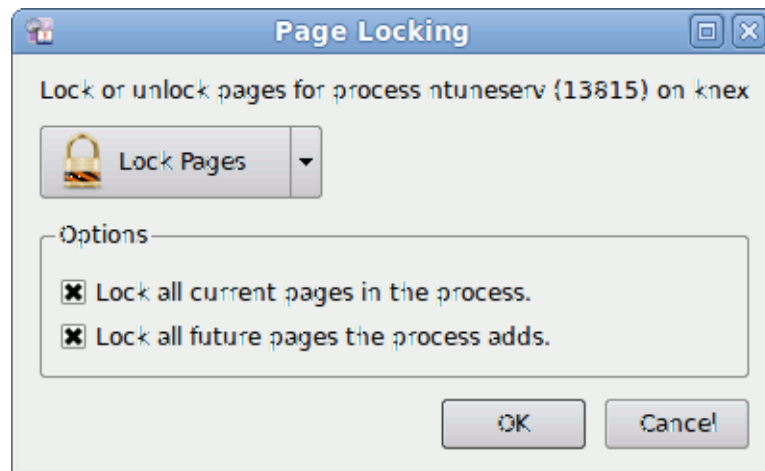


Figure 3-62. Process Page Locking Dialog

The dialog provides an option list that allows you to select either a locking or unlocking operation. When locking pages, there are two options. You can lock all current pages into memory and you can lock future pages into memory. These options correspond directly to the `MCL_CURRENT` and `MCL_FUTURE` flags associated with the `mlockall_pid(2)` system service. This capability is only available on recent RedHawk Linux kernels from Concurrent Real-Time. If your system lacks this service, a yellow diagnostic label will appear in the bottom portion of the dialog.

In order to lock the pages of your processes, or other processes, you must either be the `root` user or have the `CAP_SYS_LOCK` and `CAP_SYS_NICE` privileges. See “Capabilities” on page 1-3 for more information.

Process Details Window

The **Process Details** window is displayed from the **Process Details** option of the **Process List** context menu. This window contains memory usage, detailed information

about memory pages, file descriptors, signal status, capabilities and environment variables for the selected process in separate tabbed sections.

The information displayed in this dialog is a snapshot of the process at one point in time. Initially, it is the snapshot taken when the dialog was created. The user may press the Update button to obtain a new snapshot for the process.

The Close button may be used to close the dialog.

Memory Usage Tab

The Process Details Memory Usage tab provides an overview of memory usage for the process.

The following illustrates the Memory Usage tab:

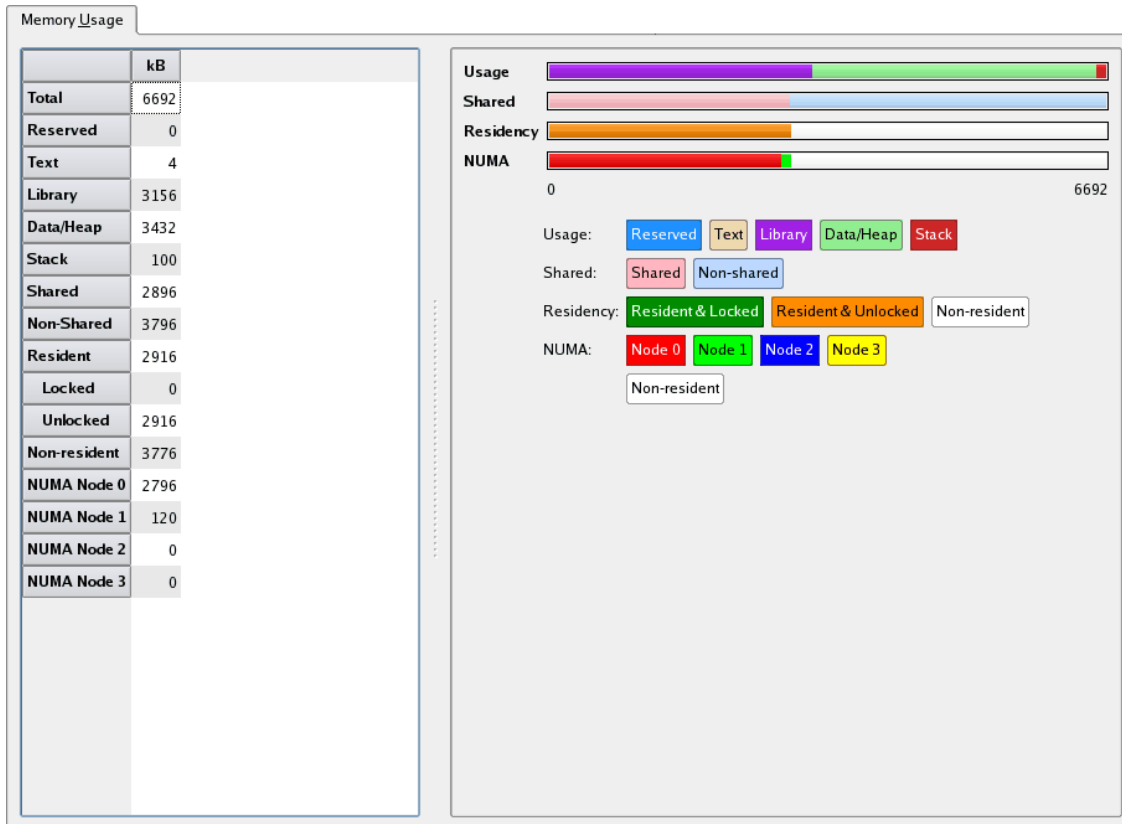


Figure 3-63. Process Details Memory Usage Tab

The total memory of the process is subdivided in four different, orthogonal, ways:

- Usage
- Shared / Nonshared
- Residency

- NUMA node

Subdivision by Usage

Total

Total virtual memory allocated for this process.

Reserved

Virtual memory reserved for I/O or by the kernel such that swapping of that memory is prohibited.

Text

Virtual memory used for executable instructions in the process' main executable.

Library

Virtual memory used for executable instructions in any shared libraries used by the process.

Data/Heap

Virtual memory used for static or heap data.

Stack

Virtual memory used for call stacks.

Subdivision by Shared / Nonshared

Total

Total virtual memory allocated for this process.

Shared

Virtual memory associated with a file.

Nonshared

Virtual memory not associated with a file.

Subdivision by Residency

Total

Total virtual memory allocated for this process.

Resident

Virtual memory stored in physical memory. This total is further subdivided into memory that is **Locked** in physical memory and that which is **Unlocked**, and therefore could become nonresident in the future.

Nonresident

Virtual memory not stored in physical memory, but rather on disk, either in swap space or in a disk file (if mapped).

Subdivision by NUMA node

One item exists for each NUMA node on the system, with one final item for non-resident memory. The item shows the total virtual memory allocated by the process from each NUMA node, or that is not resident in physical memory.

Memory Maps

The Process Details Memory Maps tab describes memory segments within the process's address space. A segment in this case is a range of memory, in ascending order, that shares Permission, Residency, NUMA Policy, and file mapping (if any) attributes.

This is the same information found in `/proc/pid/maps`, but presented in a table that allows you to change the sorting criteria by clicking on column headers.

Memory Maps							
Address Range	Size	Permissions	Resident	NUMA Policy	File Offsets	File Pathname	
0x400000 .. 0x7a1fff	0x3a2000	r-xp	0x1ed000	Default	0x0 .. 0x3a1fff	/usr/bin/ntune	
0x8a1000 .. 0x8bbfff	0x1b000	rw-p	0x1b000	Default	0x3a1000 .. 0x3bbfff	/usr/bin/ntune	
0x8bc000 .. 0x2445fff	0x1b8a000	rw-p	0x1a0c000	Default	0x0 .. 0x1b89fff	[heap]	
0x200000000 .. 0x8fffffff	0x700000000	---p	0x0	Default			
0x7fe467eff000 .. 0x7fe467efffff	0x1000	rw-p	0x1000	Default			
0x7fe468d00000 .. 0x7fe468d01fff	0x2000	r-xp	0x2000	Default	0x0 .. 0x1fff	/lib/libnss_mdns4.so.2	
0x7fe468d02000 .. 0x7fe468f00fff	0x1ff000	---p	0x0	Default	0x2000 .. 0x200fff	/lib/libnss_mdns4.so.2	
0x7fe468f01000 .. 0x7fe468f01fff	0x1000	r--p	0x1000	Default	0x1000 .. 0x1fff	/lib/libnss_mdns4.so.2	
0x7fe468f02000 .. 0x7fe468f02fff	0x1000	rw-p	0x1000	Default	0x2000 .. 0x2fff	/lib/libnss_mdns4.so.2	
0x7fe468f03000 .. 0x7fe468f18fff	0x16000	r-xp	0xf000	Default	0x0 .. 0x15fff	/lib/libresolv-2.11.1.so	
0x7fe468f19000 .. 0x7fe469117fff	0x1ff000	---p	0x0	Default	0x16000 .. 0x214fff	/lib/libresolv-2.11.1.so	
0x7fe469118000 .. 0x7fe469118fff	0x1000	r--p	0x1000	Default	0x15000 .. 0x15fff	/lib/libresolv-2.11.1.so	
0x7fe469119000 .. 0x7fe469119fff	0x1000	rw-p	0x1000	Default	0x16000 .. 0x16fff	/lib/libresolv-2.11.1.so	
0x7fe46911a000 .. 0x7fe46911bfff	0x2000	rw-p	0x1000	Default			
0x7fe46911c000 .. 0x7fe469120fff	0x5000	r-xp	0x5000	Default	0x0 .. 0x4fff	/lib/libnss_dns-2.11.1.so	
0x7fe469121000 .. 0x7fe46931ffff	0x1fff000	---p	0x0	Default	0x5000 .. 0x203fff	/lib/libnss_dns-2.11.1.so	
0x7fe469320000 .. 0x7fe469320fff	0x1000	r--p	0x1000	Default	0x4000 .. 0x4fff	/lib/libnss_dns-2.11.1.so	
0x7fe469321000 .. 0x7fe469321fff	0x1000	rw-p	0x1000	Default	0x5000 .. 0x5fff	/lib/libnss_dns-2.11.1.so	
0x7fe469322000 .. 0x7fe469323fff	0x2000	r-xp	0x2000	Default	0x0 .. 0x1fff	/lib/libnss_mdns4_minimal.so.2	
0x7fe469324000 .. 0x7fe469522fff	0x1ff000	---p	0x0	Default	0x2000 .. 0x200fff	/lib/libnss_mdns4_minimal.so.2	
0x7fe469523000 .. 0x7fe469523fff	0x1000	r--p	0x1000	Default	0x1000 .. 0x1fff	/lib/libnss_mdns4_minimal.so.2	
0x7fe469524000 .. 0x7fe469524fff	0x1000	rw-p	0x1000	Default	0x2000 .. 0x2fff	/lib/libnss_mdns4_minimal.so.2	
0x7fe469525000 .. 0x7fe469580fff	0x5c000	r-xp	0x2b000	Default	0x0 .. 0x5bfff	/usr/lib/libnvidia-ml.so.304.54	
0x7fe469581000 .. 0x7fe46977ffff	0x1ff000	---p	0x0	Default	0x5c000 .. 0x25afff	/usr/lib/libnvidia-ml.so.304.54	

Figure 3-64. Process Details Memory Maps Tab

Memory Tab

The Process Details Memory tab provides the ability to view information about individual pages in a process' address space. The display can be zoomed and panned to represent the entire address space or as few as four pages. Various methods allow navigation to all areas of the address space.

The following illustrates the Memory tab:

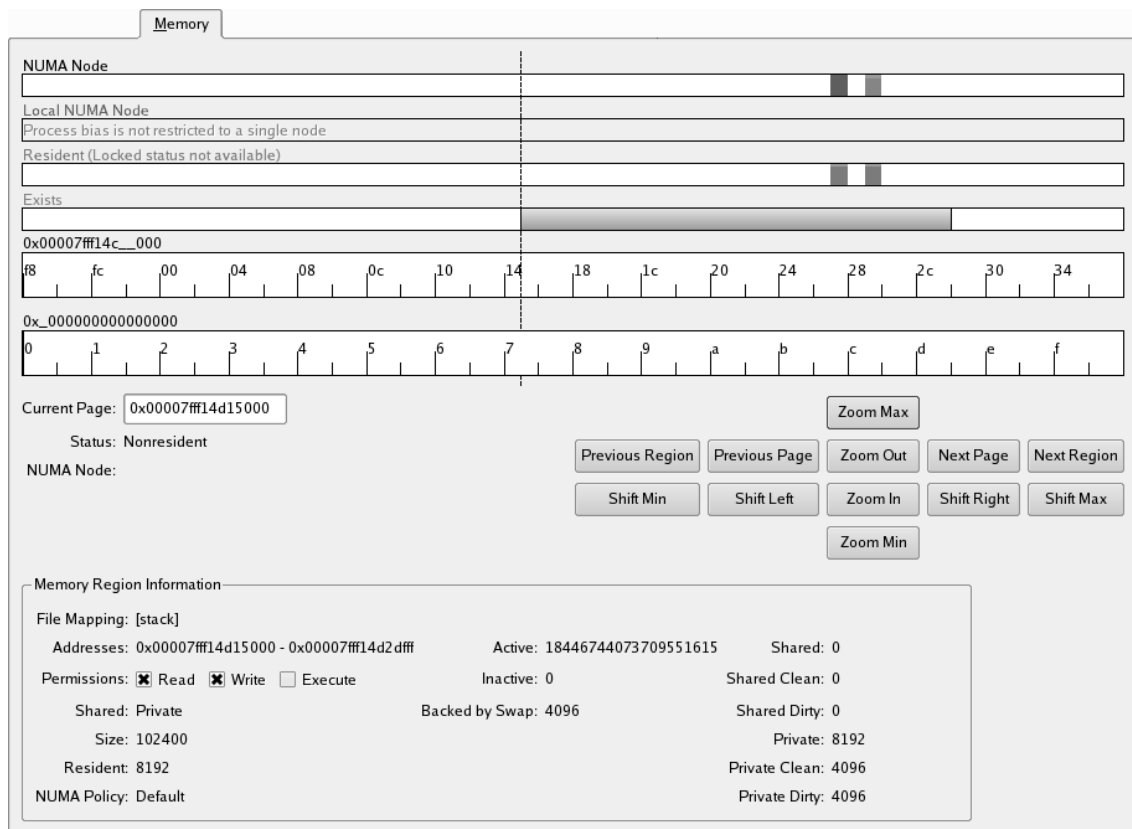


Figure 3-65. Process Details Memory Tab

The horizontal axis in the bars on this tabbed page is used to describe memory addresses.

The bottommost bar is called the **Global Address Bar** and always covers the entire range of the address space of the process. It exists to provide contextual information about the position and size of the memory region being viewed by all the other bars when they are zoomed or panned.

All the other bars are synchronized to each other to show information for the zoomed or panned range of memory addresses. A vertical dashed line spans all the bars and shows the location of the **Current Page**, about which additional information is shown below the bars.

The bar second from the bottom is called the **Address Bar** and describes the memory addresses being viewed by the current zoom & pan settings. The bars above it describe characteristics of the pages associated with those memory addresses. Their meanings are:

NUMA Node

The vertical bars in this graph indicate which NUMA node contains the physical memory associated with this virtual memory page. Each NUMA node is assigned a color. It displays bars only on a NUMA system, and only for pages which are resident. The vertical lines in this bar are synchronized with those in the **Address Bar**.

Local NUMA Node

Vertical bars are displayed in this graph only on a NUMA system, and only when the process has a CPU affinity whose CPUs are all from a single NUMA node. In that case, the vertical bars indicate pages which are contained in the memory associated with that NUMA node. The vertical lines are synchronized with those in the **Address Bar**.

Resident

Vertical bars in this bar indicate pages which are resident in memory. The vertical lines in this bar are synchronized with those in the **Address Bar**.

Exists

Grey vertical bars in this bar indicate pages which are mapped as part of the address space. The boundaries of distinct memory regions are denoted by black vertical lines. The vertical lines in this are synchronized with the **Address Bar**.

Address Bar

This bar represents the part of the address space being viewed. If zoomed all the way out, this covers the entire address space. But the panning and zooming operations allow it to represent a smaller part of the total address space. The label above it (e.g., 0x__0000000) indicates with the “_” what the numbers in the bar represent (e.g., 0x73000000).

Global Address Bar

This bar represents the entire address space. The shaded area shows which part of the address space is being viewed by all the other bars. A vertical line in this bar represents the current page. The label above it (e.g., 0x_0000000) indicates with the “_” what the numbers in the bar represent (e.g., 0x80000000).

Current Page

This gives the first address within the current page. The current page also is indicated by the vertical dashed line running through the horizontal bars. The current page may be changed by typing a new value into the **Current Page** field, or by clicking in the horizontal bars.

Status

Indicates the locked and residency status of the page, which may be one of the following:

- **Locked:** forced into physical memory (also implies Resident)
- **Resident:** in physical memory
- **Non-resident:** not in physical memory

NUMA Node

For pages resident in physical memory on a NUMA system, this shows the NUMA node which contains the page's physical memory.

Navigation Buttons

Clicking on these buttons adjusts the display:

- **Zoom Max (or Alt-Up)** zooms out to show the entire address space
- **Zoom Min (or Alt-Down)** zooms in to show four pages (the minimum number)
- **Zoom Out (or Up)** zooms out the visible range of addresses by a factor of two to view more of the address space, but with less detail
- **Zoom In (or Down)** zooms in the visible range of addresses by a factor of two to view less of the address space, but with more detail
- **Previous Region (or Alt-Left)** searches backward from the current page to the preceding memory region and makes the last page of that region the current page
- **Next Region (or Alt-Right)** searches forward from the current page to the preceding memory region and makes the first page of that region the current page
- **Previous Page (or Left)** sets the current page to the previous page
- **Next Page (or Right)** sets the current page to the next page
- **Shift Min (or Home)** sets the current page to the lowest page in the address space
- **Shift Max (or End)** sets the current page to the highest page in the address space
- **Shift Left (or Ctrl-Left)** shifts the current page to the left (lower numbered address) by approximately 25% of the addresses currently visible
- **Shift Right (or Ctrl-Right)** shifts the current page to the right (higher numbered address) by approximately 25% of the addresses currently visible

Memory Region Information

Information in this area applies to a contiguous region of memory which was mapped as one operation and which contains the current page. The memory may be private to the process or may be mapped to a file. If mapped as a file, the memory

will be from a contiguous region of that file and will have consistent permissions. The pieces of information are:

- **File Mapping** shows the name of the file with which this memory is associated, if any
- **Addresses** shows the range of memory addresses for this memory region
- **Permissions** shows the **Read**, **Write**, and **Execute** permissions for the pages in this memory region
- **Shared** shows whether the pages in the memory region are:
 - **Shared**, where modifications to the pages affect any associated file
 - **Private**, where modifications to the pages affect an in-memory copy of the content from the associated file and not the file itself
- **Size**, which shows the number of bytes in the memory region
- **Resident**, which shows the number of bytes from the memory region that are resident in physical memory
- **NUMA Policy**, which describes how memory from this region will be assigned to NUMA nodes. It may be one of:
 - **Default**, meaning that pages will be selected from the NUMA node associated with the CPU on which the process is running when the mapping occurs, if possible; or from any NUMA node otherwise
 - **Preferred**, meaning that pages will be selected from a specified NUMA node, if possible; or from any NUMA node otherwise
 - **Bind**, meaning that pages will be selected from a specified set of NUMA nodes and that those nodes are enforced strictly (i.e. if no memory is available on any of those nodes, an error will occur)
 - **Interleaved**, meaning that pages will be selected from a specified set of NUMA nodes, assigned in a round-robin order
- **Active**, shows the number of bytes in pages in this memory region which have been used recently and therefore are unlikely to be reclaimed when memory is needed
- **Inactive**, which shows the number of bytes in pages in this memory region which have been used less recently and therefore are likely to be reclaimed when memory is needed
- **Backed by Swap**, which shows the number of bytes in pages for which swap space has been allocated

- **Shared**, which shows the number of bytes in pages which are shared by at least two processes with independent address spaces. It is further subdivided into:
 - **Shared Clean**, which shows the number of bytes in Shared pages for which the physical memory is consistent with any associated file or swap space
 - **Shared Dirty**, which shows the number of bytes in Shared pages for which the physical memory is not consistent with any associated file or swap space
- **Private**, which shows the number of bytes in pages which are not shared by at least two processes with independent address spaces. It is further subdivided into:
 - **Private Clean**, which shows the number of bytes in Private pages for which the physical memory is consistent with any associated file or swap space
 - **Private Dirty**, which shows the number of bytes in Private pages for which the physical memory is not consistent with any associated file or swap space

File Descriptors Tab

The Process Details File Descriptors tab lists the files or devices associated with the file descriptors for the process.

The following illustrates the File Descriptors tab:

File Descriptors	
	Pathname/Description
0	/dev/pts/24
1	/dev/pts/24
2	/dev/pts/24
3	pipe:[10271195] (pid 4769/ntunegui fd 4)
4	pipe:[10271195] (pid 4769/ntunegui fd 3)
5	pipe:[10271198] (pid 4769/ntunegui fd 6)
6	pipe:[10271198] (pid 4769/ntunegui fd 5)
7	socket:[10271199]: tcp: local=hyena:37895 remote=hyena:x11-ssh-offset(6010) state=ESTABLISHED
8	/usr/lib/NightTune/lib/ntune.msg
9	socket:[10271209]: tcp: local=hyena:37896 remote=hyena:25517 state=ESTABLISHED
10	/proc
11	/proc/shield/irqs
12	/proc/shield/tmrs
13	/proc/shield/procs
14	/proc/ccur/switches
15	/proc/stat
16	/proc/meminfo
17	/proc/vmstat
18	/proc/diskstats
19	/proc/interrupts
20	/proc/net/dev
21	/proc/4769/fd

Figure 3-66. Process Details File Descriptors Tab

The following formats are used:

filename

The file descriptor is associated with the name *filename*.

filename (deleted)

The file descriptor is associated with a file that previously had the name *filename*, but which has been deleted since this process opened it.

pipe:[*inode*] (*other-pid*) ...

The file descriptor is associated with a pipe with the specified *inode* number. If any other processes on the sam system also have the pipe open, they will be listed as *other-pid*.

socket:[*inode*]: tcp/udp/raw: local=*ip:port* remote=*ip:port* state=*s* (*other-pid*) ...

The file descriptor is associated with a TCP, UDP, or RAW socket with the specified *inode* number. The local and remote IP addresses and ports are shown, if available. They will be displayed as symbolic names if possible. If the protocol is stateful (e.g. TCP), state information is shown. States are values like ESTABLISHED, LISTEN, FIN_WAIT1, etc. If the remote end of the socket is on the same system and some other process is connected to that remote end, then the other process will be listed as *other-pid*.

socket:[*inode*]: unix/*type*: name=*associated-filename* state=*s*

The file descriptor is associated with a UNIX domain socket with the specified *inode* number. It may have a *type* of either **stream** or **dgram**. If the socket is associated with a file, its *associated-filename* will be shown. If the protocol is stateful (e.g. stream), state information is shown. States are values like LISTENING, CONNECTED, etc.

socket:[*inode*]: packet

The file descriptor is associated with a PACKET socket with the specified *inode* number.

If the file type allows it and the kernel supports it, the following additional information may be specified:

pos=*pos*

The file offset is position *pos*.

mode=*mode*

mode describes the access mode used to open the file, which may be O_RDONLY, O_WRONLY, or O_RDWR.

flags=[*flag*...]

Each *flag* describes an additional flag used when opening the file. Flags include O_CREAT, O_EXCL, O_APPEND, O_TRUNC, O_SYNC, etc.

Signals Tab

The Process Details Signals tab shows the signal handling capabilities of the process.

The following illustrates the Signals tab:

Number	Name	Pending	Shared Pending	Blocked	Ignored	Handled	Restart	Description
1	SIGHUP	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Hangup
2	SIGINT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Interrupt
3	SIGQUIT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Quit
4	SIGILL	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Illegal instruction
5	SIGTRAP	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Trace/breakpoint trap
6	SIGABRT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Aborted
7	SIGBUS	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Bus error
8	SIGFPE	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Floating point exception
9	SIGKILL	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Killed
10	SIGUSR1	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	User defined signal 1
11	SIGSEGV	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Segmentation fault
12	SIGUSR2	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	User defined signal 2
13	SIGPIPE	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Broken pipe
14	SIGALRM	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Alarm clock
15	SIGTERM	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Terminated
16	SIGSTKFLT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Stack fault
17	SIGCHLD	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Child exited
18	SIGCONT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Continued
19	SIGSTOP	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Stopped (signal)
20	SIGTSTP	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Stopped
21	SIGTTIN	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Stopped (tty input)
22	SIGTTOU	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Stopped (tty output)
23	SIGURG	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Urgent I/O condition

Figure 3-67. Process Details Signals Tab

The columns' meanings are:

Number

The numeric value of the signal.

Name

The symbolic name of the signal.

Pending

If this column is checked, then an instance of this signal is pending for this process (and this particular thread if multi-threaded).

Shared Pending

If this column is checked, then an instance of this signal is pending for this process (and may be accepted by any thread in the process).

Blocked

If this column is checked, then the process has blocked delivery of this signal.

Ignored

If this column is checked, then the process is ignoring any instances of this signal.

Handled

If this column is checked, then the process has a signal handler for this signal.

Restart

If this column is checked, then if this signal interrupts a system call, it will be restarted after the signal is handled (SA_RESTART).

Description

A description of the signal.

Capabilities Tab

The Process Details Capabilities tab shows the capabilities associated with the process.

The following illustrates the Capabilities tab:

Number	Name	Inheritable	Permitted	Effective
0	CAP_CHOWN	<input type="checkbox"/>	✘	✘
1	CAP_DAC_OVERRIDE	<input type="checkbox"/>	✘	✘
2	CAP_DAC_READ_SEARCH	<input type="checkbox"/>	✘	✘
3	CAP_FOWNER	<input type="checkbox"/>	✘	✘
4	CAP_FSETID	<input type="checkbox"/>	✘	✘
5	CAP_KILL	<input type="checkbox"/>	✘	✘
6	CAP_SETGID	<input type="checkbox"/>	✘	✘
7	CAP_SETUID	<input type="checkbox"/>	✘	✘
8	CAP_SETPCAP	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
9	CAP_LINUX_IMMUTABLE	<input type="checkbox"/>	✘	✘
10	CAP_NET_BIND_SERVICE	<input type="checkbox"/>	✘	✘
11	CAP_NET_BROADCAST	<input type="checkbox"/>	✘	✘
12	CAP_NET_ADMIN	<input type="checkbox"/>	✘	✘
13	CAP_NET_RAW	<input type="checkbox"/>	✘	✘
14	CAP_IPC_LOCK	<input type="checkbox"/>	✘	✘
15	CAP_IPC_OWNER	<input type="checkbox"/>	✘	✘
16	CAP_SYS_MODULE	<input type="checkbox"/>	✘	✘
17	CAP_SYS_RAWIO	<input type="checkbox"/>	✘	✘
18	CAP_SYS_CHROOT	<input type="checkbox"/>	✘	✘
19	CAP_SYS_PTRACE	<input type="checkbox"/>	✘	✘
20	CAP_SYS_PACCT	<input type="checkbox"/>	✘	✘
21	CAP_SYS_ADMIN	<input type="checkbox"/>	✘	✘
22	CAP_SYS_BOOT	<input type="checkbox"/>	✘	✘

Figure 3-68. Process Details Capabilities Tab

The columns' meanings are:

Number

The numeric value of the capability.

Name

The symbolic name of the capability.

Inheritable

If this column is checked, then any new process created by this process (e.g. with `fork(2)`) will inherit the capability.

Permitted

If this column is checked, then the process is permitted to have the capability, even if it voluntarily disabled it.

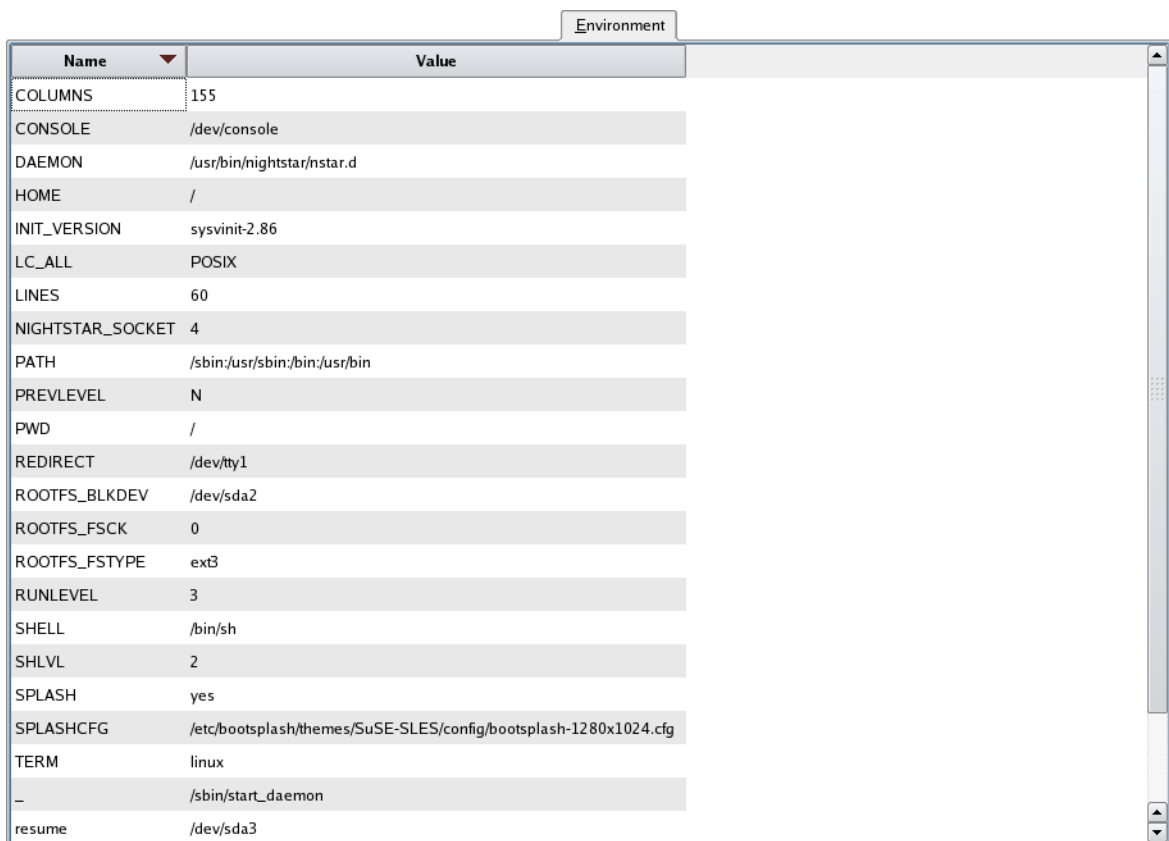
Effective

If this column is checked, then the process has the capability currently.

Environment Tab

The Process Details Environment tab lists all environment variables associated with the process and their values.

The following illustrates the Environment tab:



Name	Value
COLUMNS	155
CONSOLE	/dev/console
DAEMON	/usr/bin/nightstar/nstar.d
HOME	/
INIT_VERSION	sysvinit-2.86
LC_ALL	POSIX
LINES	60
NIGHTSTAR_SOCKET	4
PATH	/sbin:/usr/sbin:/bin:/usr/bin
PREVLEVEL	N
PWD	/
REDIRECT	/dev/tty1
ROOTFS_BLKDEV	/dev/sda2
ROOTFS_FSCK	0
ROOTFS_FSTYPE	ext3
RUNLEVEL	3
SHELL	/bin/sh
SHLVL	2
SPLASH	yes
SPLASHCFG	/etc/bootsplash/themes/SuSE-SLES/config/bootsplash-1280x1024.cfg
TERM	linux
_	/sbin/start_daemon
resume	/dev/sda3

Figure 3-69. Process Details Environment Tab

Limits Tab

The Process Details Limits tab lists the values of all resource limits for the process.

The following illustrates the Limits tab:

Number ▲	Name	Soft Limit	Hard Limit	Description
0	RLIMIT_CPU	unlimited	unlimited	Max cpu time (s)
1	RLIMIT_FSIZE	unlimited	unlimited	Max file size (bytes)
2	RLIMIT_DATA	unlimited	unlimited	Max data size (bytes)
3	RLIMIT_STACK	unlimited	unlimited	Max stack size (bytes)
4	RLIMIT_CORE	0	unlimited	Max core file size (bytes)
5	RLIMIT_RSS	unlimited	unlimited	Max resident set (bytes)
6	RLIMIT_NOFILE	1024	62537	Max processes
7	RLIMIT_AS	1024	1024	Max open files
8	RLIMIT_NPROC	65536	65536	Max locked memory (bytes)
9	RLIMIT_MEMLOCK	4294967296	unlimited	Max address space (bytes)
10	RLIMIT_LOCKS	unlimited	unlimited	Max file locks
11	RLIMIT_SIGPENDING	62537	62537	Max pending signals
12	RLIMIT_MSGQUEUE	819200	819200	Max msgqueue size (bytes)
13	RLIMIT_NICE	0	0	Max nice priority
14	RLIMIT_RTPRIO	0	0	Max realtime priority
15	RLIMIT_RTIME	unlimited	unlimited	Max realtime timeout (us)

Figure 3-70. Process Details Limits Tab

The columns' meanings are:

Number

The numeric identifier of the resource limit.

Name

The symbolic name of the resource limit.

Soft Limit

The effective limit for the resource. It is adjustable downward or upward by the user using `setrlimit(2)` or `ulimit(1)`. It cannot be adjusted above the hard limit.

Hard Limit

This is the maximum value allowed for the soft limit. It can be adjusted downward using `setrlimit(2)` or `ulimit(1)`. It cannot be adjusted upward except by the root user.

Description

A description of the source limit.

Process Fields Menu

The Process Fields menu is displayed from the Display Fields option of the Process List context menu. This menu allows you to select which fields (columns) are visible in the Process List panel.

The following illustrates the Process Fields menu:

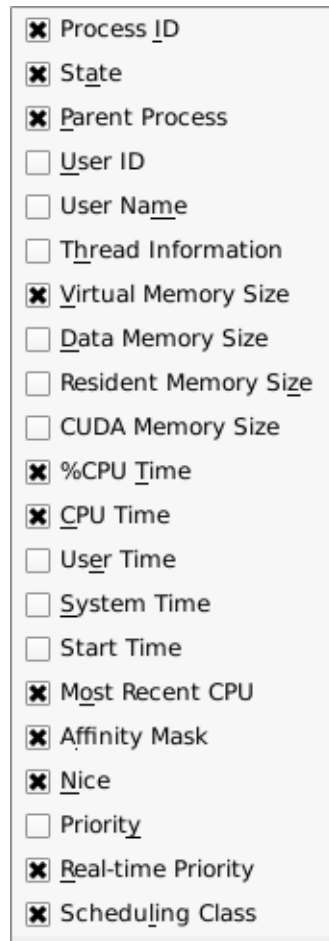


Figure 3-71. Process Fields Menu

NOTE

The **Process Fields** menu, like the other menus, can be “torn off” from the menu bar to reside in its own window separate from NightTune by clicking on the dashed line at the top of the menu. This is especially useful when making multiple changes to avoid having to select the menu after every choice.

The following paragraphs describe each of the selectable fields in more detail:

Process ID

Mnemonic: I

This field displays the **PID** column which contains the process ID as returned by `getpid(2)`.

State

Mnemonic: A

This field displays the **State** column, which contains the process state, which can be one of:

- **Running**: currently running or on a run queue and able to run
- **Waiting**: also known as sleeping
- **Disk**: performing an uninterruptible sleep (usually disk I/O)
- **Stopped**: stopped or traced
- **Paging**
- **Zombie**: a defunct process

Parent Process

Mnemonic: P

This field displays the **Parent** column, which contains the process ID of the parent process.

User ID

Mnemonic: U

The field displays the **UID** column, which contains the user ID of the process.

User Name

Mnemonic: M

This field displays the **Username** column, which contains user name corresponding to the user ID.

Thread Information

Mnemonic: H

This field displays the **Threads** column, which contains the number of threads in the process.

Virtual Memory Size

Mnemonic: V

This field displays the **Size** column, which contains the number of kB of virtual memory associated with the process.

Data Memory Size

Mnemonic: D

This field displays the **Data** column, which contains an approximation of the memory used by the program. It is essentially the total number of kB of virtual memory used by the process less memory used for instruction (`.text`) pages. It includes data pages used by shared libraries.

Resident Memory Size

Mnemonic: Z

This field displays the **Resident** column, which contains the number of kB of memory associated with the process that is resident in physical memory.

CUDA Memory Size

This field displays the **CUDA** column, which contains the number of kB of CUDA device memory used by the program. If multiple devices are present, the number includes all CUDA memory on all devices.

%CPU Time

Mnemonic: T

This field displays the **%CPU** column, which contains the percentage of CPU time used by the process.

CPU Time

Mnemonic: C

This field displays the **CPU Time** column, which contains the amount of CPU time used by the process in seconds.

User Time

Mnemonic: E

This field displays the **User** column, which contains the amount of time used on behalf of the process, excluding time spent in system calls, in seconds.

System Time

Mnemonic: S

This field displays the **System** column, which contains the amount of time used on behalf of the process for system calls in seconds.

Start Time

This field displays the **Start Time** column, which contains the time at which the process started. If the process started on a previous day, the date is included. The start time can be off by one hour if the system has been running over a period of time where daylight savings time changes.

Most Recent CPU

Mnemonic: O

This field displays the **CPU** column, which contains the number of the CPU upon which the process last executed.

Affinity Mask

Mnemonic: F

This field displays the **Affinity** column, which contains the CPU affinity mask which specifies on which CPUs the process can execute. The column may display the term **all**, indicating that the process is free to run on any CPU on the system (excluding shielded CPUs). When the CPU affinity does not designate all CPUs on the system, the mask is displayed as a hexadecimal number. The least significant bit in the mask represents logical CPU 0.

Nice

Mnemonic: N

This field displays the **Nice** column, which contains the *nice* value as set by the **nice (1)** command or the **nice (2)** system service. The nice value provides an initial basis used by the kernel for determining the priority of processes in the **SCHED_OTHER**, **SCHED_BATCH**, or **SCHED_IDLE** scheduling classes. Lower values represent more favorable priorities.

Priority

Mnemonic: Y

This field displays the `Pri` column, which contains the internal kernel priority value. Lower values represent more favorable priorities. For processes in the `SCHED_OTHER`, `SCHED_BATCH`, or `SCHED_IDLE` scheduling classes, the priority is adjusted by the kernel as the process runs, based on CPU utilization. For processes in other scheduling classes, the values are determined from the real-time priority, negated and adjusted by a bias.

Real-time Priority

Mnemonic: R

This field displays the `RPri` column, which contains the real-time priority within the process's scheduling class, as specified by the program or by the `run (1)` command. Higher values represent more favorable priorities.

Scheduling Class

Mnemonic: L

This field displays the `CL` column, which contains the scheduling class associated with the process. The possible values are:

- OT: other policy (non-real-time)
- BA: batch policy (non-real-time)
- ID: idle policy (non-real-time)
- FF: FIFO policy (real-time)
- RR: round-robin policy (real-time)

The scheduling classes are explained in a bit more detail in “Scheduling Class” on page 3-98, and defined in even greater detail in `sched_setscheduler (2)`.

Single Process Activity Panel

The Single Process Activity panel displays information about the amount of time a user-specified process spends within each of its functions. It determines this information by taking a snapshot of the location where the process is executing periodically with a

user-specified period (see “System Status” on page 2-23) measured in CPU cycles. It displays this information sorted by number of hits for each function which executed. This panel depends on performance counter support on both the CPU and the Linux kernel.

Read the sub-section on “Required Privileges for the Single Process Activity/Counters Panels” on page 3-135 for important information.

The following illustrates the panel:

Count	Percent	Function
46	5.3%	QFontEngineMultiFT::QFontEngineMultiFT(QFontEngine*, _FcPattern*, int, QFontDef const&) [libQtGui.so.4.2.1]
30	3.5%	check_bytes [kernel]
15	1.7%	_init [libQtGui.so.4.2.1]
14	1.6%	do_select [kernel]
13	1.5%	_select_nocancel [libc-2.5.so]
13	1.5%	find_busiest_group [kernel]
12	1.4%	QCoreApplication::sendPostedEvents(QObject*, int) [libQtCore.so.4.2.1]
12	1.4%	QEventDispatcherUNIX::activateTimers() [libQtCore.so.4.2.1]
12	1.4%	_int_malloc [libc-2.5.so]
12	1.4%	malloc [libc-2.5.so]
11	1.3%	_raw_spin_unlock [kernel]
10	1.2%	PackageTable::SortFilterProxyModel::acceptRow(Package::Entry*, int, QModelIndex const&) const
10	1.2%	QEventDispatcherUNIXPrivate::doSelect(QFlags<QEventLoop::ProcessEventsFlag>, timeval*) [libQtCore.so.4.2.1]
10	1.2%	_int_free [libc-2.5.so]
9	1.0%	_init [libQtCore.so.4.2.1]
9	1.0%	memset_c [kernel]
9	1.0%	sub_preempt_count [snd] [kernel]
8	0.9%	operator<<(QDebug, QMatrix const&) [libQtGui.so.4.2.1]
7	0.8%	QEventDispatcherUNIX::registeredTimers(QObject*) const [libQtCore.so.4.2.1]
7	0.8%	free [libc-2.5.so]
7	0.8%	FT_Get_X11_Font_Format [libfontconfig.so.2.1.0]
7	0.8%	operator new(unsigned long) [libstdc++.so.6.0.8]
7	0.8%	copy_user_generic_string [kernel]
7	0.8%	_raw_spin_lock [kernel]
7	0.8%	schedule [snd] [kernel]

Figure 3-72. Single Process Activity Panel

The header indicates the proportion of activity broken into 4 categories:

- **Static**, functions which exist in the static executable
- **Shared object**, functions which exist in shared objects dynamically linked into the executable
- **Kernel**, functions executed within the kernel while the process is actively scheduled on that CPU (generally, kernel activity on behalf of the process)
- **Unknown**, functions which could not be categorized above (such functions would be very exotic)

It also displays the user-specified **Sample Period**, measured in CPU cycles. If the sample period is set too low, the kernel may throttle the number of snapshots taken, artificially increasing the sample period beyond the user-specified number. If this happens, the header will display (throttled). Finally, if any errors occurred, the most recent will be displayed in the header. Any previous errors can be viewed in a tooltip by hovering the mouse over the most recent error.

The table displays the list of functions. Static functions are displayed in black. Functions in dynamically linked shared objects are displayed in blue and are annotated with the name of the shared object, such as [libc-2.5.so]. Kernel functions are displayed in tan and are annotated with [kernel]. Unknown functions, if any, are displayed in red.

For each function, it displays **Count**, the number of times that the function was encountered by a snapshot. It also displays **Percent**, the percentage of times that the snapshot encountered the function out of the total number of snapshots.

Single Process Activity Context Menu

While positioned in a single process activity panel, right-clicking displays a context menu, which contains the following content:

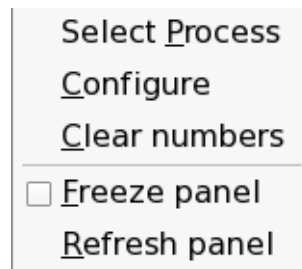


Figure 3-73. Single Process Activity Context Menu

The following paragraphs describe the menu items in detail:

Select Process

Mnemonic: P

The menu item opens a **Select Process** dialog to specify the process to be watched. See “Single Process Panel” on page 3-6 for details.

Configure

Mnemonic: C

The menu item opens a **Configure** dialog to configure how the process should be watched. See “Configure Activity Dialog” on page 3-124 for details.

Clear numbers

Mnemonic: C

This menu item manually resets all the counts for every function to zero. Use this to clear away all existing counts and start fresh, for instance right before starting some operation within the application to test.

Freeze/Unfreeze panel

Mnemonic: F

This menu item toggles the **Freeze** setting for the panel. When frozen, data values are not refreshed automatically. This menu item overrides the **Freeze** setting for the window, but only applies to the particular panel.

Refresh panel

Mnemonic: R

This menu item causes all data within the panel to be refreshed once, regardless of the **Freeze** setting.

Configure Activity Dialog

The Configure Activity dialog allows the user to specify how a **Single Process Activity** panel should watch its process. The following illustrates the dialog:

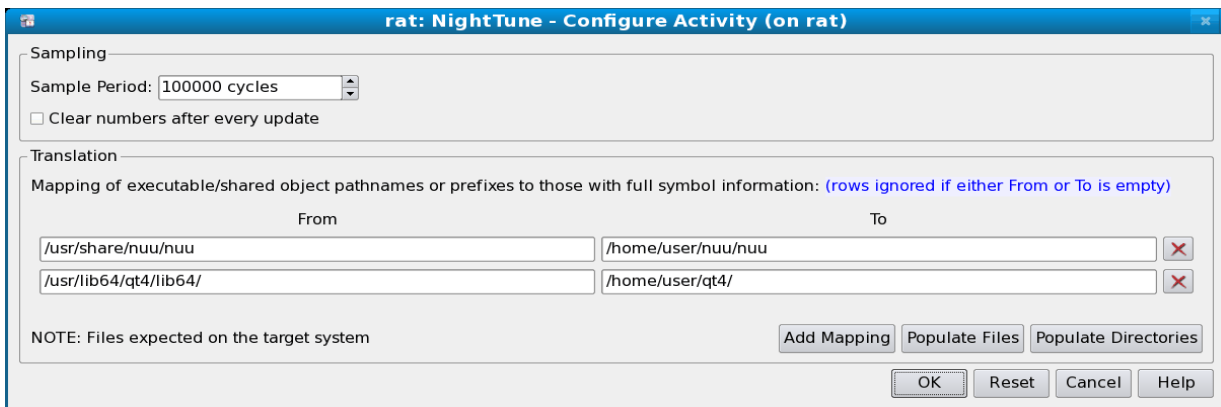


Figure 3-74. Single Process Configure Activity Dialog

The **Sampling** area contains the **Sample Period**, which determines the number of cycles between samples of the process. Note that if this period is excessively small, the kernel will throttle the number of samples, artificially raising it to a tolerable number. If the **Clear numbers after every update** checkbox is checked, then whenever the panel is updated (e.g. based on the **Single Process Update Interval**), the counts are cleared. This results in the panel displaying only counts for the most recent update interval. If unchecked, then the counts accumulate over the lifetime of the panel.

The **Translation** area allows the user to override the name of the executable or any dynamically loaded shared objects. This is useful if the actual executable or shared object(s) do not have full symbols (e.g. they were stripped), but files with those symbols exist elsewhere.

Each row constitutes a mapping. A pathname in the **From** column is translated to a corresponding pathname in the **To** column. Alternately, a pathname prefix (e.g. directory path) in the **From** column will be substituted with the pathname prefix in the **To** column for any matching pathname. If either the **From** or **To** columns is empty, the mapping is ignored.

To add an additional mapping, use the **Add Mapping** button. To delete a mapping, either press the **✕** button to the right of the mapping, or clear either the **From** or **To** columns.

To simplify the process of adding mappings, the **Populate Files** button may be used. It adds template mappings for every executable and shared object currently loaded by the process, with the **From** column containing the pathname and the **To** column left empty. Because the **To** columns are empty, the mappings start out ignored. But the user may add pathnames to any of the **To** columns to quickly create useful mappings. Similarly, the **Populate Directories** button adds template mappings, but only for each unique directory prefix for the currently loaded executable and shared objects.

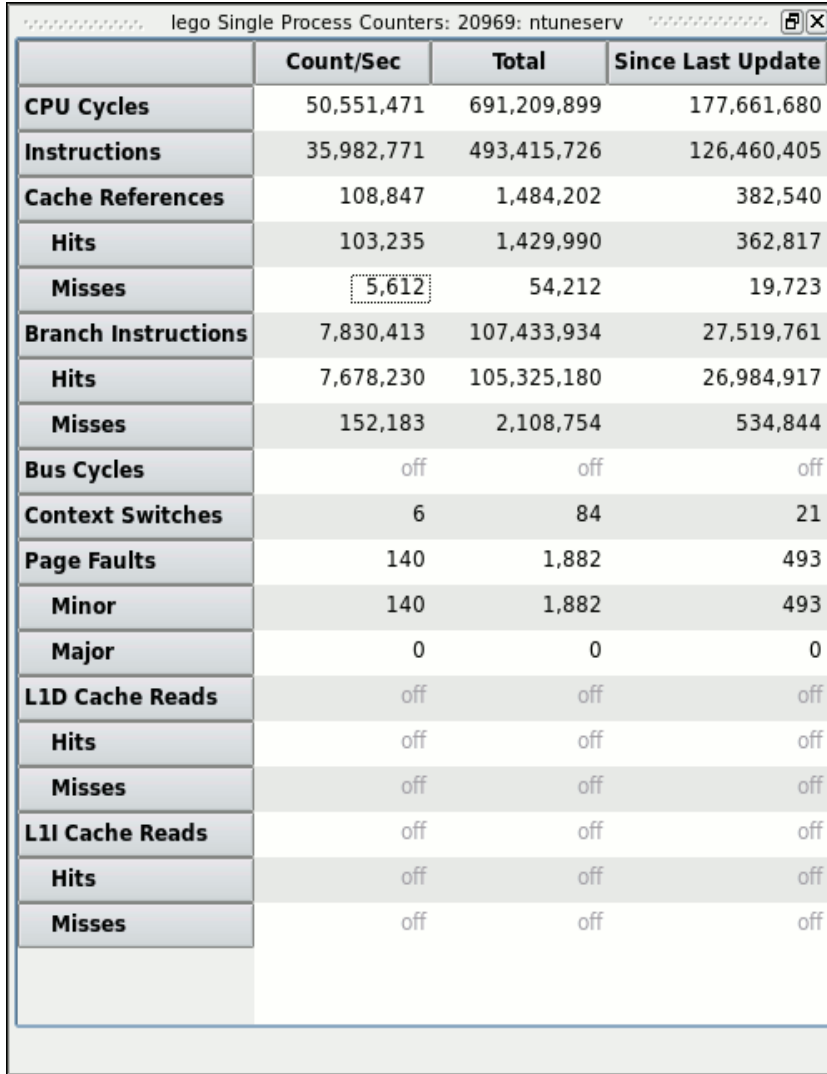
Single Process Counters Panel

The **Single Process Counters** panel displays various hardware and kernel counters for a single process. Although it is a single process panel, it is divided into **Text**, **Bar graph**, and **Line graph** panes much like **System Status** panels (see “**System Status Panel**” on page 3-1). This panel depends on performance counter support on both the CPU and the Linux kernel.

Read the sub-section on “**Required Privileges for the Single Process Activity/Counters Panels**” on page 3-135 for important information.

Single Process Counters Text Pane

The following illustrates the Single Process Counters Text pane:



The screenshot shows a window titled 'lego Single Process Counters: 20969: ntuneserv'. It contains a table with four columns: 'Count/Sec', 'Total', and 'Since Last Update'. The rows list various performance metrics such as CPU Cycles, Instructions, Cache References, Branch Instructions, Bus Cycles, Context Switches, Page Faults, and L1D/L1I Cache Reads. The 'Misses' value for Cache References is highlighted with a dotted border.

	Count/Sec	Total	Since Last Update
CPU Cycles	50,551,471	691,209,899	177,661,680
Instructions	35,982,771	493,415,726	126,460,405
Cache References	108,847	1,484,202	382,540
Hits	103,235	1,429,990	362,817
Misses	5,612	54,212	19,723
Branch Instructions	7,830,413	107,433,934	27,519,761
Hits	7,678,230	105,325,180	26,984,917
Misses	152,183	2,108,754	534,844
Bus Cycles	off	off	off
Context Switches	6	84	21
Page Faults	140	1,882	493
Minor	140	1,882	493
Major	0	0	0
L1D Cache Reads	off	off	off
Hits	off	off	off
Misses	off	off	off
L1I Cache Reads	off	off	off
Hits	off	off	off
Misses	off	off	off

Figure 3-75. Single Process Counters Text Pane

For each piece of information, three values are displayed: the count per second for the given interval, the total since either the creation of the panel or the last clearing of the numbers, and the difference since the last update (e.g. typically the number in the last update interval). There are two special values:

- n/a, indicating that the counter could not be installed by the kernel, usually because the maximum number of simultaneous counters supported by the CPU was exceeded
- off, indicating that the counter was not configured to be active

The information displayed in this area includes:

CPU Cycles

The number of CPU clock cycles.

Instructions

The number of instructions executed.

Cache References

The number of times a memory reference attempted to access any memory cache, regardless of whether or not the desired memory item was present in the cache.

Cache References: Hits

The number of times a memory reference attempted to access any memory cache and the desired memory item was in the cache.

Cache References: Misses

The number of times a memory reference attempted to access any memory cache and the desired memory item was not in the cache.

Branch Instructions

The number of branch instructions executed.

Branch Instructions: Hits

The number of branch instructions executed for which branch prediction guessed correctly whether or not the branch would be taken.

Branch Instructions: Misses

The number of branch instructions executed for which branch prediction guessed incorrectly whether or not the branch would be taken.

Bus Cycles

The number of bus clock cycles.

Context Switches

The number of times the process was context switched out for another process.

Page Faults

The number of times a memory access requested a page not currently hardware mapped for this process.

Page Faults: Minor

The number of times a memory access requested a page not currently hardware mapped for this process, but where the page was loaded in memory already for some other reason (e.g. for another process).

Page Faults: Major

The number of times a memory access requested a page not currently hardware mapped for this process, and where the page had to be loaded from disk.

L1D Cache Reads

The number of times a memory read attempted to read from the level 1 data cache, regardless of whether or not the desired memory item was present in the cache.

L1D Cache Reads: Hits

The number of times a memory read attempted to read from the level 1 data cache and the desired memory item was in the cache.

L1D Cache Reads: Misses

The number of times a memory read attempted to read from the level 1 data cache and the desired memory item was not in the cache.

L1I Cache Reads

The number of times a memory read attempted to read from the level 1 instruction cache, regardless of whether or not the desired memory item was present in the cache.

L1I Cache Reads: Hits

The number of times a memory read attempted to read from the level 1 instruction cache and the desired memory item was in the cache.

L1I Cache Reads: Misses

The number of times a memory read attempted to read from the level 1 instruction cache and the desired memory item was not in the cache.

Single Process Counters Bar Graph Pane

The following illustrates the Single Process Counters Bar Graph pane:

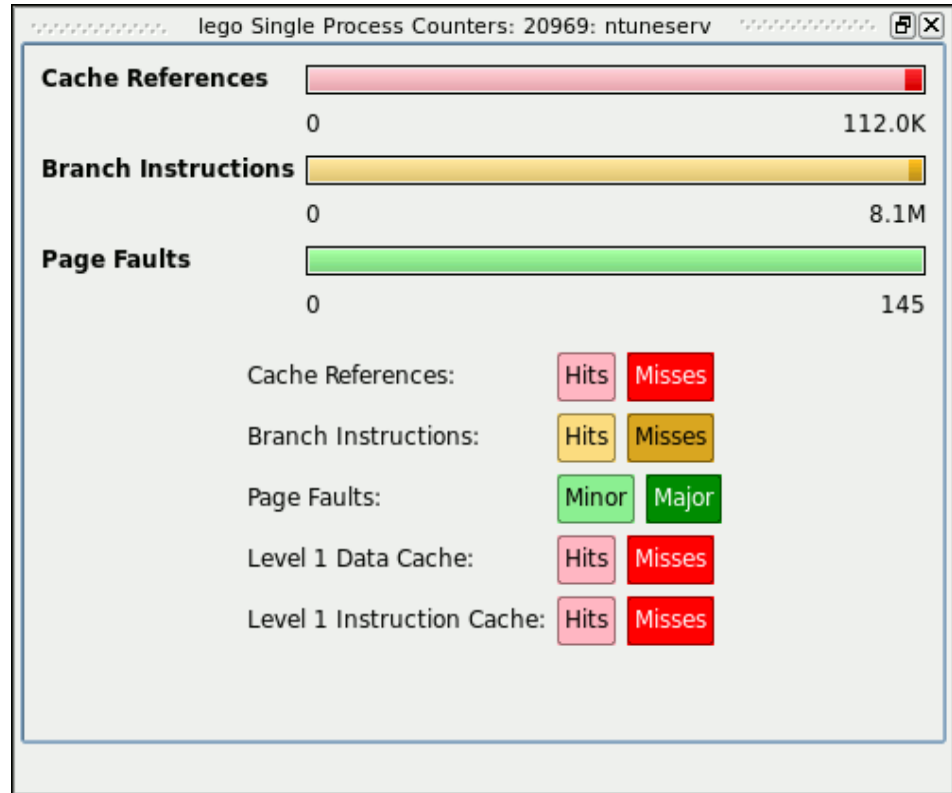


Figure 3-76. Single Process Counters Bar Graph Pane

Certain counters have values which are related to each other. Those are displayed in bar graph fashion. The bars are:

- Cache References, divided into Hits and Misses.
- Branch Instructions, divided into Hits and Misses.
- Page Faults, divided into Minor and Major.
- L1D Cache Reads, divided into Hits and Misses.
- L1I Cache Reads, divided into Hits and Misses.

The numbers displayed are all in count per second.

If any of the values necessary to display the bar graph are n/a or off, then the entire bar is greyed out.

Single Process Counters Line Graph Pane

The Single Process Counters Line Graph pane provides individual line graphs for each counter. The following illustrates the Single Process Counters Line Graph pane:

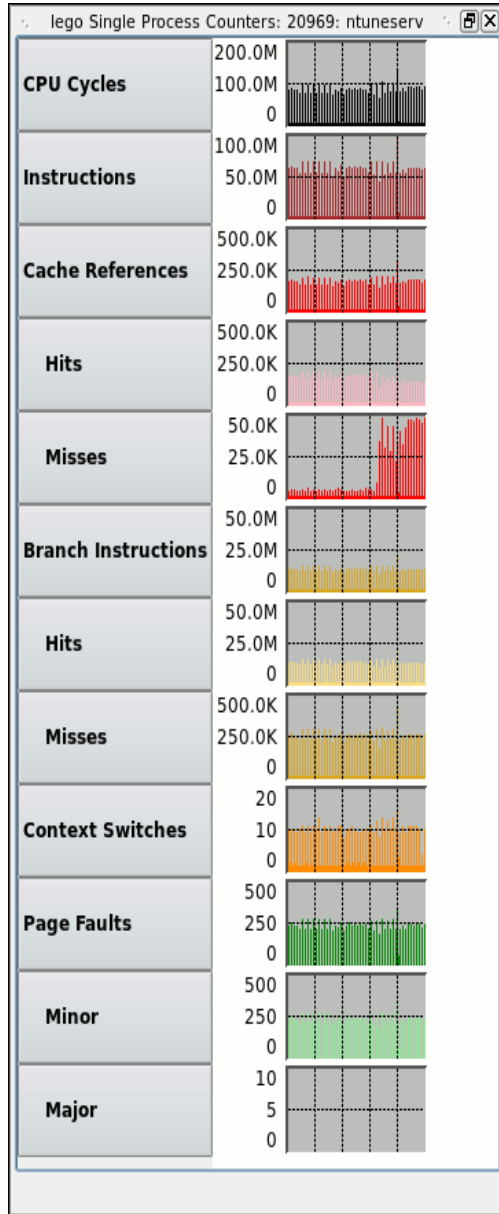


Figure 3-77. Single Process Counters Line Graph Pane

The counter per second of each counter is displayed vertically in each graph.

Single Process Counters Context Menu

While positioned in a single process counters panel, right-clicking displays a context menu, which contains the following content:

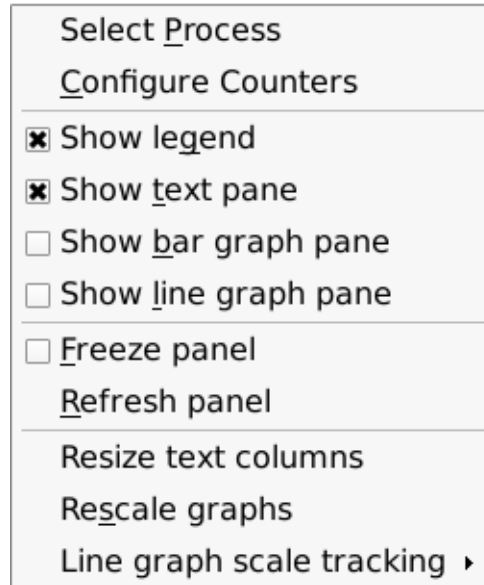


Figure 3-78. Single Process Counters Context Menu

The following paragraphs describe the menu items in detail:

Select Process

Mnemonic: P

The menu item opens a **Select Process** dialog to specify the process to be watched. See “Single Process Panel” on page 3-6 for details.

Configure Counters

Mnemonic: C

The menu item opens a **Configure Counters** dialog to configure which counters should be watched. See “Configure Counters Dialog” on page 3-134 for details.

Show legend

Mnemonic: G

This menu item toggles the visibility of the legend defining the colors used in the panel.

Show/Hide text pane

Mnemonic: T

This menu item toggles the visibility of the Text pane within the panel.

Show/Hide bar graph pane

Mnemonic: B

This menu item toggles the visibility of the Bar graph pane within the panel.

Show/Hide line graph pane

Mnemonic: L

This menu item toggles the visibility of the Line graph pane within the panel.

Freeze/Unfreeze panel

Mnemonic: F

This menu item toggles the **Freeze** setting for the panel. When frozen, data values are not refreshed automatically. This menu item overrides the **Freeze** setting for the window, but only applies to the particular panel.

Refresh panel

Mnemonic: R

This menu item causes all data within the panel to be refreshed once, regardless of the **Freeze** setting.

Resize text columns

This menu item causes NightTune to resize each text column such that it is wide enough to contain the widest value or text within that column.

Rescale graphs

Mnemonic: S

This menu item causes NightTune to rescale all graphs within the panel based on the current data for the panel.

Line graph scale tracking

This menu item displays a cascade menu with the following content:

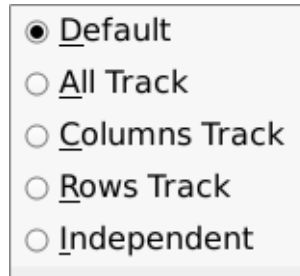


Figure 3-79. Line graph scale tracking cascade menu

The radio buttons in this menu determine how line graphs in the panel will rescale with respect to one another.

Default

The default tracking depends on the particular panel, but generally is designed for line graphs displaying the same types of information. For instance, in the **Disk Usage Panel**, the Usage graphs will track together, all the Ops/Sec graphs will track together, all the Sectors/Sec graphs will track together, and the two Average Time graphs will track together.

All Track

This setting causes all line graphs in the panel to rescale to a common maximum.

Columns Track

This setting causes all the rows in each column to rescale to a common maximum.

Rows Track

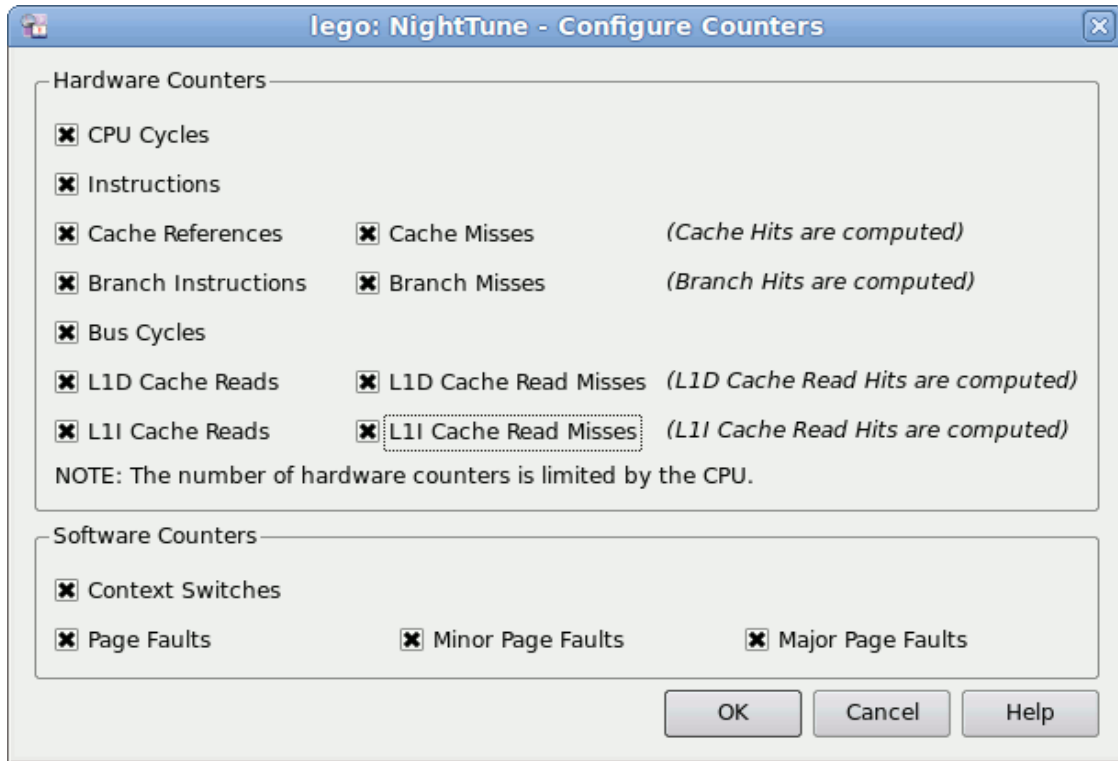
This setting causes all the columns in each row to rescale to a common maximum.

Independent

This setting causes each line graph in the panel to rescale independently of the others.

Configure Counters Dialog

The Configure Counters dialog allows the user to specify which counters a Single Process Counters panel should watch. The following illustrates the dialog:



The counters are divided into hardware counters and software counters.

Hardware counters require both support by the CPU and the linux kernel, and there are a limited number of simultaneous counters supported for each CPU. If the number of counters supported by the CPU is exceeded, arbitrarily-chosen counters will be disabled by the kernel and their values will be displayed as *n/a*. Software counters do not rely on support from the CPU because they count linux kernel events. They do still rely on Linux kernel support.

Each counter may be enabled or disabled by checking or unchecking its checkbox. Note that some counters displayed by the Single Process Counters panel are computed from two other counters. As such, they have no checkboxes, but rely on two other counters being enabled. For instance, cache hits are computed from cache references and cache misses.

Required Privileges for the Single Process Activity/Counters Panels

Both the **Single Process Activity** and **Single Process Counters** panels utilize process-specific performance counters. These counters are only supported on modern Intel-based systems and require a recent RedHawk kernel from Concurrent Real-Time

If you intend to use these panels to monitor other users' processes, you must either be the `root` user or have the `CAP_SYS_PTRACE` capability. See "Capabilities" on page 1-3 for more information on capabilities and how to set them up for your user account.

Additionally, for the **Single Process Activity** panel to show you symbolic function names associated with the process's activity, you must have read access to the program file and shared libraries associated with the process.

NightTune Logging

NightTune can log any changes you make to process scheduling attributes, CPU shielding, or interrupt affinity. Additionally, all the activities that NightTune monitors can be logged.

Two logging output formats are supported: XML and NightTrace data format.

Logging occurs on the target system(s) but is controlled from the host system (host and target can be the same or different systems). Logging occurs at the intervals specified in the **Preferences Dialog** for the three classes of information that can be logged: CPU shielding status, process attributes, and system metrics. Additionally, changes made to process attributes, interrupt affinity, or CPU shielding through NightTune are immediately logged (on option).

Logging is configured through the **Logging dialog**, which can be launched from the **File** menu or using the **Ctrl+L** shortcut key sequence.

Logging Dialog

Logging attributes are part of a NightTune configuration. Changes you make in this dialog are not saved for subsequent use in another NightTune invocation unless you also save the NightTune configuration (e.g. **File -> Save Configuration**).

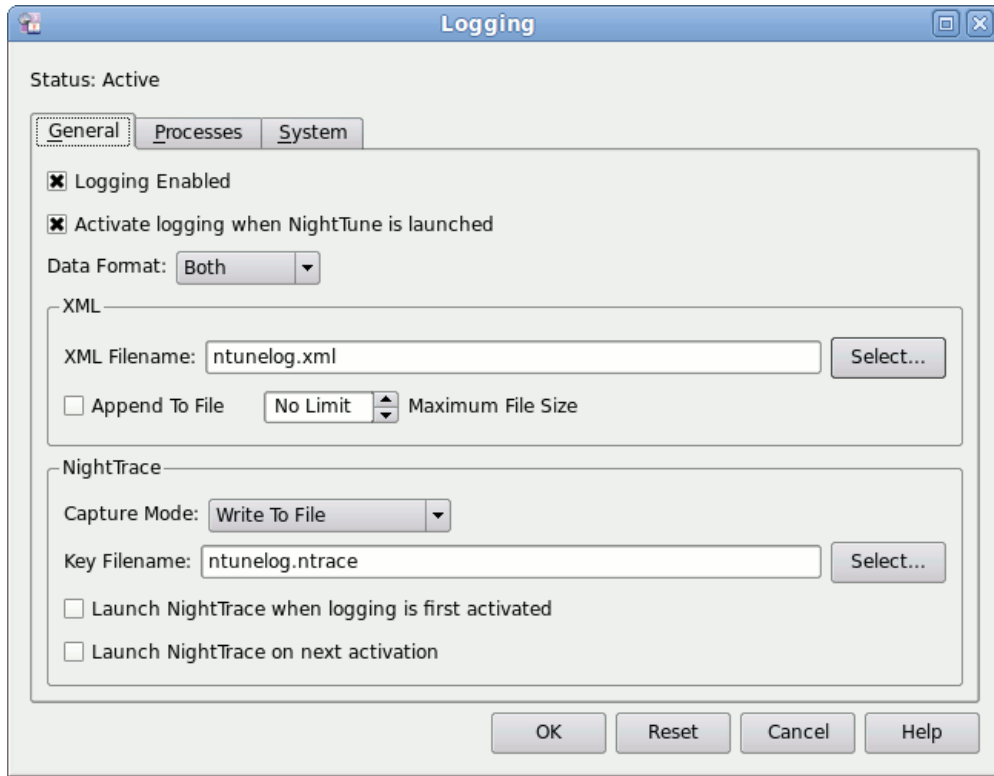


Figure 4-1. Logging Dialog - General Tab

The Logging dialog is organized with three tabs: General, Processes, and System.

General Tab

This tab controls the basic logging attributes, including the output format and whether logging should be enabled or disabled.

Enabled

This checkbox controls whether logging is active in the current session. When checked, logging is currently active, or if it was not checked when the dialog was launched, logging will be activated when the OK button is pressed.

Conversely, if you clear the checkbox in the dialog, logging will stop when the OK button is pressed.

Activate logging when NightTune is launched

This checkbox controls whether logging will be automatically activated when NightTune launches (assuming that you save this configuration for subsequent use).

When creating a logging configuration for use with the `--no-gui` option to `ntune` for batch-mode operation, you should make sure this checkbox is checked.

Data Format

You can select XML output, NightTrace output, or Both.

It is important to understand that logging occurs on the target system, not the host system (unless of course the host and target are the same).

XML

XML output is written to the log file as specified in the XML Filename field. Many network browsers support formatting XML files if the filename ends in the suffix `.xml`. Additionally, there are open-source, GPL XML parsers and APIs available (for example, TinyXML at <http://sourceforge.net/projects/tinyxml>). The Python scripting language, among others, also supports parsing XML files.

XML FileName

The filename can be specified as a relative file name or an absolute filename.

When logging data for multiple systems from a single NightTune session, you might want to specify an absolute pathname as opposed to a relative pathname. Relative pathnames will be relative to the user's home directory on remote systems. If you are running in an NFS-environment where your home directory is shared between multiple systems, you could inadvertently have multiple asynchronous writes to the same file which would corrupt data.

If you use the **Select...** button to browse for a filename, remember that the files shown to you are on the host system, but the actual file when logged will be on the target system -- i.e. the pathname you chose with the file selection dialog may be invalid on the target system.

Append To File

When checked, new logging sessions are appended to the specified file; otherwise, the file is truncated when a logging session begins.

Maximum File Size

This field allows you to set a size limit for the total accumulated XML output size, in megabytes.

The limit applies to the sum of all XML output files (since logging will occur to one file for each target connected to NightTune).

You can use the Up and Down arrows to the right of the field to increment and decrement the value. Specifying a value of zero means there is no limit and the XML files may grown huge.

When a non-zero limit is specified, NightTune will automatically stop XML logging when the limit is exceeded. In graphical mode, a pop-up dialog is shown to indicate this event. In batch mode, a message is printed to `stderr`.

If NightTrace output is also active and the XML size limit is reached, NightTune logging continues in NightTrace data format. You can control NightTrace output limits using `ntraceud` or NightTrace (`ntrace`).

In graphical mode, Logging Status Bar uses yellow and red background colors when the current XML size approaches 50% and 90% of the size limit, respectively.

NightTrace

When NightTrace data format is specified, NightTune logs data using the NightTrace Logging API. To collect the data, a NightTrace daemon must be run by the user. You can do this by invoking `ntraceud` with the filename as specified in the **Key Filename** field (as described below), or you can run `ntrace` and control the daemon graphically.

Remember that logging occurs on the target system, so `ntraceud` must be run on each target system that is connected, or, `ntrace` must launch a daemon from the host system that controls remote NightTrace daemons. Data can be written to the specified **Key Filename**, or it can be streamed directly into a graphical `ntrace` session.

NightTune automatically creates a NightTrace graphical session file which contains the daemon definitions for each target system that is involved in logging. It also contains the event format descriptions so that the logged data is automatically displayed in a meaningful way within NightTrace.

Capture Mode

The capture mode can be **Write To File** or **Stream To NightTrace**.

When **Write To File** is selected, the session file that is created includes daemon definitions that write to the specified **Key Filename**.

When **Stream To NightTrace** is selected, the session file that is created includes daemon definitions that stream data directly to the graphical `ntrace` session.

This setting really only affects the session file that is created. You can change your mind while NightTune is actively logging and use either mode by simply using `ntraceud` (**Write To File**) or a daemon definition in `ntrace` which selects the capture mode.

Key Filename

This filename serves three purposes.

1. It identifies instances of applications that are logging NightTrace data (in this case, NightTune server processes) and their associated daemon.

2. It identifies the file which will contain the logged data if the **Capture Mode** is **Write To File**.
3. It determines the name of the NightTrace session file which is created by NightTune. The name of the session file is the name of the **Key Filename** with **.session** appended to it. Unlike the **Key Filename** files themselves, which are relative to the target system(s), the session file will be created once (even if multiple targets are in use) on the host system.

A relative or absolute pathname may be specified. Relative pathnames are relative to the current working directory on the host system, but relative to the user's home directory for remote targets. A remote target is a system specified via the **--target** command line option or a system selected via the **Connect** dialog in the NightTune GUI.

This is best understood by example. See "Logging Examples" on page 4-12.

Launch NightTrace when logging is first activated

When checked, **ntrace** will be launched on the host system the first time that logging is initiated in this session (or when NightTune is restarted subsequently with this option saved to the configuration). This only occurs once per NightTune session; use the **Launch NightTrace on next activation** checkbox described below to (re)start NightTrace in the current session.

When launched, the NightTrace session file is automatically passed to **ntrace**, which includes the daemon definition(s) and the event name and format descriptions.

Launch NightTrace on next activation

When checked, **ntrace** will be launched on the host system the next time that logging is initiated, or if logging is already active, the next time **OK** is pressed. This setting only applies to the current session and the checkbox is automatically cleared as soon as **ntrace** is launched. It is not saved as part of the configuration.

When launched, the NightTrace session file is automatically passed to **ntrace**, which includes the daemon definition(s) and the event name and format descriptions.

Processes Tab

This tab controls whether processes attributes are to be logged, and if so, which attributes and for which processes.

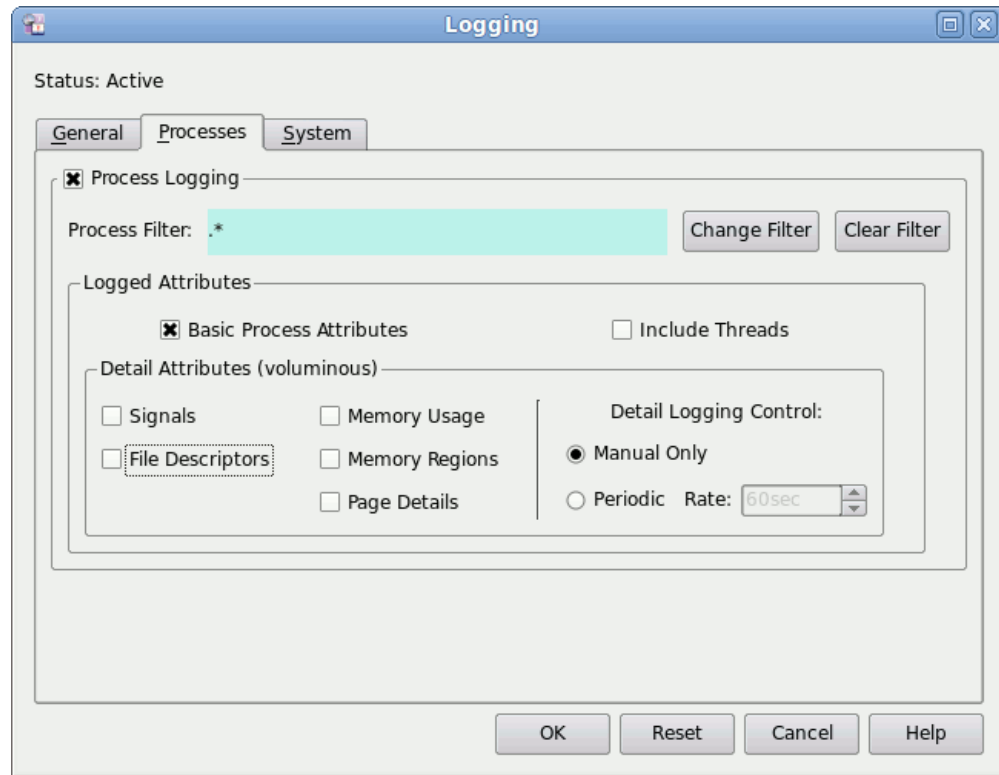


Figure 4-2. Logging Dialog - Processes Tab

Process Logging

This checkbox controls whether process logging occurs.

Process Filter

This text field provides a description of the current process filter. By default, the filter includes all processes (*. * is regular expression syntax that matches anything).

To change the filter, press the **Change Filter** button which launches the standard **Filter Processes** dialog with some slight modifications. See “Filter Processes Dialog” on page 3-91 for more information on filtering.

The standard **Filter Processes** dialog includes some additional checkboxes which are not available in this instance of the dialog, because they are either too expensive to implement (high-overhead) at the time of logging or are redundant with others settings in the **Processes** tab.

Logging Attributes

Basic Process Attributes

This checkbox controls whether the basic process attributes are logging for each process that matches the **Process Filter**. These include the attributes that can be viewed directly in the Process List Panel display.

They include PID, parent PID, UID, user name, priority, scheduling class, CPU affinity, CPU utilization, and memory utilization among others.

Include Threads

This checkbox controls whether information is logged for individual threads of processes that match the **Process Filter**. This checkbox is recommended; otherwise, multi-threaded process logging descriptions will only include attributes which are common to the entire process.

Detail Attributes

These attributes require significant processing to calculate and their size can be very large.

As such, these attributes require that you have a non-null process filter defined -- the intention being to select a single or a small set of processes.

Additionally, these attribute are not sampled at the same rate that the basic process attributes are sampled. By default, they are not automatically sampled at all -- you must manually request a sample by pressing the **Sample** button in the **Logging Status Bar**. Alternatively, you can request periodic sampling by setting the **Detail Logging Control** as described below.

File Descriptors

This checkbox controls whether descriptions of process file descriptors are logged. This information can be voluminous and incurs significant overhead to calculate. Use of this checkbox requires a non-null process filter.

This data is similar in nature to that shown in the **File Descriptors Tab** of the **Process Details Window**.

Signals

This checkbox controls whether descriptions of all signals are logged. Use of this checkbox requires a non-null process filter.

This data is similar in nature to that shown in **Signals Tab** of the **Process Details Window**.

Memory Usage

This checkbox controls whether descriptions of memory usage are logged. Use of this checkbox requires a non-null process filter.

This data is similar in nature to that shown in the **Memory Usage Tab** of the **Process Details Window**.

Memory Regions

This checkbox controls whether descriptions of all memory regions are logged. In this context, a region is a range of memory with similar mapping attributes; e.g. the text (instruction) section of the program or the heap area. Use of this checkbox requires a non-null process filter.

This data is similar in nature to that shown in the **Memory Tab** of the **Process Details Window**.

Page Details

This checkbox controls whether descriptions of all memory segments are logged. In this context, a segment is a range of memory with identical attributes. This information is voluminous and incurs significant overhead to calculate. Use of this checkbox requires a non-null process filter.

This data is similar in nature to that shown in the **Memory Tab** of the **Process Details Window**. Depending on the target system's capabilities, it may include residency, locked, and NUMA node information.

Detail Logging Control

This area determines the frequency at which detailed process attributes are logged. By default, the sampling control is set to **Manual** -- thus no logging of process details occurs until you press the **Sample** button in the **Logging Status Bar**.

Alternatively, you can request periodic sampling and specify the periodic rate.

WARNING

Sampling of process details requires significant overhead and the size of the information is voluminous. Specifying a frequent rate combined with a process filter that selects more than a few processes may overwhelm NightTune or impede system throughput.

IMPORTANT

If you are creating a logging configuration for subsequent use in batch-mode (without graphical interactive control) and the control is set to **Manual**, a single sample of process details will occur during the NightTune session -- immediately upon launching NightTune. No other samples of process details will be taken during that session.

System Tab

The **System** tab controls whether system metrics are logged and whether changes made to the system or processes from within NightTune are logged.

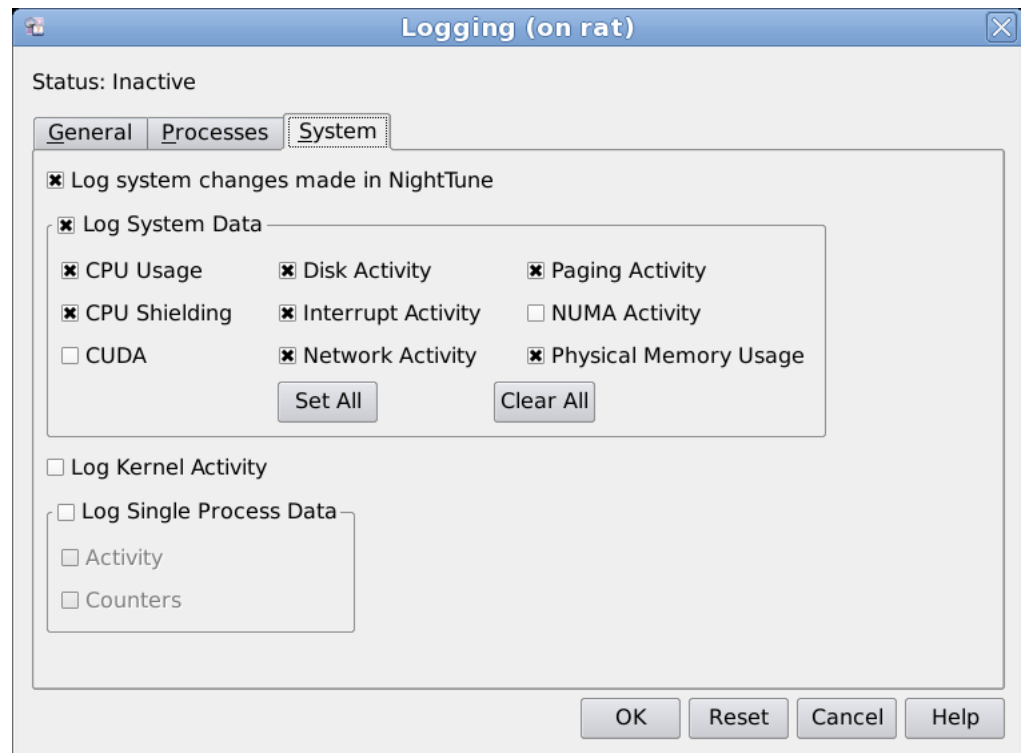


Figure 4-3. Logging Dialog - System Tab

Log system changes made in NightTune

This checkbox controls whether tuning changes made in the current NightTune session are logged. These includes changes to CPU shielding, interrupt CPU affinity, and process scheduling.

Log System Data

This checkbox controls whether system metrics are logged.

CPU Usage

CPU Usage is logged when this box is checked. This information is similar in nature to that shown in the CPU Usage Text Pane.

CPU Shielding

CPU shielding status is logged when this box is checked. This information is similar in content to that shown graphically in the CPU Shielding and Binding Panel.

CUDA

CUDA information is logged when this box is checked. This information is similar in nature to that shown in the **CUDA Text Pane**.

Disk Activity

Disk activity is logged when this box is checked. This information is similar in nature to that shown in the **Disk Activity Text Pane**.

Interrupt Activity

Interrupt activity is logged when this box is checked. This information is similar in nature to that shown in the **Interrupt (Detail) Activity Text Pane**.

Network Activity

Network activity is logged when this box is checked. This information is similar in nature to that shown in the **Network Activity Text Pane**.

Paging Activity

Paging and swap activities are logged when this box is checked. This information is similar in nature to that shown in the **Memory Activity Text Pane** and the **Swap Text Pane**.

NUMA Activity

NUMA pool usage is logged when this box is checked. This information is similar in nature to that shown in the **NUMA Activity Text Pane**.

Physical Memory Usage

Physical memory usage is logged when this box is checked. This information is similar in nature to that shown in the **Physical Memory Text Pane**.

Log Kernel Activity

This checkbox controls whether kernel activity is logged.

Log Single Process Data

This checkbox controls whether single process data is logged. The particular processes logged are determined by the single process panels.

Activity

This checkbox controls whether single process activity data is logged. The particular processes logged are determined by the single process activity panels. This information mirrors the **Single Process Activity Panel**.

Counters

This checkbox controls whether single process counters data is logged. The particular processes logged are determined by the single process counters panels. This information mirrors the Single Process Counters Panel.

Logging Status Bar

When logging is active, the Logging Status Bar is shown at the bottom right section of the main window.

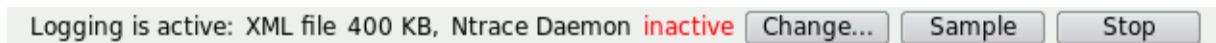


Figure 4-4. Logging Status Bar

The status bar will not appear until logging is initially activated -- either by using the Logging Dialog or loading a configuration file which specifies active logging.

The status bar includes status on the output methods in use for logging, along with a button which launches the Logging Dialog, a button to immediately take a Sample and log data, and a button which starts or stops logging.

For XML output, the combined size of all XML output files involved in the current logging session is shown. If an XML file size maximum limit has been specified in the Logging Dialog, the background of the size shown changes to yellow when the total size reaches 50% of the maximum limit, and subsequently changes to red when 90% of the limit is reached. When the full limit is reached or exceeded, NightTune will disable XML logging and indicate this with a pop-up dialog. If NightTrace logging was in effect, it continues to be; otherwise, all logging is therefore disabled.

For NightTrace output, a notation is shown which indicates whether a daemon (or daemons in the case of multiple targets) is active. You can launch a daemon in several ways:

- Run **ntraceud** outside of NightTune on each target system involved, and specify the name of the **Key Filename** specified in the Logging Dialog.
- Invoke NightTrace from the Tools menu and create daemon definitions using the Import option of the Daemons menu in NightTrace.
- Press the Change... button to launch the Logging Dialog and check the Launch NightTrace when logging is first activated checkbox and then press OK. NightTrace will be launched with the session file created by NightTune and you can launch the pre-defined daemons using the graphical user interface.

The latter technique is recommended.

Change...

Pressing this button launches the **Logging Dialog**, which allows you to disable logging or change any of the logging attributes.

Sample

Pressing this button causes NightTune to immediately sample all system data and log the desired data (as directed by settings in the **Logging Dialog**). This control is independent of sampling that occurs periodically.

As discussed in a section above, by default, process detail attributes are never logged except when you press the **Sample** button. See “Detail Logging Control” on page 4-8 for more information.

Start/Stop

Pressing this button toggles the logging state by either starting or stopping logging.

Logging Examples

This section demonstrates several logging scenarios. Logging is controlled by the settings made in the **Logging Dialog** which can be made persistent by saving the configuration for use with subsequent NightTune sessions.

The first step will be to use the **Logging Dialog** to select logging attributes and to save that configuration which we will use in the following examples.

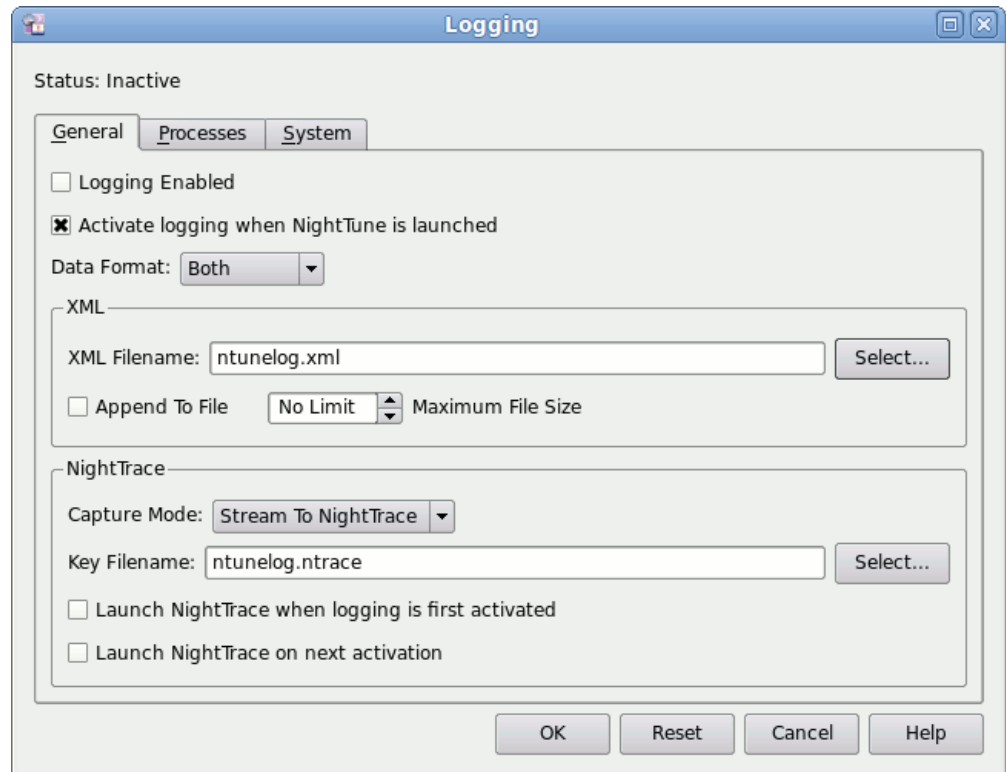
NOTE

Some of the examples in this chapter make use of NightTrace. These examples assume you have a working knowledge of NightTrace. If you aren't yet comfortable with NightTrace, you may want to first go through the NightTrace section of the NightStar Tutorial. The tutorial can be found online from the **Help** menu of any NightStar tool.

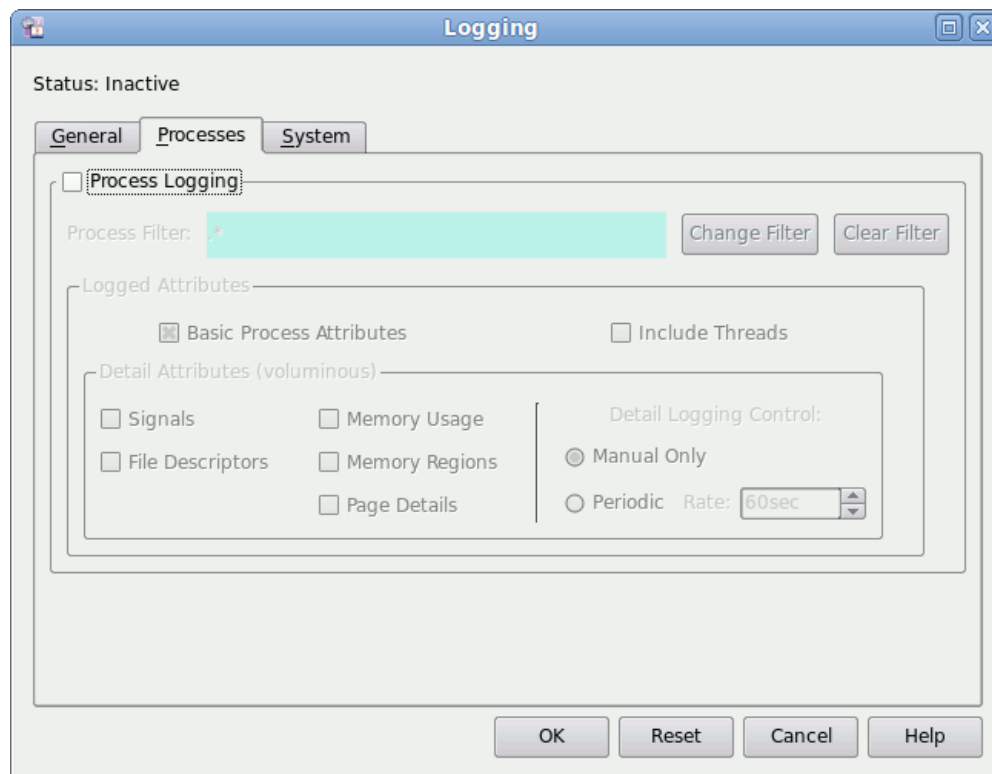
Configuring Logging

- Launch NightTune
- Launch the **Logging Dialog** using the **Logging** option of the **File** menu or the **Ctrl+L** keyboard shortcut.
- Clear the **Logging Enabled** checkbox if it is already checked
- Check the **Activate logging when NightTune is launched** checkbox

- Change the Data Format to Both.
The dialog should look similar to the following figure.

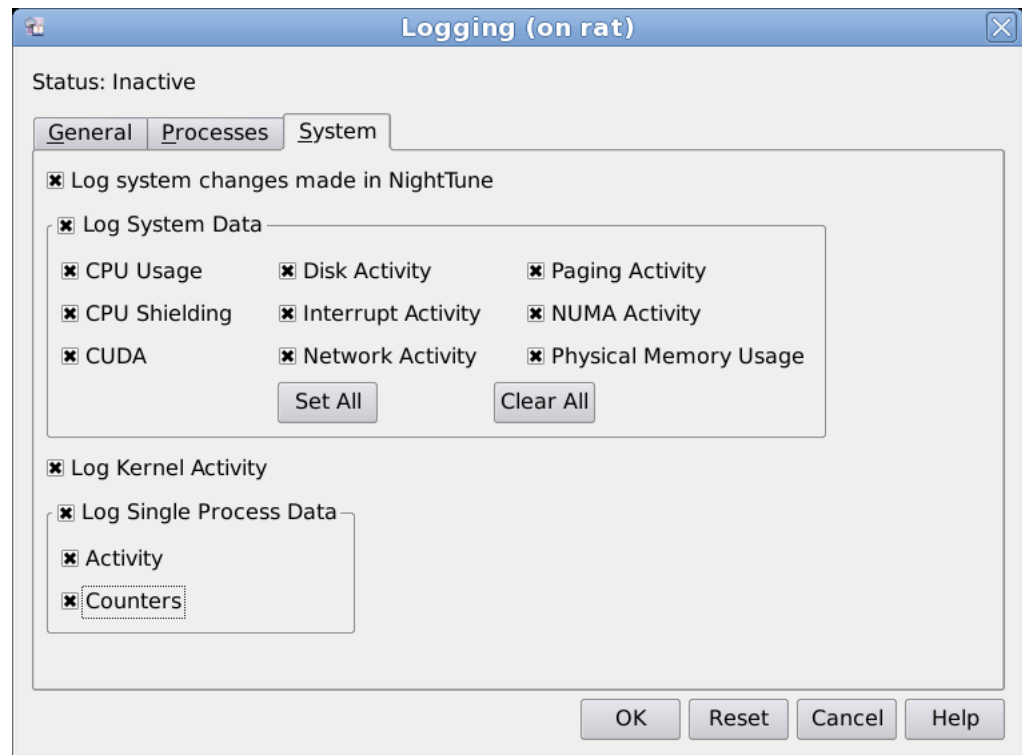


- On the **Processes** tab, clear the **Process Logging** checkbox. The dialog should look similar to the following figure.



- On the **System** tab, check the **Log System Data** checkbox.

- Press the Set All button to check all available attributes. The dialog should look similar to the following figure.



- Press the OK button to close the dialog.

At this point we have defined how we want to log data and what we want to log, but logging is still disabled.

- Select Save Config File As... from the File menu and save the file to `/tmp/myconfig`.
- Exit NightTune.

We have now created a NightTune configuration which logs system metrics in both XML and NightTrace data format. We will use this configuration in the following examples.

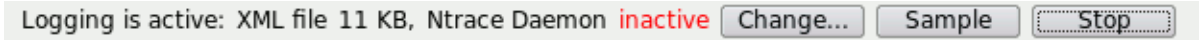
Using a Logging Configuration in the GUI

We will now utilize the configuration we created in the previous section.

- Invoke NightTune with the configuration file we saved in the previous section:

```
ntune --config=/tmp/myconfig
```

Since the configuration specified that logging should occur when NightTune is launched, it immediately becomes active and the Logging Status Bar at the bottom of the main window reports on its status.

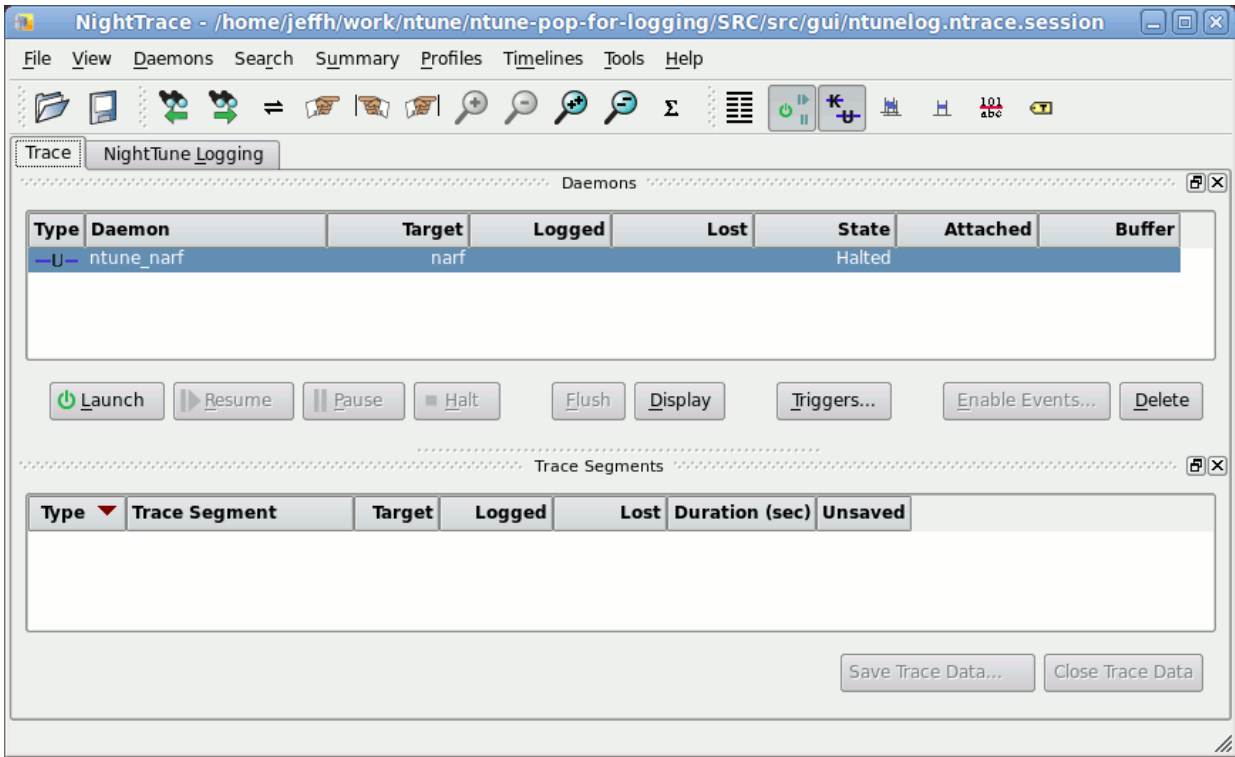


Note that logging is active and 11 KB of data has already been logged to the XML output file.

Also note that there is no active NightTrace daemon collecting NightTrace data logged from NightTune.

- Press the Change... button
- Check the Launch NightTrace on next activation checkbox
- Press the OK button

NightTrace is automatically launched and the NightTrace session file created by NightTune is specified as an argument.



On the Trace tab in NightTrace, a daemon definition is already provided which is configured to collect events logged from NightTune.

- Click on the daemon definition line in the Daemons panel to select it.
- Press the Launch button
- Press the Resume button

A daemon is now active and begins to collect data logged by NightTune.

In a few seconds, the Logging Status Bar in NightTune will update and show that the daemon is active.

Logging is active: XML file 75 KB, Ntrace Daemon active

- Press the Stop button in the Logging Status Bar in NightTune to disable logging.
- In NightTrace, press the Halt button in the Daemons panel.
- In NightTrace, click on the NightTune Logging tab.

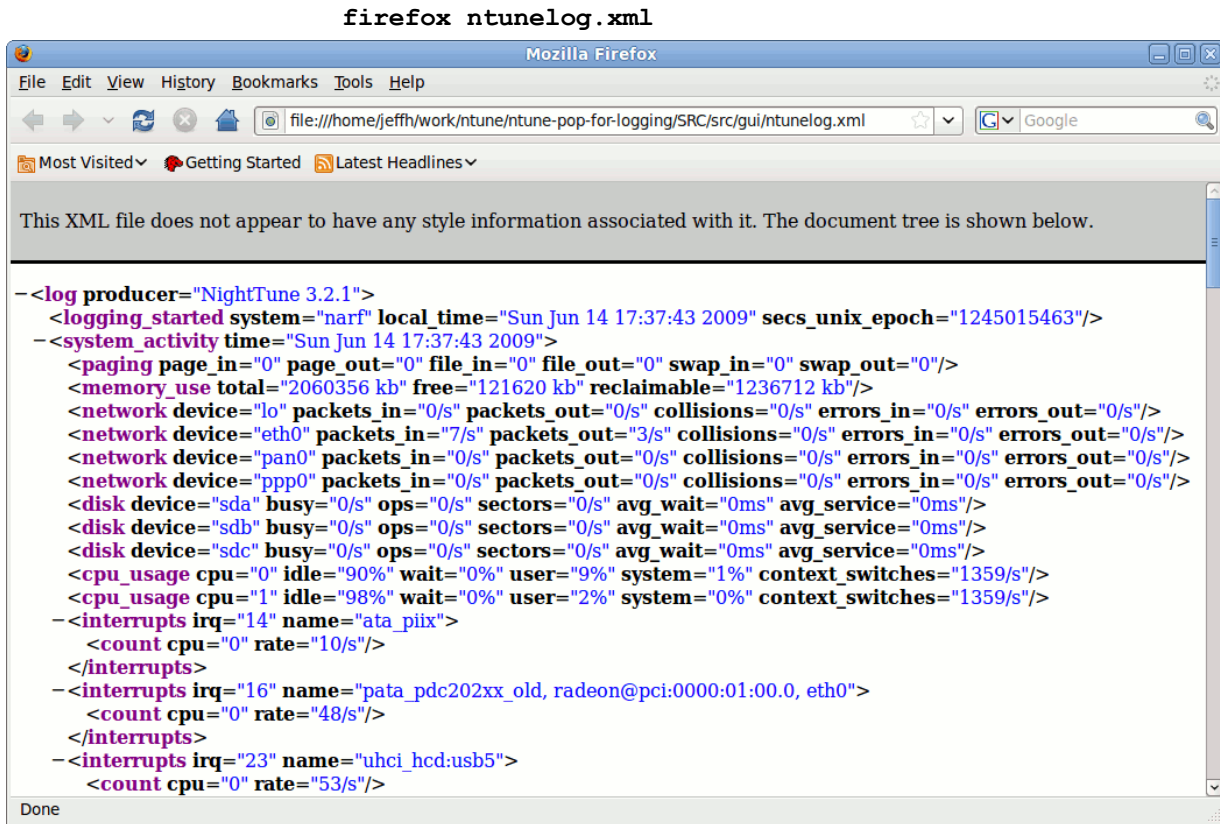
The screenshot shows the NightTrace application window. The title bar reads "NightTrace - /home/jeffh/work/ntune/ntune-pop-for-logging/SRC/src/gui/ntunelog.ntrace.session(Unsa)". The menu bar includes File, View, Daemons, Search, Summary, Profiles, Timelines, Tools, and Help. The toolbar contains various icons for file operations, search, and playback. The main window has tabs for "Trace", "NightTune Logging", and "ntune_narf". The "Events" panel is active, displaying a table of logged events.

Offset	Event	Time (sec)	Tag	Description
6	network_activity	0.385336753		Network Activity (/sec) for device ppp0: packages_in=0 packets_out=0 colli
7	disk_activity	0.385337159		Disk Activity for device sda: busy=0 operations=0/s sectors=/s avg_wait=0.
8	disk_activity	0.385337457		Disk Activity for device sdb: busy=0 operations=0/s sectors=/s avg_wait=0.
9	disk_activity	0.385337740		Disk Activity for device sdc: busy=0 operations=0/s sectors=/s avg_wait=0.
10	cpu_activity	0.385338224		CPU Activity (%) for CPU 0: idle=89 wait=0 user=10 system=1 context_swit
11	cpu_activity	0.385338502		CPU Activity (%) for CPU 1: idle=44 wait=0 user=54 system=2 context_swit
12	irq_activity	0.385339256		IRQ Activity: irq=ata_piix (14) on CPU 0 10/s
13	irq_activity	0.385339560		IRQ Activity: irq=pata_pdc202xx_old, radeon@pci:0000:01:00.0, eth0 (16).
14	irq_activity	0.385339941		IRQ Activity: irq=uhci_hcd:usb5 (23) on CPU 0 20/s
15	irq_activity	0.385340319		IRQ Activity: irq=LOC (-17) on CPU 0 167/s
16	irq_activity	0.385340549		IRQ Activity: irq=LOC (-17) on CPU 1 162/s
17	irq_activity	0.385340803		IRQ Activity: irq=RES (-18) on CPU 0 12/s
18	irq_activity	0.385341026		IRQ Activity: irq=RES (-18) on CPU 1 48/s
19	memory_activity	2.431019992		Memory Paging (kb/s): pages_in=0 pages_out=15 file_pages_in=0 file_pag.
20	memory_physical	2.431020490		Physical Memory Usage (kb): total=2060356 free=105072 reclaimable=122
21	network_activity	2.431021237		Network Activity (/sec) for device lo: packages_in=0 packets_out=0 colli.
22	network_activity	2.431021571		Network Activity (/sec) for device eth0: packages_in=8 packets_out=4 colli.
23	network_activity	2.431021882		Network Activity (/sec) for device pan0: packages_in=0 packets_out=0 colli.
24	network_activity	2.431022167		Network Activity (/sec) for device ppp0: packages_in=0 packets_out=0 colli.
25	disk_activity	2.431022571		Disk Activity for device sda: busy=0 operations=1/s sectors=/s avg_wait=0.
26	disk_activity	2.431022856		Disk Activity for device sdb: busy=0 operations=0/s sectors=/s avg_wait=0.
27	disk_activity	2.431023147		Disk Activity for device sdc: busy=0 operations=0/s sectors=/s avg_wait=0.
28	cpu_activity	2.431023596		CPU Activity (%) for CPU 0: idle=95 wait=0 user=5 system=0 context_swit.
29	cpu_activity	2.431023857		CPU Activity (%) for CPU 1: idle=92 wait=0 user=7 system=1 context_swit.

The Events panel in NightTrace is populated with individual events which represent logged data from NightTune.

Now let's take a look at the XML data.

- In a terminal session, invoke **firefox** (or some other file viewer) to see the contents of the XML data file:



Each logging session starts with a `log` element and includes a `logging_started` and `logging_stopped` element which shows the current time and the name of the system.

Each sample of system metrics are enclosed in a `system_activity` element which is stamped with the time the sample was taken.

- Exit NightTrace and NightTune.

Process Logging

We'll start with the logging configuration we saved in the section "Configuring Logging" on page 4-12. Once loaded, we'll modify it to log information about selected processes.

- Invoke NightTune with the configuration file we saved above:

```
ntune --config=/tmp/myconfig
```

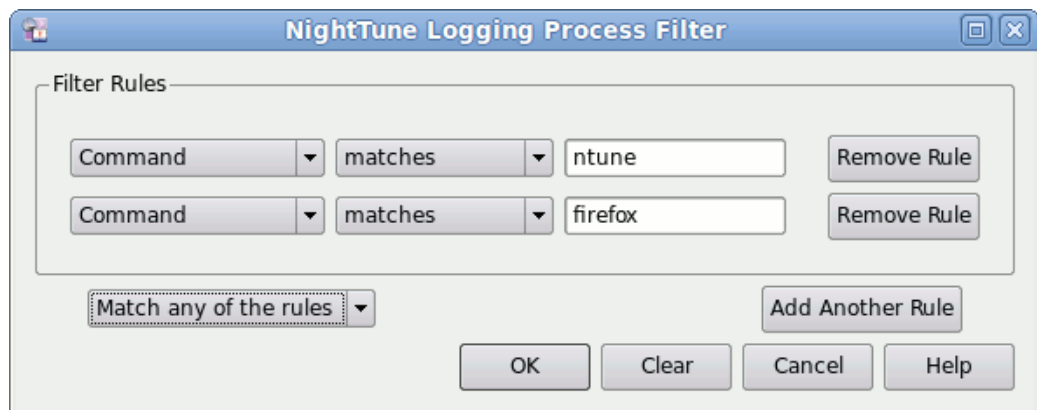
- Press the Change... button in the Logging Status Bar.
- On the General tab, change the Data Format Mode to XML.
- On the System tab, clear the checkbox that says Log System Data.
- On the Processes tab, check the Process Logging checkbox.

- Press the Change Filter button to launch the Filter Processes Dialog.

The NightTune Logging Process Filtering dialog appears.

- Create a rule which matches all commands named **ntune**
- Create another rule which matches all commands named **firefox**
- Ensure the option in the lower left portion of the dialog says Match Any of the Rules.

The dialog should look similar to the following figure.



- Press the OK button to close the Filter Process dialog.
- Press the OK button to close the Logging dialog.

NightTune is now logging data for any process named **ntune** or **firefox**.

- If you don't have **firefox** running, launch it and visit a couple of web sites.
- Exit NightTune.

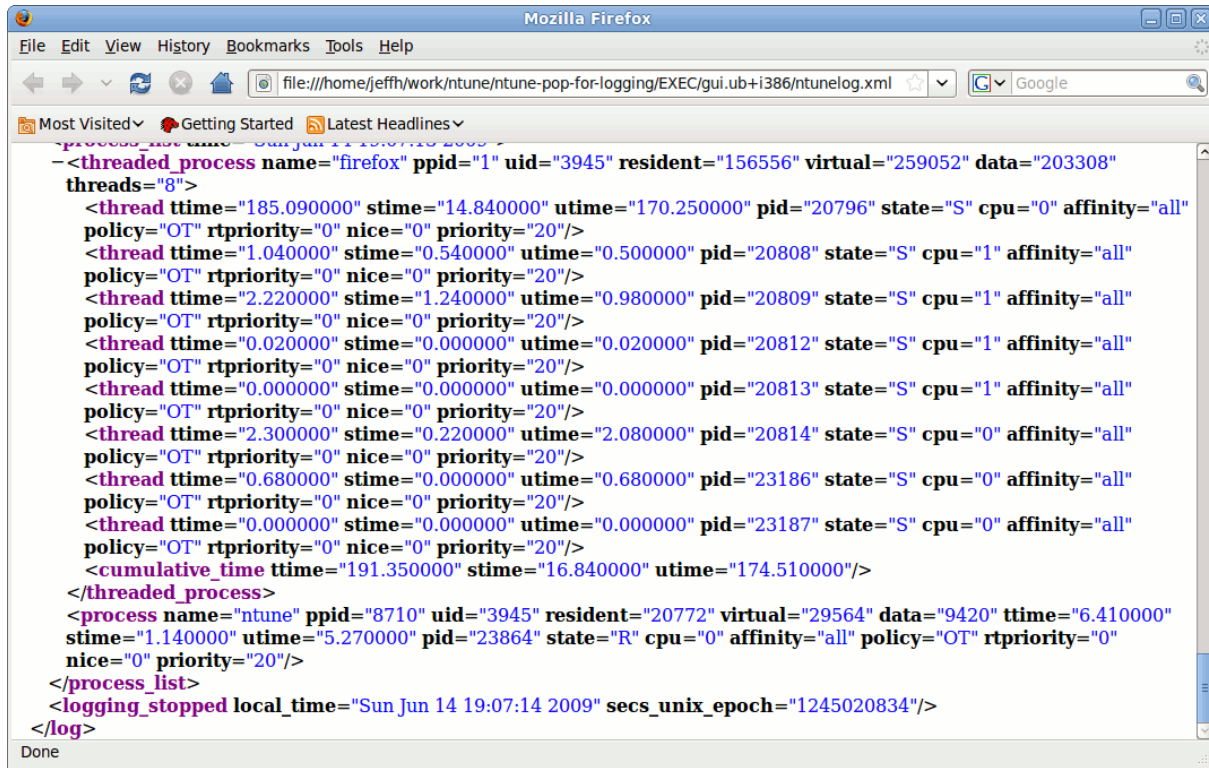
The XML file now contains process data.

- View it with **firefox** or another text viewer:

firefox ntunelog.xml

- Scroll to the end of the file in **firefox** (or position the file at the end in whatever viewer you are using).

The XML file will have output similar to the following figure.



For each process that matched the process filter, an element is logged. For multi-threaded processes, a `threaded_process` element is logged which includes attributes which apply to the process as a whole. Additionally, a `thread` element is logged for each thread, with attributes specific to that thread (because we checked the Log Threads checkbox in the Processes tab of the Logging Dialog). For single-threaded processes, a `process` element is logged contain the process's attributes.

Batch Mode Logging

In this example we'll invoke NightTune in batch mode to log data.

- Since our configuration file specifies that NightTune will log in NightTrace format, launch a NightTrace user daemon to collect the data:

```
ntraceud ntunelog.ntrace
```

That command causes a daemon to run in the background even though the command you invoked finishes quickly.

- Invoke NightTune using the saved configuration file from the initial section in this chapter, using the following arguments:

```
ntune --config=/tmp/myconfig --wait=10
```

The `--wait` option implies the `--no-gui` option and causes NightTune to log data silently without launching the graphical user interface. At the end of 10 seconds, `ntune` will terminate logging and exit.

- Now terminate the NightTrace daemon using the following command:

```
ntraceud --quit ntunelog.ntrace
```

- Now view the logged data using the following NightTrace command:

```
ntrace --listing ntunelog.ntrace.session \  
ntunelog.ntrace
```

The `ntunelog.ntrace.session` file was created by NightTune and has all the information in it that NightTrace needs to generate a descriptive listing of the contents.

The output from the `ntrace` command will include lines similar to the following, except that the unmodified listing is quite wide and has been edited in the figure below to remove some fields and provide line-wrapping to make it easier to read.

```
4: memory_activity time=2.927795641s
   Memory Paging (kb/s): pages_in=0 pages_out=0
                        file_pages_in=0 file_pages_out=0
                        swap_in=0 swap_out=0
5: memory_physical time=2.927796112s
   Physical Memory Usage (kb): total=2060356 free=209976
                                reclaimable=1174636;
   Swap Usage: total=6032368 free=5742568 used=0
6: network_activity time=2.927796799s
   Network Activity (/sec) for device eth0: packages_in=5
                                           packets_out=55 collision=0
                                           errors_in=0 errors_out=0
7: disk_activity    time=2.927798107s
   Disk Activity for device sda: busy=0 operations=2/s
                                sectors=/s avg_wait=0ms
                                avg_service=0ms
8: cpu_activity     time=2.927799115s
   CPU Activity (%) for CPU 0: idle=97 wait=0 user=2
                                system=1
                                context_switches=201/s
9: cpu_activity     time=2.927799388s
   CPU Activity (%) for CPU 1: idle=100 wait=0 user=0
                                system=0
                                context_switches=201/s
10: irq_activity    time=2.927800153s
    IRQ Activity: irq=ata_piix (14) on CPU 0 20/s
11: irq_activity    time=2.927800461s
    IRQ Activity: irq=pata_pdc202xx_old,
                 radeon@pci:0000:01:00.0, eth0 (16) on CPU 0 3/s
17: irq_activity    time=2.927801000s
    IRQ Activity: irq=LOC (-17) on CPU 0 31/s
...

```

Batch Mode Remote Logging

NightTune can log data on remote target systems either through the graphical user interface or in batch mode. In this example, we'll use batch mode and specify the remote target on the NightTune command line.

- Invoke NightTune and give it the name of a target system that has NightTune installed on it (and is accessible from the current system on the network). For example,

```
ntune --config=/tmp/myconfig --no-gui \  
--target=jojo@raptor x
```

The command above will initiate logging on the target system *raptor* as the user *jojo*. Before logging is initiated, **ntune** will need to authenticate *jojo* on the target system, so it may prompt you for *jojo*'s ssh passphrase or password. If you have already installed appropriate ssh keys in your current session, then no prompting will occur. (see **ssh-add(1)** and **ssh-agent(1)** for more information on installing ssh keys).

Once authenticated, a daemon is launched in the background and the **ntune** command you invoked (as shown above) will return; logging however is still active.

- After a period of at least 10 seconds, terminate the logging session with the following command:

```
ntune --quit x
```

The parameter *x* is the handle that identifies the daemon launched by the **ntune** command in the first step of this example.

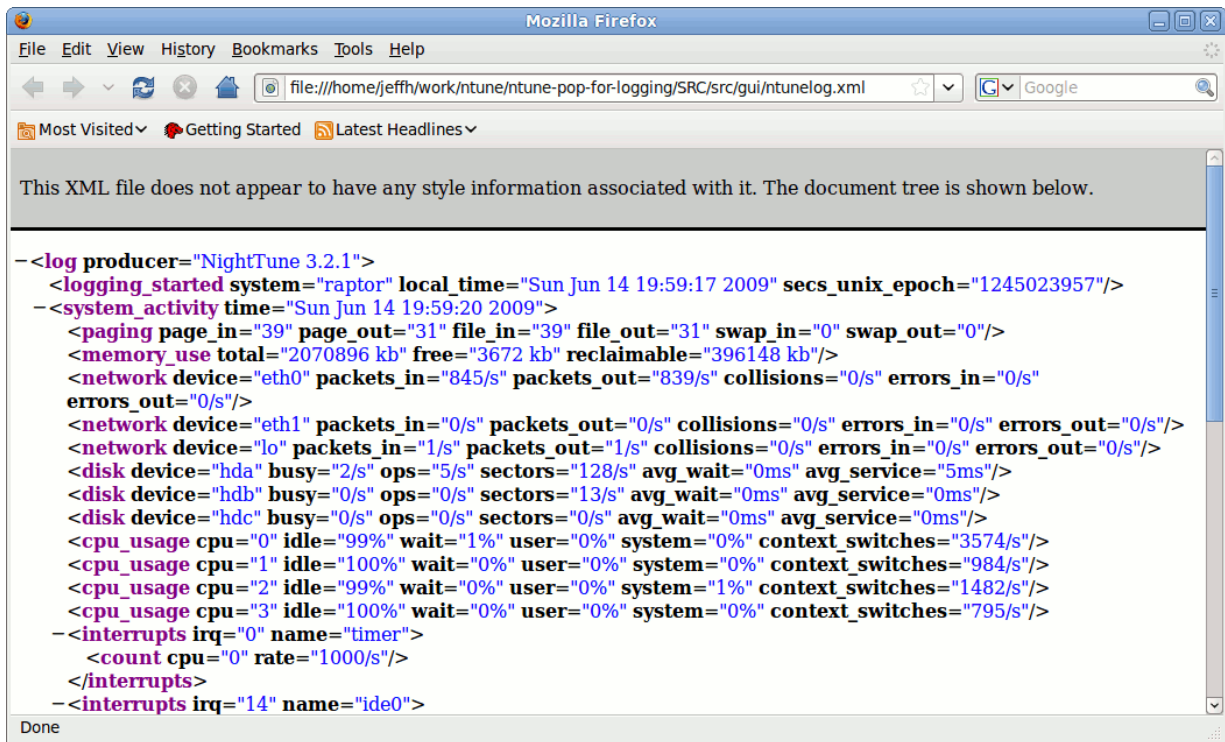
Now view the XML output file on the target system, which will be located in *jojo*'s home directory (since the XML output file was specified with a relative pathname in our configuration).

- Copy the file from the remote system back to the host system; for example:

```
ssh jojo@raptor:ntunelog.xml .
```


- Invoke **firefox** or another file viewer to see the data:

```
firefox ./ntunelog.xml
```



Note that multiple target options can be specified on the same ntune command line to log data on multiple remote systems.

This chapter guides you in operating NightTune to execute the following specific tasks:

- Monitoring User Processes (see “Monitoring User Processes” on page 5-2)
- Changing User Process Scheduling Attributes (see “Changing User Process Scheduling Attributes” on page 5-6)
- Shielding a CPU (see “Shielding a CPU” on page 5-10)
- Changing the CPU Affinity of an Interrupt (see “Changing the CPU Affinity of an Interrupt” on page 5-14)

Monitoring User Processes

If the Process List panel is not visible currently, select the Process List panel from the Monitor menu or click on the Process List tool icon.

The Process List panel will appear in the window as illustrated below:

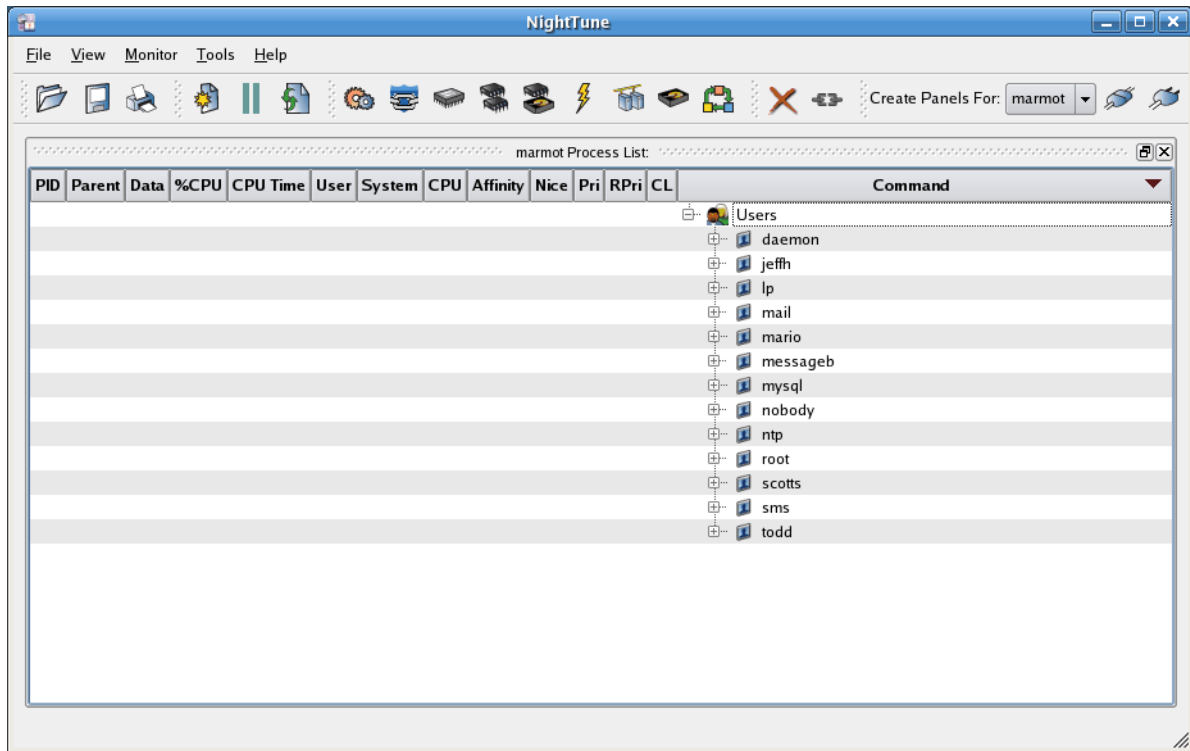


Figure 5-1. Monitoring User Processes

Selecting the User Process

The Process List panel is populated with user lines which, when expanded, describe the processes associated with each user.

Click on the plus sign associated with the user name of the process you wish to monitor. The list of processes is expanded beneath the user name as shown below. If you wish to see children of a given process (e.g. init), then click on its plus sign.

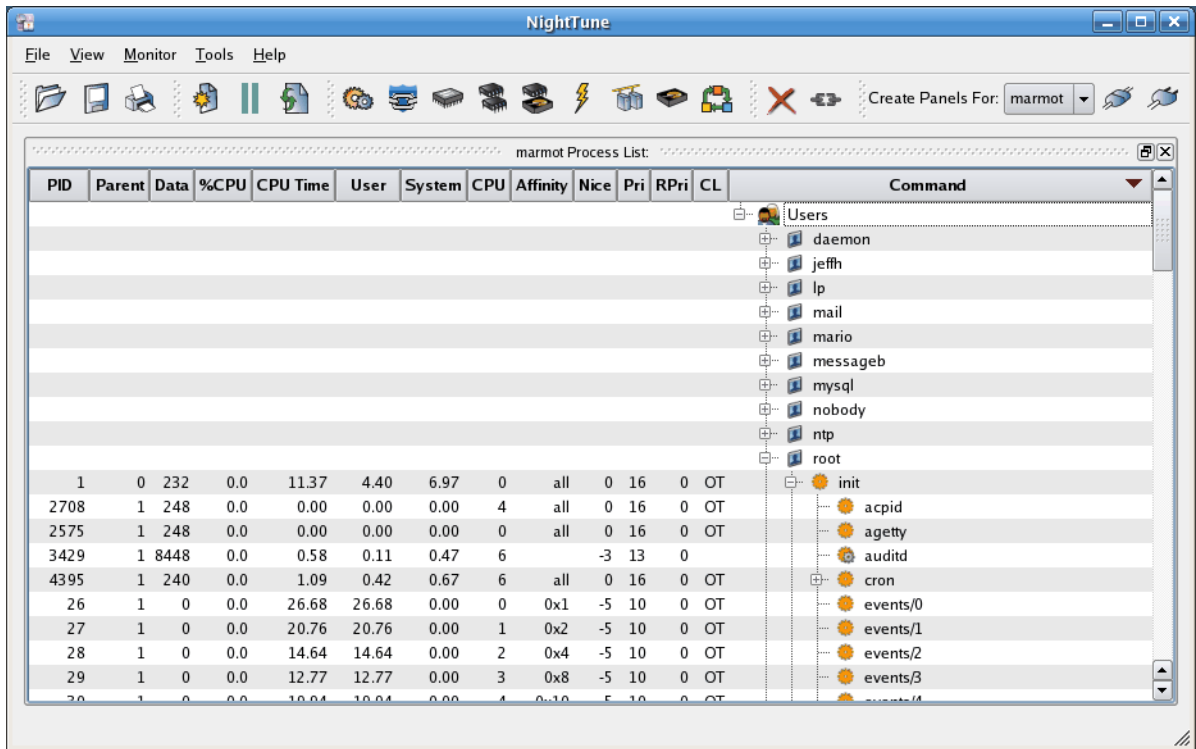


Figure 5-2. Selecting the User Process

Process attributes are displayed and updated periodically.

If a process is multi-threaded, the list of individual threads will be displayed when the parent process is selected and Show Threads is checked on the Process List context menu as shown below:

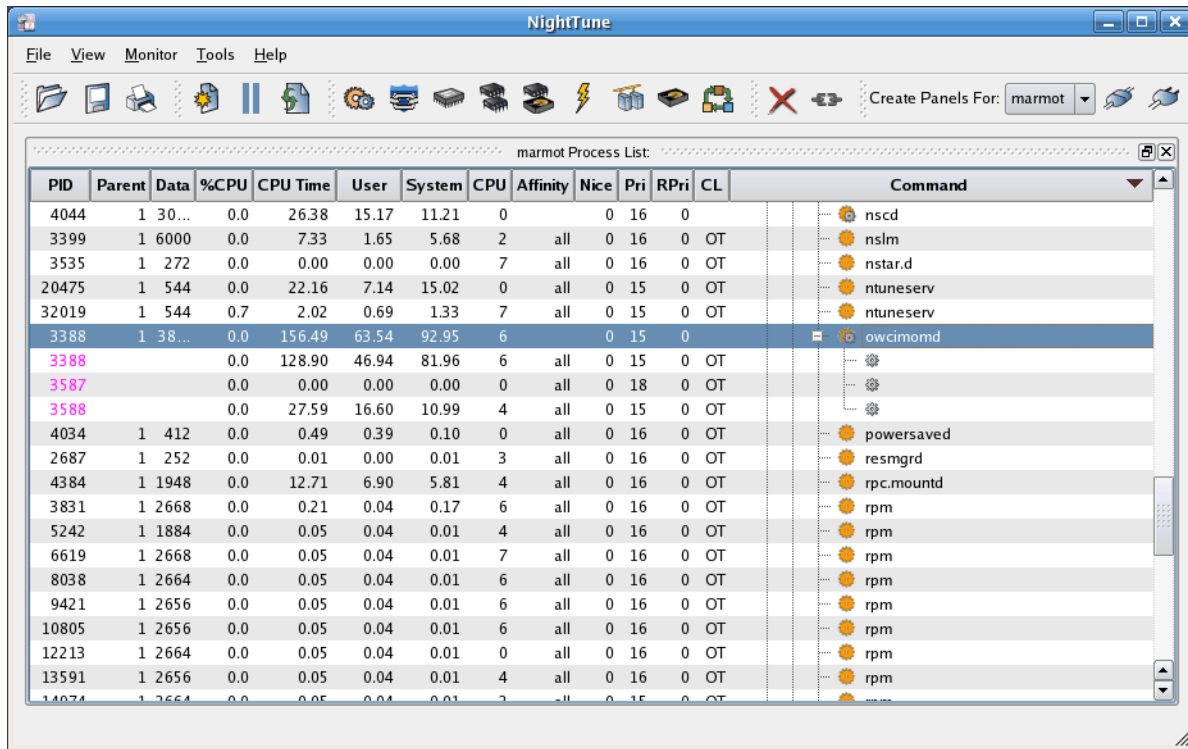


Figure 5-3. Monitoring Multi-threaded Processes

Customizing the Process Information

Select the Display Fields item from the Process List context menu to display the Process Fields menu.

NOTE

The Process Fields menu, like the other menus, can be “torn off” from the menu bar to reside in its own window separate from NightTune by clicking on the dashed line at the top of the menu. This is especially useful when making multiple changes to avoid having to select the menu after every choice.

Select or deselect process fields of interest to customize the display.

In the illustration below, the following process fields have been selected for display:

- Process ID
- Thread Information
- Virtual Memory Size
- Resident Memory Size
- CPU Time
- Most Recent CPU
- Affinity Mask
- Real-time Priority
- Scheduling Class

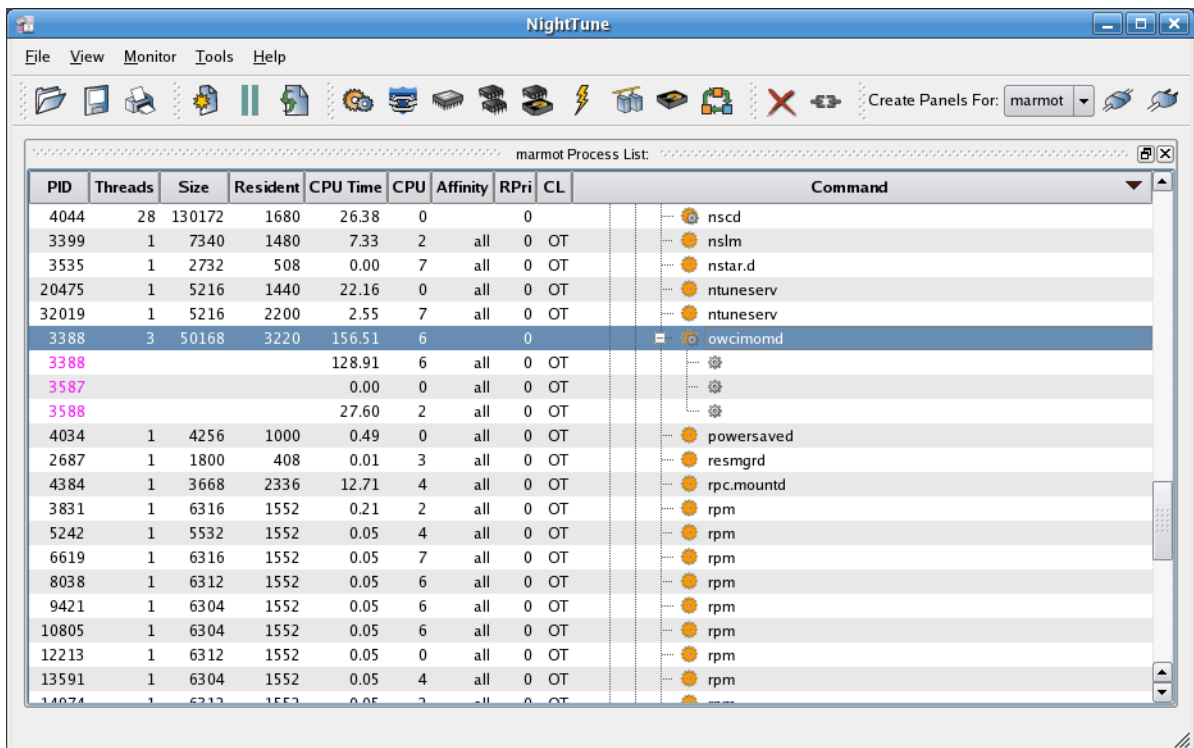


Figure 5-4. Customizing the Process Information

See “Process List Panel” on page 3-83 for more information on process monitoring.

Changing User Process Scheduling Attributes

Select the thread or process by clicking on the line displayed for that thread or process, then right-click to display the Process List context menu. Select Process Scheduler from this menu to display the Process Scheduler dialog, as shown below:

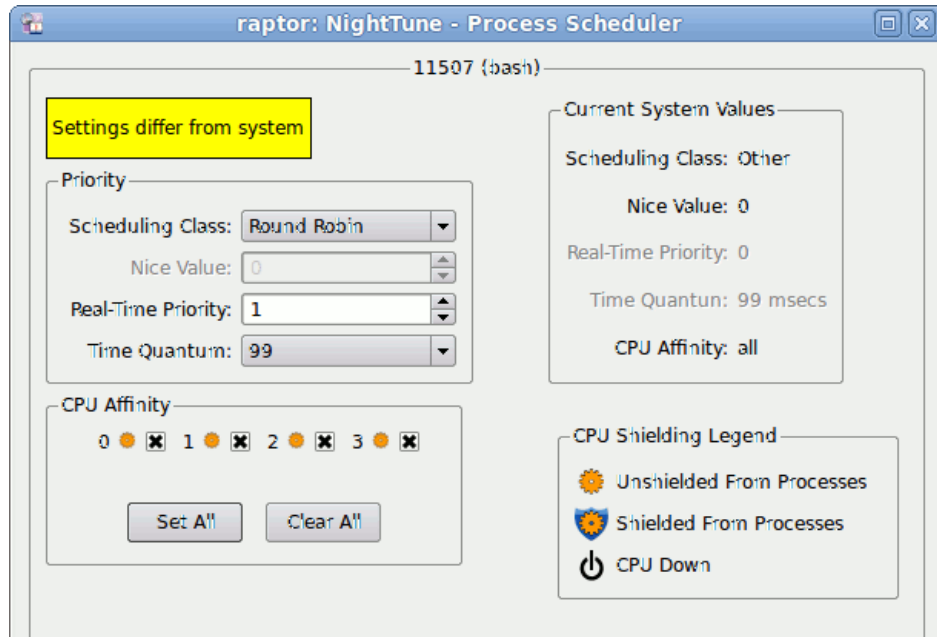


Figure 5-5. Process Scheduler Dialog

The current scheduling attributes are shown in the dialog.

Change the Scheduling Class to Round Robin, the Real-time Priority to 20, and clear all CPU Affinity boxes except for CPU 0, which should be selected.

Click on Apply to apply the changes to the process; the Process Scheduler dialog displays the changes and the Process List panel display changes on the next refresh indicating the new scheduling attributes, as shown below:

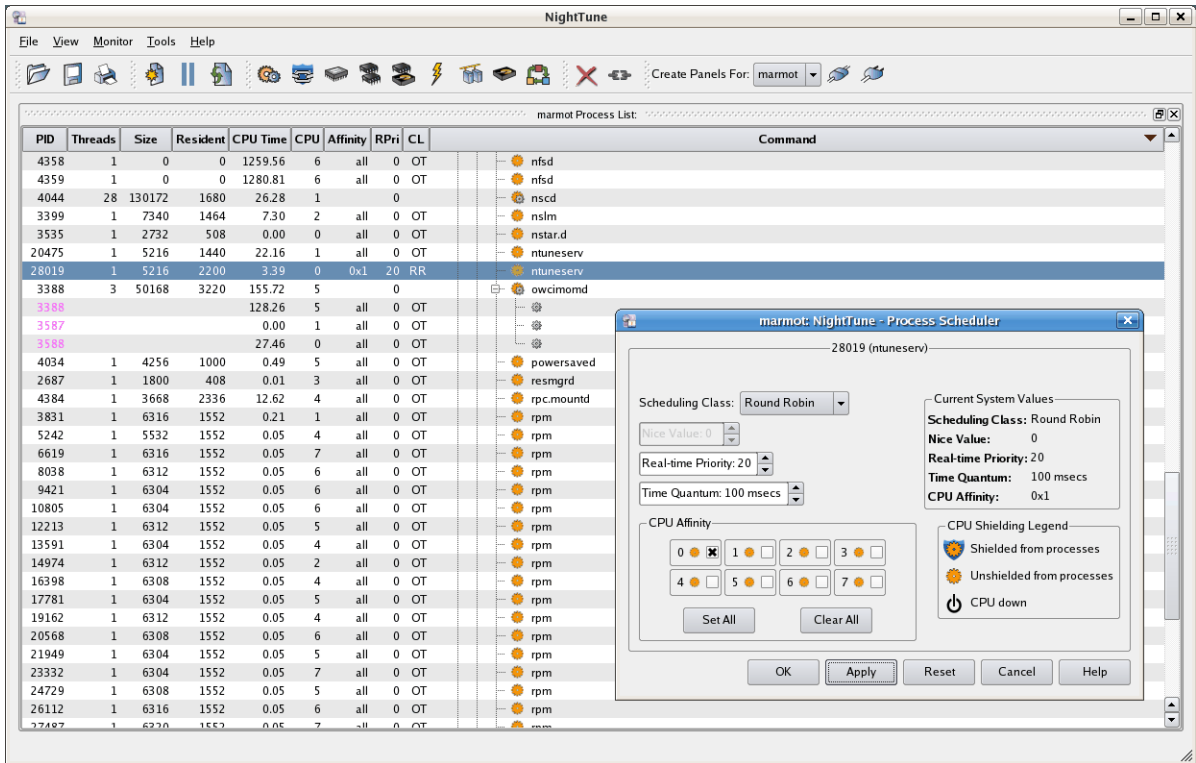


Figure 5-6. Changing User Process Scheduling Attributes

Using Drag and Drop to Change Process CPU Affinity

NightTune allows you to use drag and drop actions to change the CPU affinity of a process, group of processes, or individual threads.

If the CPU Shielding and Binding panel is not visible currently, select the CPU Shielding and Binding panel from the Monitor menu or click on the CPU Shielding and Binding tool icon.

The CPU Shielding and Binding panel will appear in the window next to the Process List panel as shown below:

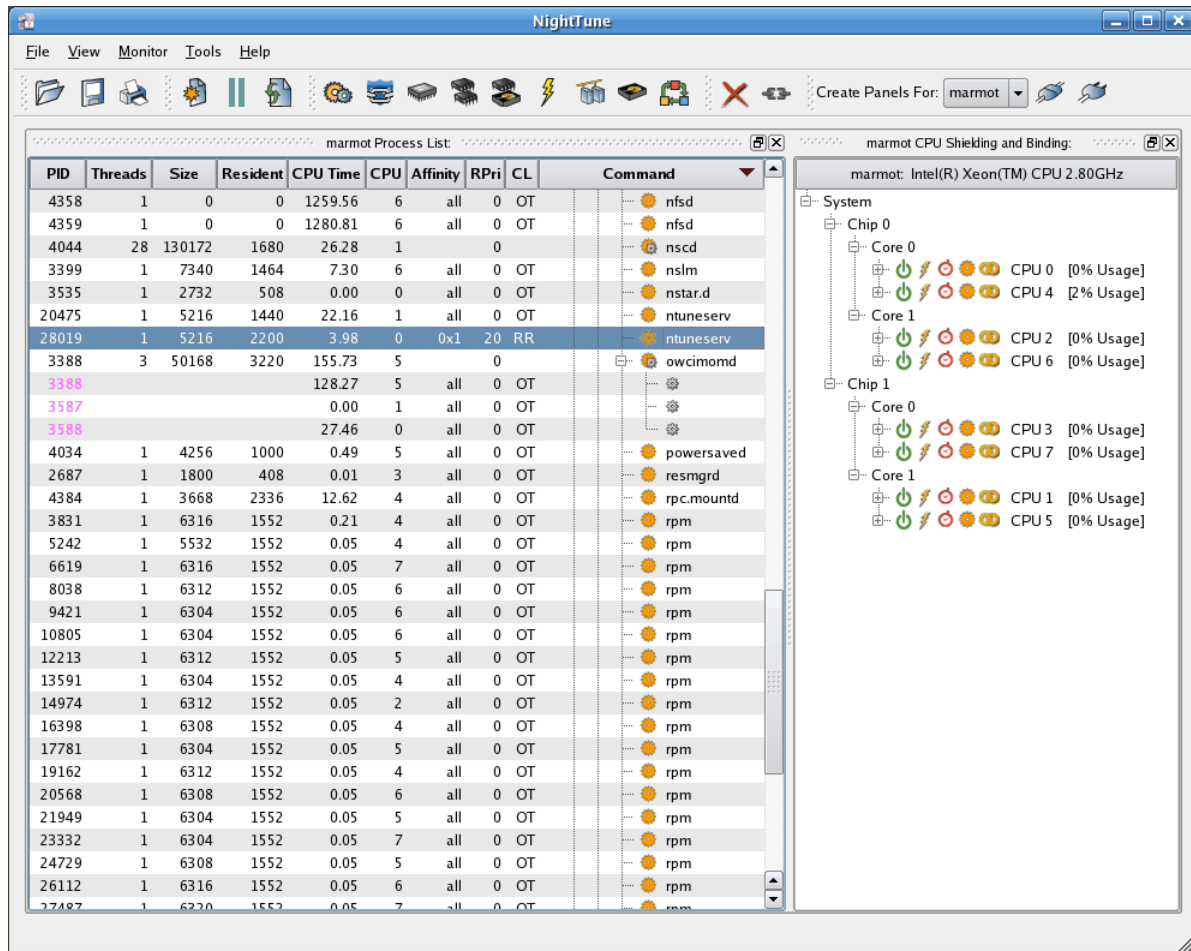


Figure 5-7. Viewing CPU Status

For each CPU displayed, a list of processes and interrupts that are bound to the CPU are available in the CPU Shielding and Binding panel. To see them for a CPU, select that CPU, right click and select Expand All.

A process or interrupt is considered bound to a CPU if the CPU affinity of the process or interrupt specifies only that single CPU.

When you changed the CPU affinity of your process in the section above to CPU 0, it became bound to CPU 0. The process now appears under Bound Processes for CPU 0, as shown below:

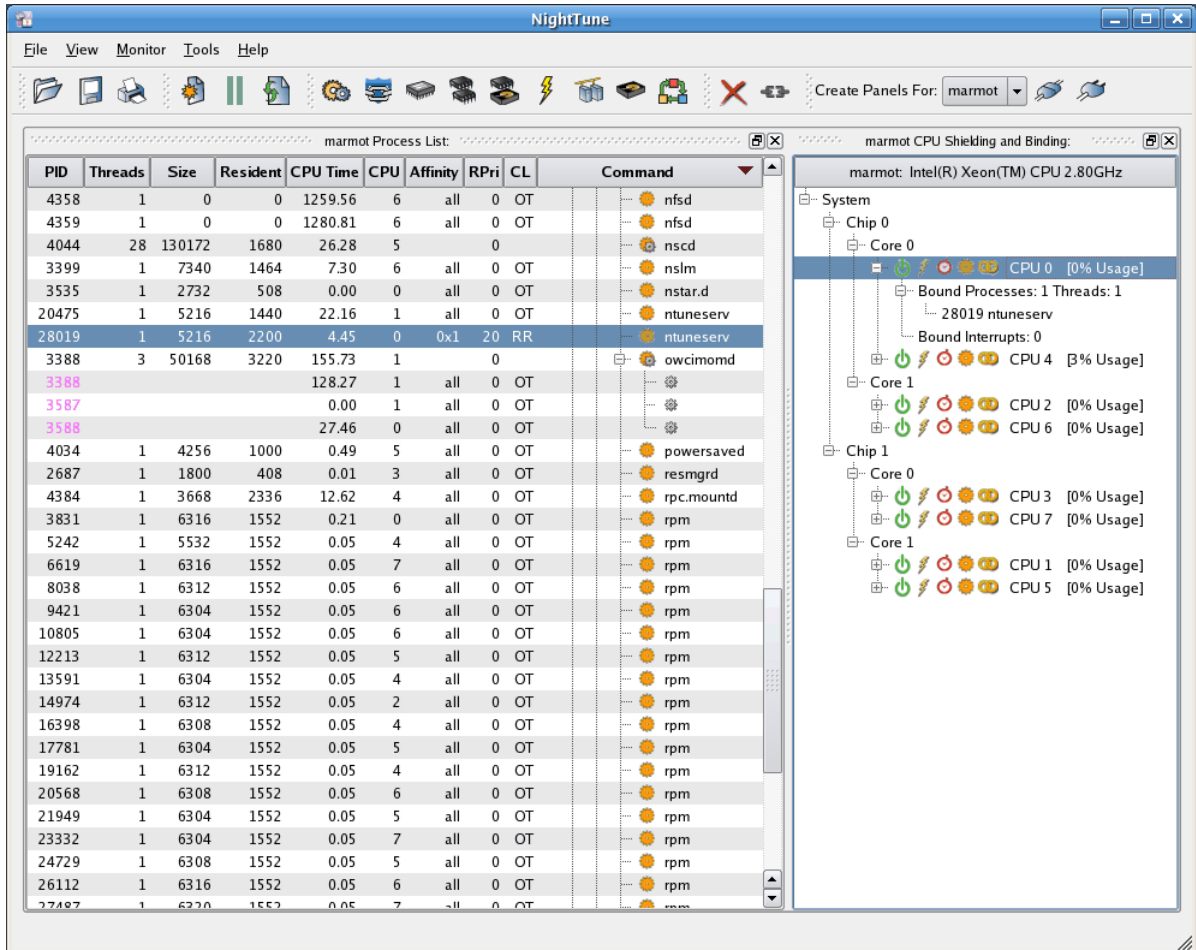


Figure 5-8. Bound Processes

To change the CPU of that process again, press and hold the middle mouse button over the process entry under CPU 0 Bound Processes. Drag the pointer to CPU 1 and release the mouse button. Then right click on CPU 1 and select **Expand All** to see the bound processes for that CPU.

The new CPU affinity is reflected in the Process List panel and under Bound Processes for CPU 1:

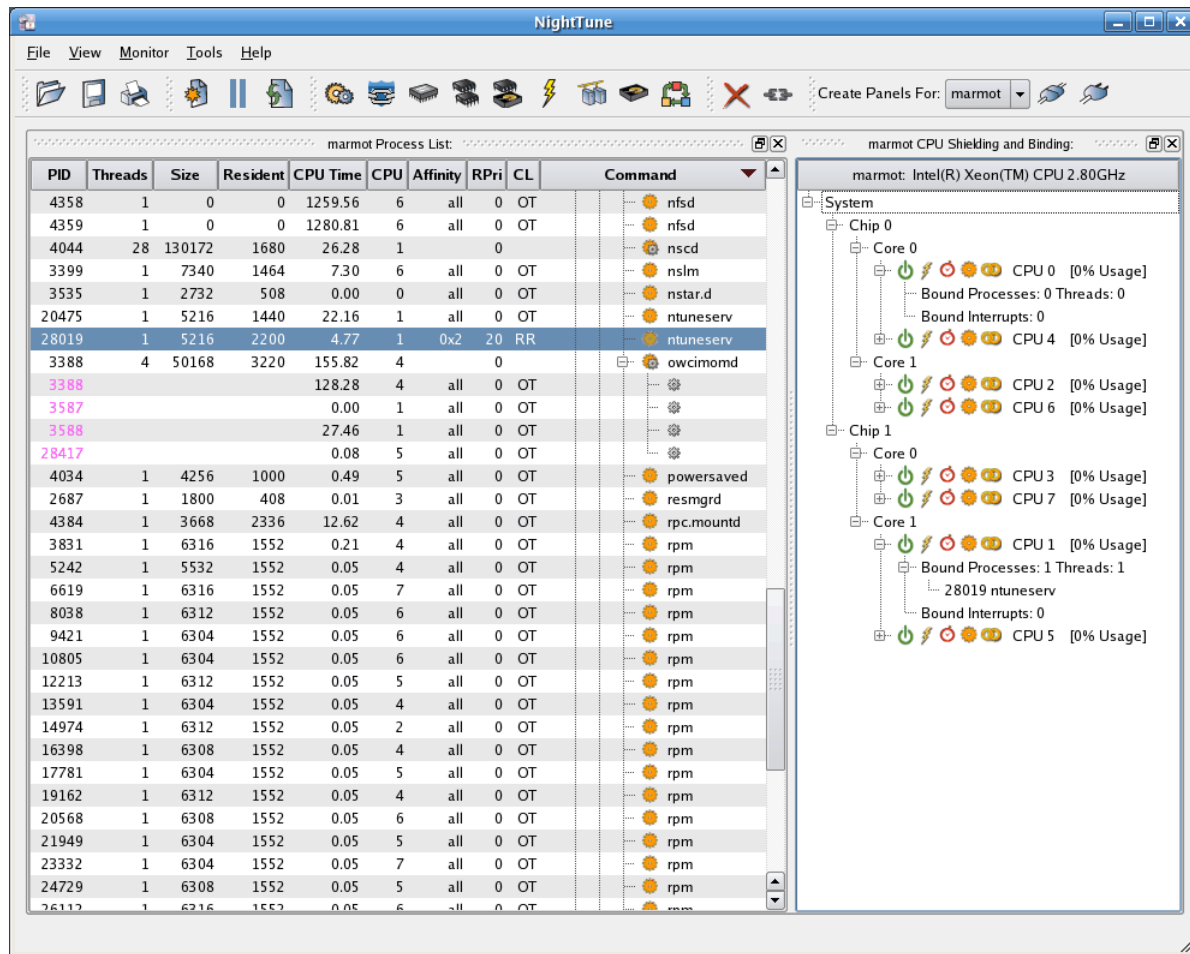


Figure 5-9. Using Drag and Drop to Change Process CPU Affinity

Similarly, you can drag a process or group of processes from the Process List panel and drop them onto a CPU entry in the CPU Shielding and Binding panel to change their CPU affinity.

See “Process List Panel” on page 3-83 and “CPU Shielding and Binding Panel” on page 3-10 for more information on process CPU affinity.

Shielding a CPU

This section describes activities associated with shielding a CPU.

Close the Process List panel from the window by clicking on the X in the upper right corner of the panel.

Ensure that the CPU Shielding and Binding panel is in the window.

Next, ensure that bound processes and interrupts are visible for all CPUs by right clicking on the System item and selecting Expand All.

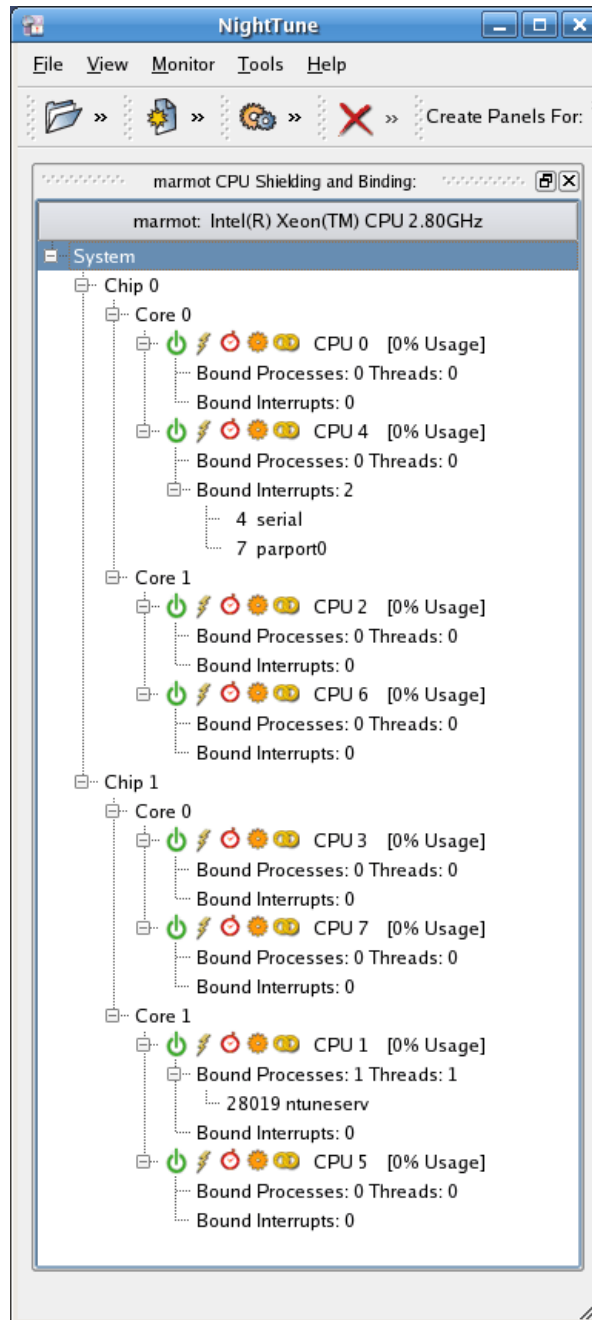


Figure 5-10. CPU Shielding and Binding Panel

The CPU Shielding and Binding panel above describes a system with two physical CPU chips, each of which has two cores, each of which has two hyper-threaded logical CPUs. CPU 0 and CPU 4 are the two logical CPUs which comprise physical chip 0's core

0; CPU 2 and CPU 6 are the two logical CPUs which comprise physical chip 0's core 1; etc.

NOTE

Not all systems support hyper-threading. Systems with hyper-threading have two CPUs per core in the CPU Shielding and Binding panel. However, the sibling CPU is not always a logical CPU with a consecutive number. In this example, CPU 4 is CPU 0's hyper-threaded sibling.

The panel above indicates that no shielding is currently active and that a single process has been bound to CPU 1.

To specify shielding on a CPU, right-click on the CPU Shielding and Binding panel and select the Change Shielding menu item. The CPU Shielding dialog appears, as shown below:

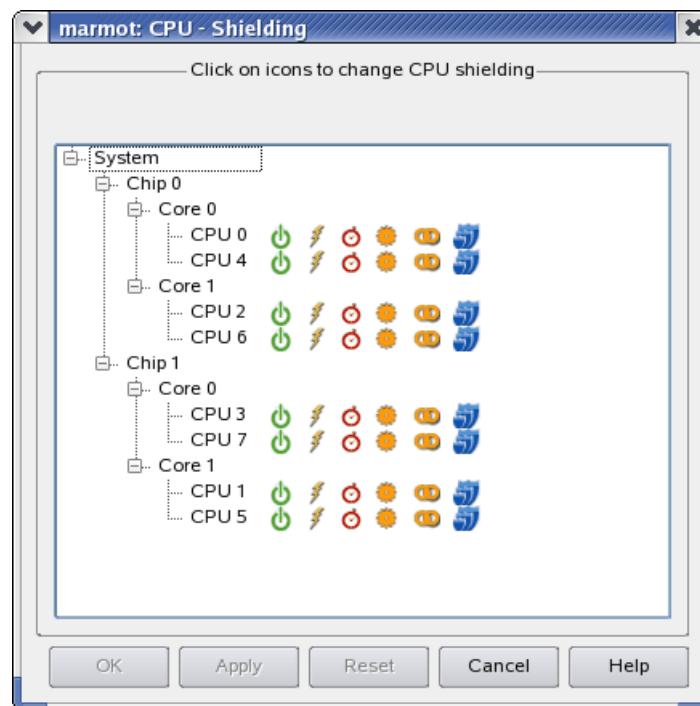



Figure 5-11. CPU Shielding Dialog

To specify maximum shielding on CPU 5, click the maximum shield icon.  in the box for CPU 5.

The CPU Shielding dialog display changes to indicate that CPU 5 is to be shielded from interrupts, processes, the local timer, and hyper-threading. To achieve the shielding from hyper-threading for CPU 5, its hyper-threading sibling, CPU 1, is marked **DOWN** automatically. If your system does not support hyper-threading, CPU 1 will remain unchanged.

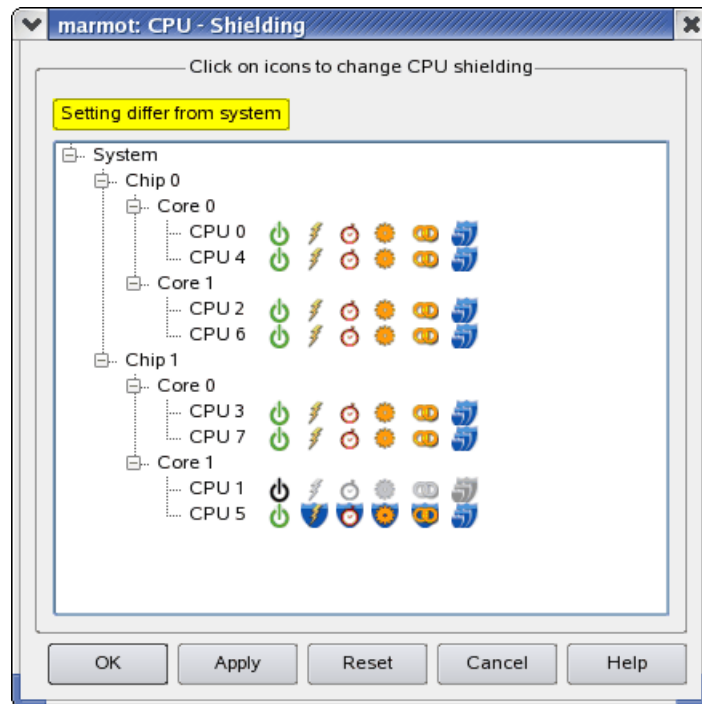


Figure 5-12. Shielding a CPU

Click on Apply to apply the shielding changes.

In the illustration above, there was a process bound to CPU 1. This prevents CPU 1 from being marked down. A diagnostic dialog similar to the following will appear if this situation exists:

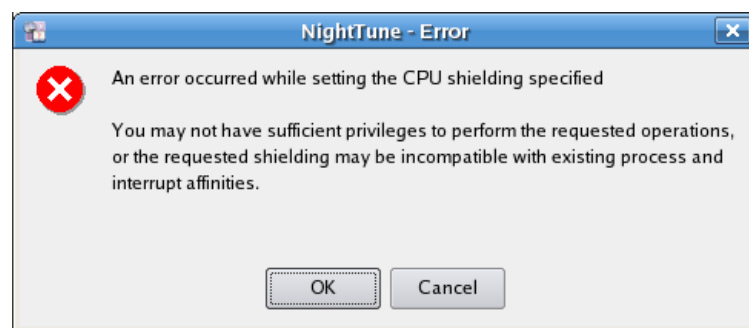


Figure 5-13. Error Shielding CPU

The process that is bound to CPU 1 must first be moved to another CPU before it can be marked Down.

Press Cancel to close the CPU Shielding dialog. Drag the process in the Bound Processes list under CPU 1 using the mouse button from CPU 1 to CPU 3. Re-open the

CPU Shielding dialog by right-clicking on the CPU Shielding and Binding panel and selecting the **Change Shielding** menu item. In the CPU Shielding dialog, click the **Max Shield** icon on CPU 1. Now click the OK button again and the CPU Shielding dialog disappears and the CPU Shielding and Binding panel will reflect the changes.

Interrupts may also be bound to specific CPUs. The **Bound Interrupts** entry in the CPU Shielding and Binding panel displays the list of interrupts that are bound. Bound interrupts prevent downing a CPU just as bound processes do. Interrupts may be dragged from the list to other CPUs in the same manner as with processes.

See “CPU Shielding and Binding Panel” on page 3-10 for more information on CPU shielding.

Changing the CPU Affinity of an Interrupt

Add the **Interrupt Activity** panel to the window using the **Monitor** menu or the **Interrupt Activity** tool icon. Close all other panels if they exist by clicking on the X in the upper right corner of each panel.

If all the **Interrupt Activity** panes are displayed, hide the bar graph and line graph panes by unchecking the **Show bar graph pane** and **Show line graph pane** items from the **Interrupt Activity Context Menu**. The context menu is displayed by right-clicking while positioned in the **Interrupt Activity** panel.

The NightTune window will display the **Interrupt Activity Text** pane as shown below:

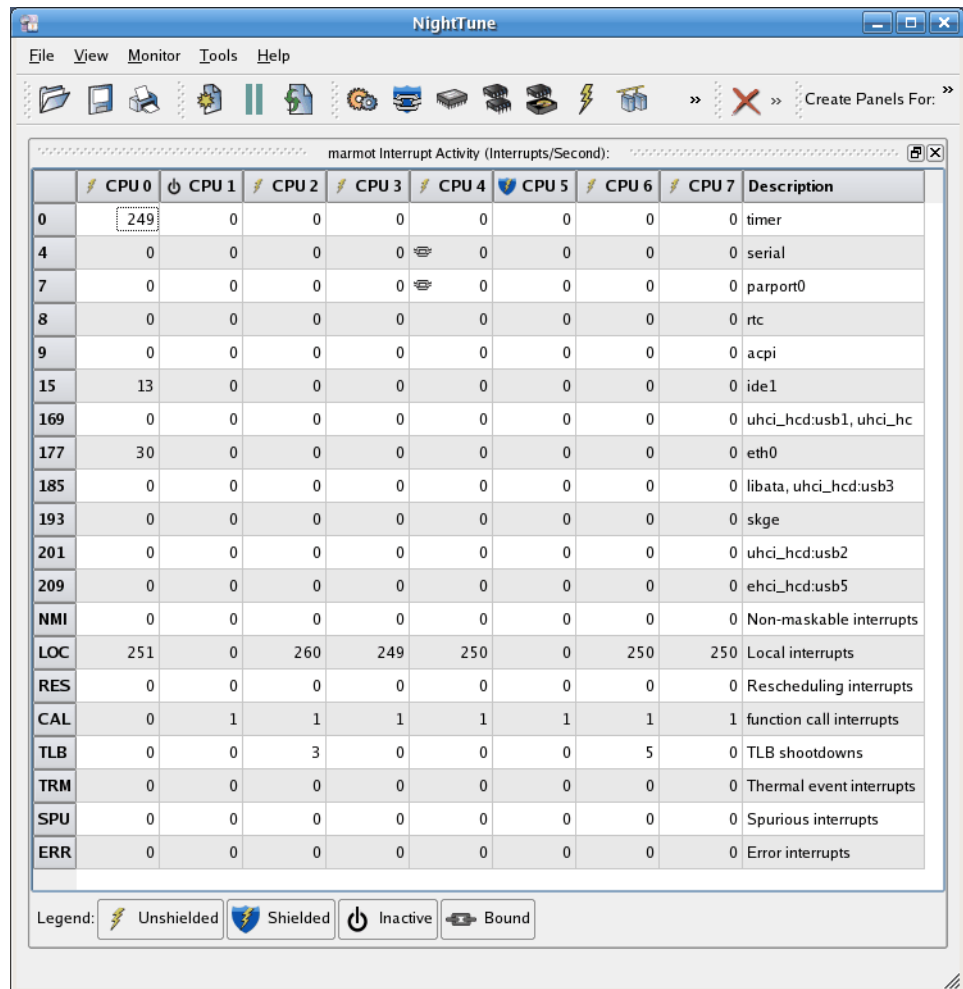


Figure 5-14. Monitoring Interrupt Activity

Click on the line associated with an interrupt, right-click to display the Interrupt Affinity context menu, then select the Set CPU Affinity menu item to display the Interrupt Affinity dialog, as shown below:

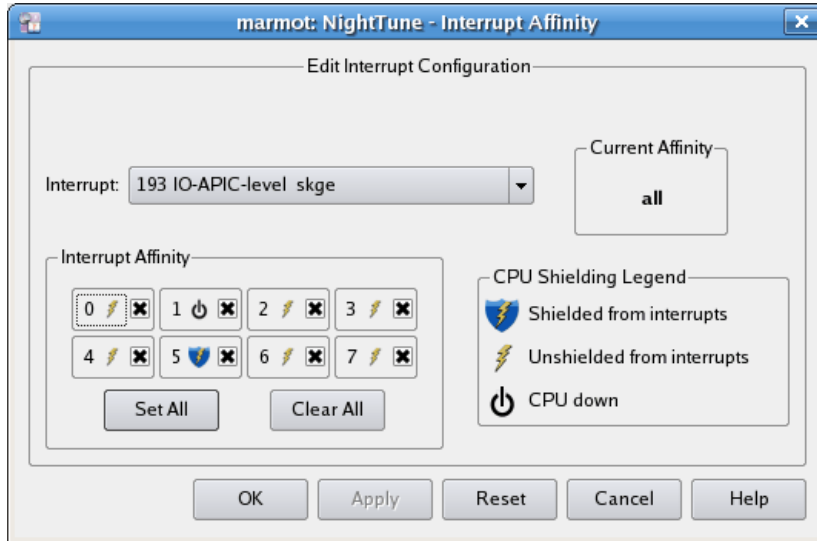


Figure 5-15. Interrupt Affinity Dialog

Clear all the CPU affinity settings by clicking on **Clear All**. Then select CPU 0 by clicking on its checkbox. Apply the change by clicking on the **Apply** button.

The CPU affinity change will be reflected in the **Interrupt Activity** panel. A Bound icon will now appear in the cell for the selected interrupt on CPU 0, indicating that its affinity mask has selected CPU 0 but not all other CPUs.

The illustration below reflects the affinity change for the selected interrupt:

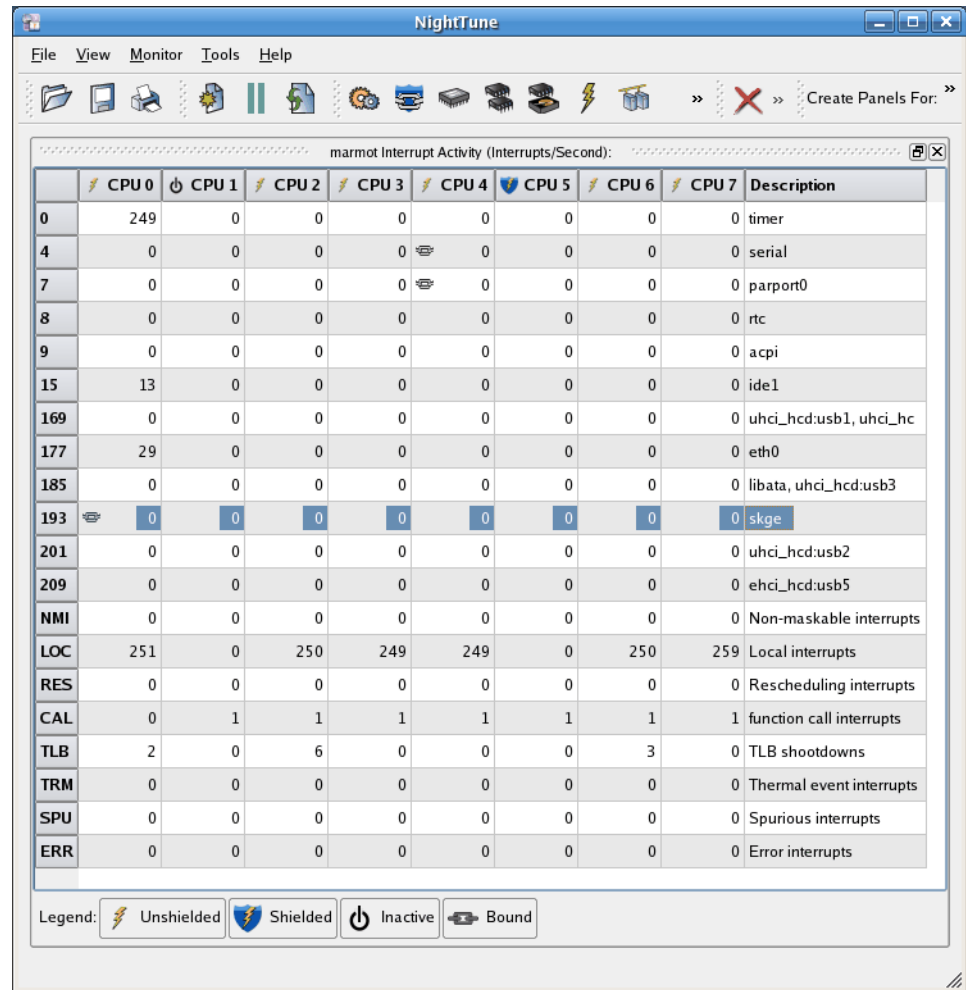


Figure 5-16. Changing the CPU Affinity of an Interrupt

Using Drag and Drop to Change Interrupt CPU Affinity

NightTune allows you to use drag and drop actions to change the affinity of an interrupt.

In the CPU Shielding and Binding panel, under CPU 0, is an item called Bound Interrupts. Under it is a list of interrupts. Click the line describing an interrupt, then drag the pointer to another CPU and release the mouse button.

The Bound Processes and the Bound Interrupts lists under the destination CPU reflect the change, as shown below:

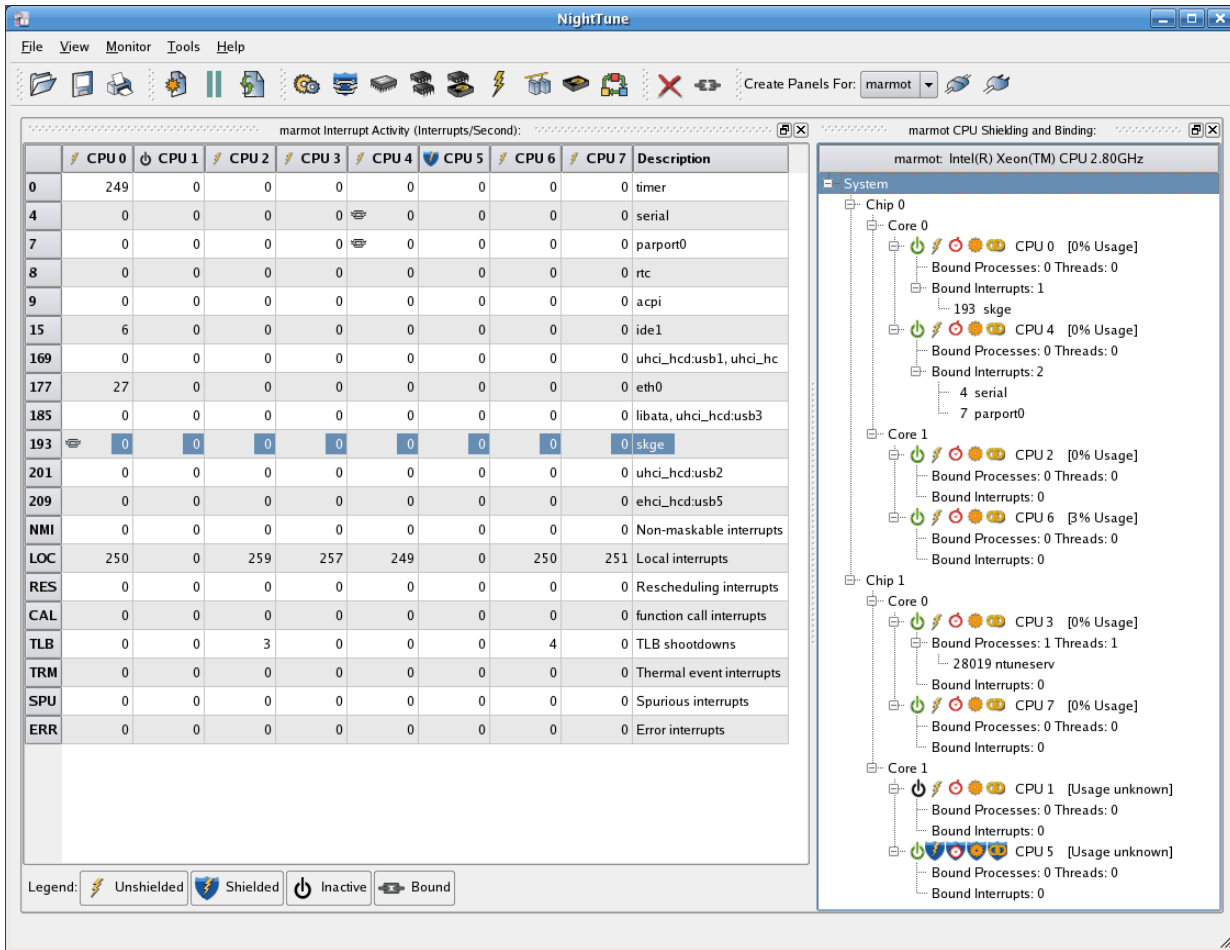


Figure 5-17. Using Drag and Drop to Change Interrupt CPU Affinity

See “Interrupt Activity Panel and Interrupt Activity Detail Panel” on page 3-45 for more information on interrupt CPU affinity.

NightStar RT Licensing

NightStar RT uses the NightStar License Manager (NSLM) to control access to the NightStar RT tools.

License installation requires a licence key provided by Concurrent (see “License Keys” on page A-1). The NightStar RT tools request a licence (see “License Requests” on page A-2) from a license server (see “License Server” on page A-2).

Two license modes are available, fixed and floating, depending on which product option you purchased. Fixed licenses can only be served to NightStar RT users from the local system. Floating licenses may be served to any NightStar RT user on any system on a network.

Tools are licensed per system, per concurrent user. A single license is shared among any or all of the NightStar RT tools for a particular user on a particular system. The intent is to allow n developers to fully utilize all the tools at the same time while only requiring n licenses. When operating the tools in remote mode, where a tool is launched on a local system but is interacting with a remote system, licenses are required only from the host system.

You can obtain a license report which lists all licenses installed on the local system, current usage, and expiration date for demo licenses (see “License Reports” on page A-3).

The default configuration includes a strict firewall which interferes with floating licenses. See “Firewall Configuration for Floating Licenses” on page A-3 for information on handling such configurations.

See “License Support” on page A-4 for information on contacting Concurrent for additional assistance with licensing issues.

License Keys

Licenses are granted to specific systems to be served to either local or remote clients, depending on the license model, fixed or floating.

License installation requires a license key provided by Concurrent. To obtain a license key, you must provide your system identification code. The system identification code is generated by the `nslm_admin` utility:

```
nslm_admin --code
```

System identification codes are dependent on system configurations. Reinstalling Linux on a system or replacing network devices may require you to obtain new license keys.

To obtain a license key, use the following URL:

<http://www.ccur.com/NightStarRTKeys>

Provide the requested information, including the system identification code. Your license key will be immediately emailed to you.

Install the license key using the following command:

```
nslm_admin --install=xxxx-xxxx-xxxx-xxxx-xxxx
```

where `xxxx-xxxx-xxxx-xxxx-xxxx` is the key included in the license acknowledgment email.

License Requests

By default, the NightStar RT tools request a license from the local system. If no licenses are available, they broadcast a license request on the local subnet associated with the system's hostname.

You can control the license requests for an entire system using the `/etc/nslm.config` configuration file.

By default, the `/etc/nslm.config` file contains a line similar to the following:

```
:server @default
```

The argument `@default` may be changed to a colon-separated list of system names, system IP addresses, or broadcast IP addresses. Licenses will be requested from each of the entities found in the list, until a license is granted or all entries in the list are exhausted.

For example, the following setting prevents broadcast requests for licenses, by only specifying the local system:

```
:server localhost
```

The following setting requests a license from `server1`, then `server2`, and then a broadcast request if those fail to serve a license:

```
:server server1:server2:192.168.1.0
```

Similarly, you can control the license requests for individual invocations of the tools using the `NSLM_SERVER` environment variable. If set, it must contain a colon-separated list of system names, system IP addresses, or broadcast IP addresses as described above. Use of the `NSLM_SERVER` environment variable takes precedence over settings defined in `/etc/nslm.config`.

License Server

The NSLM license server is automatically installed and configured to run when you install NightStar RT.

The **nslm** service is automatically activated for run levels 2, 3, 4, and 5. You can check on these settings by issuing the following command:

```
/sbin/chkconfig --list nslm
```

In rare instances, you may need to restart the license server via the following command:

```
/sbin/service nslm restart
```

See **nslm(1)** for more information.

License Reports

A license report can be obtained using the **nslm_admin** utility.

```
nslm_admin --list
```

lists all licenses installed on the local system, current usage, and expiration date (for demo licenses). Use of the **--verbose** option also lists individual clients to which licenses are currently granted.

Adding the **--broadcast** option will list this information for all servers that respond to a broadcast request on the local subnet associated with the system's hostname.

See **nslm_admin(1)** for more options and information.

Firewall Configuration for Floating Licenses

RedHawk does not support a firewall configuration by default, because iptables support is disabled. However, it is possible to build a custom kernel with iptables support enabled. If that is done, and floating licenses are used, the iptables firewall rules must be configured to allow the license requests and responses to pass.

If the system with iptables support and firewall rules is serving licenses, then the firewall rules must be arranged to allow license requests on UDP port 25517 and TCP port 25517 from any systems that will make license requests. For example, in a simple firewall, rules like the following, inserted before any DROP or REJECT rules, might work:

```
iptables -A INPUT -p udp -m udp -s subnet/mask --dport 25517 -j ACCEPT
iptables -A INPUT -p tcp -m tcp -s subnet/mask --dport 25517 -j ACCEPT
```

If the system with iptables support and firewall rules is running NightStar RT tools and receiving floating licenses, then the firewall rules must be arranged to allow license responses on UDP port 25517 from any system serving licenses. For example, in a simple firewall, rules like the following, inserted before any DROP or REJECT rules, might work:

```
iptables -A INPUT -p udp -m udp -s subnet/mask --sport 25517 -j ACCEPT
```

License Support

For additional aid with licensing issues, contact the Concurrent Software Support Center at our toll free number 1-800-245-6453. For calls outside the continental United States, the number is 1-954-283-1822. The Software Support Center operates Monday through Friday from 8 a.m. to 5 p.m., Eastern Standard Time.

You may also submit a request for assistance at any time by using the Concurrent Computer Corporation web site at <http://real-time.ccur.com/support> or by sending an email to support@ccur.com.

Kernel Dependencies

Concurrent's RedHawk kernel provides features and performance gains that are critical for the optimal operation of the NightStar RT tools.

The NightStar RT tools can operate in a host-only mode on many different Linux distributions without a RedHawk kernel, cross-targeting to RedHawk systems.

Additionally, the NightStar RT tools can function on such host systems in target mode without the RedHawk kernel, but will lack the numerous advantages afforded by running with it.

Advantages for NightView

The following advantages are afforded NightView when a RedHawk kernel is running:

- Application speed conditions

Provides “execution-speed” patches, conditions, and ignore counts.

- Signal handling

Allows NightView to pass signals directly to a particular process, avoiding context switching and stopping the process if the signal is handled.

- Branch tracking

Allows NightView to show you a history of branches. This is especially useful for programs that end up in unexpected locations, usually the result of returning from a routine with a corrupted stack frame. The branch history often allows you to locate where the program execution went awry.

Advantages for NightTrace

The following advantages are afforded NightTrace when a RedHawk tracing kernel is running:

- Kernel tracing

Users of NightTrace gain the ability to obtain kernel trace data and combine that with user trace data. Kernel tracing is an incredibly powerful feature that not only provides insight into the operating system kernel but also provides useful information relating to the execution of user applications.

The RedHawk real-timekernel is provided in three flavors:

- Tracing
- Debug
- Plain

The Tracing and Debug flavors provide the features required for NightTrace kernel tracing. These kernels can be selected at boot-time from the boot-loader menu.

- CUDA Application Tracing

While not specifically a RedHawk kernel feature, RedHawk provides an optimized NVidia driver along with a pre-built NightTrace Illuminator for the CUDA API library. This illuminator automatically instruments user applications that utilize the CUDA API so that you can see all API function entries and returns. This includes the execution of user routines on the GPU itself along with the amount of time spent executing on the GPU.

Advantages for NightProbe

The following advantages are afforded NightProbe when a RedHawk kernel is running:

- Minimal intrusion

Allows NightProbe to read and write variables without stopping the process for each sample or write operation.

- Sampling performance

Allows NightProbe to use direct memory fetches for data sampling (as opposed to programmed I/O) which is important for high-rate data acquisition.

- Concurrent debugging/probing

Allows NightProbe to probe programs already under the control of a debugger or another NightProbe session.

- PCI Device probing

Allows NightProbe to probe PCI device memory via the Base Address Register (BAR) file system.

The PCI BAR File System is only available with the RedHawk kernel from Concurrent Computer Corporation. On other systems, PCI Device probing will be disabled within NightProbe.

Advantages for NightTune

The following advantages are afforded NightTune when a RedHawk kernel is running:

- Context switch rate

Allows NightTune to display the context switch counts per CPU instead of for the overall system.
- CPU shielding

Individual CPUs can be shielded from interrupts and processes allowing CPUs to be dedicated solely to specific interrupts and processes that are bound to the CPU.
- CPU sibling interference

Individual CPUs can be marked down to avoid interfering with hyperthreaded sibling CPUs and dual-core sibling CPUs. Hyperthreaded CPUs share all the resources of their sibling CPU. Dual-core CPUs share the CPU cache and a path to memory with their sibling CPU.
- Detailed memory information

Detailed process memory descriptions include the residency and lock state of any page in a process, and their association with physical memory pools for NUMA systems.
- Kernel Activity and Single Process Activity panels

Provides non-intrusive monitoring of kernel or process/thread activity, including percent of time spent in individual routines in the kernel, in shared libraries, and in user processes. Routines are described using their symbolic name.
- Single Process Counter

Provides non-intrusive monitoring of low-level CPU operations, such as cpu cycles, instructions, bus cycles, branches, cache hits and misses, page faults, cpu migrations, and context switches for individual processes/threads.
- CUDA Configuration and Activity

While not specifically a RedHawk kernel feature, RedHawk provides an optimized NVidia driver that allows NightTune to show detailed CUDA configuration information as well as CUDA device activity, including GPU usage, fan speed, GPU memory usage, etc.

Frequency Based Scheduler

The Frequency Based Scheduler is only available on RedHawk systems from Concurrent Computer Corporation. It is required for all NightSim usage.

FBS Process Deadlines are only available for use on RedHawk 5.2.1 and later systems.

On systems without FBS Process Deadline support, the “Apply Deadline” group box will appear shaded and disabled.

NightSim is only included in NightStar distributions intended for use on RedHawk systems.

A

Add Page 2-8, 2-18

B

BAR1 Memory Free 3-22
BAR1 Memory Total 3-22
BAR1 Memory Used 3-22
Below Application Clock Time 3-23
Below Base Clock Time 3-23
Board Liimit Violation Time 3-23
Board Temperature 3-21
Bound Interrupts 3-10, 3-51
Bound Processes 3-10
Bound to System 2-24

C

Capabilities 1-3, 3-114
Changing

- CPU Shielding 3-11
- Interrupt CPU Affinity 3-51, 5-14
- Scheduling Attributes 3-97, 5-6

Command Line Options 1-6
Commas 2-24
Config File

- Current Name 2-23
- Load 2-2, 2-18
- Save 2-2, 2-18
- Save As 2-3

Connect to a Remote System 2-14, 2-20
Context Switches Panel

- Bar Graph Pane 3-8
- Line Graph Pane 3-8
- Menu Selection 2-11
- Text Pane 3-7, 3-126, 3-129, 3-130
- Toolbar Icon 2-19

CPU Affinity 3-51, 3-99, 3-120, 5-6, 5-7, 5-14
CPU Monitoring 1-1

CPU Shielding and Binding Panel 3-10

- CPU Shielding Dialog 3-11
- Drag and Drop Operations 3-13
- Menu Selection 2-11
- Pop-up Menu 3-14
- Refresh Intervals 2-23
- Toolbar Icon 2-19

CPU Shielding Dialog 3-11
CPU Shielding Operations 3-11
CPU Time 3-119
CPU Usage Panel

- Bar Graph Pane 3-18
- Line Graph Pane 3-18
- Menu Selection 2-11
- Text Pane 3-16
- Toolbar Icon 2-19

Create panels for 2-20
CUDA BAR1 Memory Free 3-22
CUDA BAR1 Memory Total 3-22
CUDA BAR1 Memory Used 3-22
CUDA Below Application Clock Time 3-23
CUDA Below Base Clock Time 3-23
CUDA Board Limit Violation Time 3-23
CUDA Board Temperature 3-21
CUDA Configuration Panel

- Configure Devices Dialog 3-39
- Menu Selection 2-11
- Pop-up Menu 3-38

CUDA Devices Configuration Dialog 3-39
CUDA Energy Consumption 3-21
CUDA Fan Speed 3-21
CUDA GPU Temperature 3-21
CUDA GPU USage 3-21
CUDA Low Utilization Violation Time 3-23
CUDA Memory Activity 3-21
CUDA Memory Free 3-22
CUDA Memory Size 3-119
CUDA Memory Temperature 3-21
CUDA Memory Total 3-22
CUDA Memory Used 3-22
CUDA Panel

- Bar Graph Pane 3-24
- Line Graph Pane 3-26
- Menu Selection 2-11
- Text Pane 3-20

- Toolbar Icon 2-19
- CUDA PCIe Replays 3-23
- CUDA PCIe Rx Throughput 3-22
- CUDA PCIe Tx Throughput 3-22
- CUDA Performance State 3-21
- CUDA Power Usage 3-21
- CUDA Power Violation Time 3-23
- CUDA Reliability Violation Time 3-23
- CUDA Sync Boost Violation Time 3-23
- CUDA Text Pane 4-10
- CUDA Thermal Violation Time 3-23
- CUDA VR1 Temperature 3-21
- CUDA VR2 Temperature 3-21
- CUDA VR3 Temperature 3-22
- CUDA VR4 Temperature 3-22

D

- Data Memory Size 3-119
- Debug 3-97
- Delete Current Page 2-8
- Disconnect from a Remote System 2-15, 2-20
- Disk Activities
 - Average Service Time 3-43
 - Average Wait Time 3-43
 - Read Operations 3-42
 - Read Sector Transfers 3-43
 - Total Operations 3-42
 - Total Sector Transfers 3-43
 - Write Operations 3-42
 - Write Sector Transfers 3-43
- Disk Activity Panel
 - Bar Graph Pane 3-44
 - Line Graph Pane 3-45
 - Menu Selection 2-11
 - Text Pane 3-42
 - Toolbar Icon 2-19
- Display Localized Numbers 2-24
- Display Options 2-24
- Drag and Drop 3-85, 3-98, 5-7, 5-17

E

- Energy Consumption 3-21
- Environment variable
 - NSLM_SERVER A-2
- Environment Variables 3-115
- Exit NightTune 2-7

F

- Fan Speed 3-21
- File Descriptors 3-110
- File Menu 2-1, 2-2
- Filter Processes Dialog 3-91
- Find bar 3-86
- Fixed licenses A-1
- Floating licenses A-1
- Freeze Page 2-8, 2-9, 2-18

G

- GPU Temperature 3-21
- GPU Usage 3-21
- Guide to Operations 5-1

H

- Help Menu 2-1, 2-17
- Hyper-threading 3-12, 5-11

I

- Idle Time 3-17
- Interrupt Activity Panel
 - Affinity Dialog 3-51, 5-15
 - Bar Graph Pane 3-48
 - Drag and Drop Operations 3-47
 - Interrupt Affinity Dialog 5-16
 - Line Graph Pane 3-49
 - Menu Selection 2-11
 - Pop-up Menu 3-50
 - Text Pane 3-46
 - Toolbar Icon 2-19
- Interrupt Affinity 3-51, 5-14
- Interrupt Affinity Dialog 3-51, 5-16
- Interrupt Detail Activity Panel
 - Bar Graph 3-48
 - Line Graph Pane 3-49
 - Menu Selection 2-12
 - Pop-up Menu 3-50
 - Text Pane 3-46
- Interrupts, Bound 3-10
- IRQ Time 3-17

K

- Kernel Activity Panel
 - Menu Selection 2-12
 - Refresh Intervals 2-23
 - Toolbar Icon 2-19
- Kernel Virtual Memory 3-57
- Kernel Virtual Memory Panel
 - Bar Graph Pane 3-58
 - Line Graph Pane 3-59
 - Menu Selection 2-12
 - Text Pane 3-57
- Kill Process 2-20

L

- Library Calls, trace 3-96
- License A-1
 - fixed A-1
 - installation A-1
 - keys A-1
 - modes A-1
 - ns1m_admin A-1, A-3
 - report A-3
 - requests A-2
 - server A-2
 - support A-4
- License manager 1-2
- licenses 1-2
- Load Config File 2-2, 2-18
- Load System Default Config 2-2
- Local Operation 1-3
- Localized Numbers 2-24
- Logging 1-2, 2-7
- Low Utilization Violation Time 3-23
- ltrace 3-96

M

- Memory
 - Kernel Virtual 3-57
 - NUMA 3-75
 - Physical 3-60
 - Swap 3-65
- Memory Activity 3-21
- Memory Activity Panel
 - Bar Graph Pane 3-56
 - Line Graph Pane 3-57
 - Menu Selection 2-12

- Text Pane 3-55
- Memory Free 3-22
- Memory Temperature 3-21
- Memory Total 3-22
- Memory Used 3-22
- Menu Bar 2-1
 - File 2-1, 2-2
 - Help 2-1, 2-17
 - Monitor 2-1, 2-10
 - Tools 2-1, 2-15
 - View 2-1, 2-7
- Monitor Menu 2-1, 2-10
- Monitoring
 - CPU Status 1-1, 3-16
 - CUDA 3-20
 - Kernel Activity 3-53
 - Processes 1-1, 3-83, 5-2
 - Single Process 1-2
 - Single Process Activity 3-121
 - Single Process Counters 3-125
 - System Activities 1-1
- Most Recent CPU 3-120

N

- Network Activity
 - Collision Rate 3-68
 - Input Packet Rate 3-67
 - Output Packet Rate 3-67
 - Packet Error Rate 3-68
- Network Activity Panel
 - Bar Graph Pane 3-68
 - Line Graph Pane 3-69
 - Menu Selection 2-12
 - Text Pane 3-67
 - Toolbar Icon 2-19
- New Page 2-8, 2-18
- Nice Value 3-99, 3-120, 5-6
- NightProbe Monitor 2-15
- NightSim Scheduler 2-16
- NightStar Licence Manager 1-2
- NightTrace Analyzer 2-16
- NightView 3-97
- NightView Debugger 2-16
- NLSM 1-2
 - ns1m_admin A-1, A-3
 - NSLM_SERVER A-2
- NUMA Activity Panel
 - Bar Graph Pane 3-76
 - Line Graph Pane 3-77
 - Menu Selection 2-13
 - Text Pane 3-75

- NUMA Configuration Panel
 - Context Menu 3-79
 - CPUs Pane 3-77
 - Distance Pane 3-78
 - Menu Selection 2-13
 - NUMA Panel
 - Bar Graph Pane 3-73
 - Line Graph Pane 3-74
 - Menu Selection 2-13
 - Text Pane 3-70
 - NUMA Systems 3-70, 3-75, 3-77
- O**
- Options 1-6
- P**
- Page
 - Add 2-8, 2-18
 - Freeze 2-8, 2-9, 2-18
 - Refresh 2-8, 2-19
 - Page Transfer Rates 3-55
 - Panels 3-1
 - Context Switches 3-7
 - CPU Shielding and Binding 3-10
 - CPU Usage 3-16
 - CUDA 3-20
 - CUDA Configuration 3-27
 - Disk Activity 3-42
 - Interrupt Activity 3-45, 5-14
 - Interrupt Detail Activity 3-45
 - Kernel Activity 3-53
 - Kernel Virtual Memory 3-57
 - Memory Activity 3-55
 - Network Activity Table 3-67
 - NUMA 3-70
 - NUMA Activity 3-75
 - NUMA Configuration 3-77
 - PCI Configuration 3-79
 - Physical Memory 3-60
 - Process List 3-83, 5-2
 - Single Process Activity 3-121
 - Single Process Counters 3-125
 - Swap 3-65
 - Parent Process ID 3-118
 - PCI 3-79
 - PCI Configuration Panel
 - Context Menu 3-82
 - PCIe Replays 3-23
 - PCIe Rx Throughput 3-22
 - PCIe Tx Throughput 3-22
 - Performance State 3-21
 - Physical Memory Allocation 3-60
 - Physical Memory Panel
 - Bar Graph Pane 3-62
 - Line Graph Pane 3-64
 - Menu Selection 2-12
 - Text Pane 3-60
 - Toolbar Icon 2-19
 - Power Usage 3-21
 - Power Violation Time 3-23
 - Preferences 2-2, 2-7, 2-22
 - Print NightTune Window 2-18
 - Priority 3-120, 3-121
 - Privileges 1-3
 - Process
 - Bound 3-10
 - Kill 2-20
 - Monitor 2-19
 - Unbind 2-20
 - Process Details
 - Capability Tab 3-114
 - Environment Tab 3-115
 - File Descriptors Tab 3-110
 - Limits Tab 3-116
 - Memory Maps 3-104
 - Memory Tab 3-105
 - Memory Usage Tab 3-102
 - Signals Tab 3-112
 - Process Fields
 - CPU Affinity 3-120
 - CPU Time 3-119
 - CUDA Memory Size 3-119
 - Customize 5-4
 - Data Memory Size 3-119
 - Most Recent CPU 3-120
 - Nice Value 3-120
 - Parent Process ID 3-118
 - Priority 3-120
 - Process ID 3-118
 - Real-Time Priority 3-121
 - Resident Memory Size 3-119
 - Scheduling Class 3-121
 - State 3-118
 - System Time 3-120
 - Threads 3-119
 - User ID 3-118
 - User Name 3-118
 - User Time 3-120
 - Virtual Memory Size 3-119
 - Process ID 3-118
 - Process List Panel 5-4
 - Context Menu 3-87

- Drag and Drop 3-85
- Filter Processes Dialog 3-91
- Find bar 3-86
- Menu Selection 2-13
- Process Details Window 3-101
- Process Fields Menu 3-117
- Process Scheduler Dialog 3-97, 3-100
- Refresh Intervals 2-23
- Toolbar Icon 2-19
- Process Monitoring 1-1, 5-2
- Process Scheduler Dialog 3-97
- Process Scheduling Operations 3-100, 5-6

R

- Real-Time Priority 3-99, 3-121, 5-6
- Refresh Page 2-8, 2-19
- Reliability Violation Time 3-23
- Remote Operation 1-3, 1-7, 2-14, 2-20
- Rename Current Page 2-8
- Resident Memory Size 3-119
- Resource Limits 3-116

S

- Save Config File 2-2, 2-18
- Save Config File As 2-3
- Scheduling Class 3-98, 3-121
- Set Checkpoint for all Text Panes 2-8
- Shield-All 3-12
- Shielding a CPU 5-10
- Signal Handling 3-112
- Single Process Activity Panel
 - Menu Selection 2-13
 - Toolbar Icon 2-19
- Single Process Counters Panel
 - Menu Selection 2-13
 - Toolbar Icon 2-19
- Single Process Monitoring 1-2
- Single Process Panel
 - Refresh Intervals 2-23
- SoftIrq Time 3-17
- State 3-118
- strace 3-96
- Swap Memory Panel
 - Menu Selection 2-12
 - Toolbar Icon 2-19
- Swap Panel
 - Bar Graph Pane 3-66
 - Line Graph Pane 3-66

- Text Pane 3-65
- Sync Boost Violation Time 3-23
- System
 - Bind to Current 2-24
 - Connect 2-14, 2-20
 - Current 2-15, 2-20
 - Disconnect 2-15, 2-20
 - Target 1-7
- System Activity Monitoring 1-1
- System Calls, trace 3-96
- System Status Panel
 - Refresh Intervals 2-23
- System Time 3-17, 3-120

T

- Tabbed Pages 2-1
 - Delete Current Page 2-8
 - New Page 2-8, 2-18
 - Rename Current Page 2-8
- Thermal Violation Time 3-23
- Threads 3-119
- Time Quantum 3-99, 5-6
- Toolbar 2-18
- Tools Menu 2-1, 2-15
- Trace Library Calls 3-96
- Trace Output Window 3-96
- Trace System Calls 3-96

U

- Unbind Process 2-20
- User ID 3-118
- User Name 3-118
- User Time 3-17, 3-22, 3-120
- Using NightTune 5-1

V

- View Menu 2-1, 2-7
- Virtual Memory Size 3-119
- vmalloc 3-57
- VR1 Temperature 3-21
- VR2 Temperature 3-21
- VR3 Temperature 3-22
- VR4 Temperature 3-22

W

Windows

Menu Bar 2-1

Multiple 2-1