JUNIPER NETWORKS

Technology Overview

Frequently Asked Questions: Routing Protocol Process Memory

Release
11.4

Published: 2012-03-16

Juniper Networks, Inc.
1194 North Mathilda Avenue
Sunnyvale, California 94089
USA
408-745-2000
www.juniper.net

Technology Overview Frequently Asked Questions: Routing Protocol Process Memory
Release 11.4

Revision History
October 2011—R1 Junos OS 11.4

The information in this document is current as of the date on the title page.

YEAR 2000 NOTICE

Juniper Networks hardware and software products are Year 2000 compliant. Junos OS has no known time-related limitations through the year 2038. However, the NTP application is known to have some difficulty in the year 2036.

END USER LICENSE AGREEMENT

The Juniper Networks product that is the subject of this technical documentation consists of (or is intended for use with) Juniper Networks software. Use of such software is subject to the terms and conditions of the End User License Agreement ("EULA") posted at http://www.juniper.net/support/eula.html. By downloading, installing or using such software, you agree to the terms and conditions of that EULA.

Table of Contents

Introduction

This document discusses the software processes that run on the Juniper Networks Junos operating system, and provides an overview of the routing protocol process (rpd). It also presents the most frequently asked questions about the routing protocol process memory utilization, operation, interpretation of related command outputs, and troubleshooting the software process.

Routing Protocol Process Memory FAQ Overview

The Juniper Networks Junos operating system (Junos OS) is based on the FreeBSD Unix operating system. The open source software is modified and hardened to operate in the device's specialized environment. For example, some executables have been deleted while other utilities have been de-emphasized. Additionally, certain software processes have been added to enhance the routing functionality. The result of this transformation is the kernel, the heart of the Junos OS software.

The kernel is responsible for generating multiple processes that perform the actual functions of the device. Each process operates in its own protected memory space, providing isolation between the processes and resiliency in the event of a process failure. This is important in a core routing platform because a single process failure does not cause the entire device to cease functioning.

Some of the common software processes include the routing protocol process (rpd) that controls the device's protocols, the device control process (dcd) that controls the device's interfaces, the management process (mgd) that controls user access to the device, the chassis process (chassisd) that controls the device's properties itself, and the Packet Forwarding Engine process (pfed) that controls the communication between the device's Packet Forwarding Engine and the Routing Engine. Besides the above processes, there are other specialized processes that support additional functionality, such as the Simple Network Management Protocol (SNMP), Virtual Router Redundancy Protocol (VRRP), and Class of Service (CoS).

The routing protocol process is a software process within the Routing Engine software that controls the routing protocols that run on the device. Its functionality includes all protocol messages, routing table updates, and implementation of routing policies.

The routing protocol process starts all configured routing protocols and handles all routing messages. It maintains one or more routing tables, which consolidate the routing information learned from all routing protocols. From this routing information, the routing protocol process determines the active routes to network destinations and installs these routes into the Routing Engine's forwarding table. Finally, it implements the routing policy, which allows you to control the routing information that is transferred between the routing protocols and the routing table. Using the routing policy, you can filter and limit the transfer of information as well as set properties associated with specific routes.

Related Documentation

- Routing Protocol Process Memory FAQs on page 2

Routing Protocol Process Memory FAQs

The following sections present the most frequently asked questions and answers related to the routing protocol process memory utilization, operation, interpretation of related command outputs, and troubleshooting the software process.

Routing Protocol Process Memory Utilization FAQs

This section presents frequently asked questions and answers related to the memory usage of the routing protocol process.

Why does the routing protocol process use excessive memory?

The routing protocol process uses hundreds of megabytes of RAM in the Routing Engine to store information needed for the operation of routing and related protocols, such as BGP, OSPF, ISIS, RSVP, LDP, and MPLS. Such huge consumption of memory is common for the process, as the information it stores includes routes, next hops, interfaces, routing policies, labels, and label-switched paths (LSPs). Because access to the RAM memory is much faster than access to the hard disk, most of the routing protocol process information is stored in the RAM memory instead of using the hard disk space. This ensures that the performance of the routing protocol process is maximized.

How can I check the amount of memory the routing protocol process is using?

You can check the routing protocol process memory usage by entering the **show system processes** and the **show task memory** Junos OS command-line interface (CLI) operational mode commands.

The **show system processes** command displays information about software processes that are running on the device. You can check the routing protocol process memory usage by using the **show system processes** command with the **extensive** option.

The **show task memory** command displays a report generated by the routing protocol process on the memory utilization for routing protocol tasks on the Routing Engine. Although the report generated by the routing protocol process is on its own memory usage, it does not display all the memory used by the process. The value reported by the routing protocol process does not account for the memory used for the **TEXT** and **STACK** segments, or the memory used by the process's internal memory manager. The **show task memory** command also does not include the memory which has been deactivated by the routing protocol process, although some or all of that deactivated memory has not actually been freed by the kernel.

For more information about checking the routing protocol process memory usage, see Check Routing Protocol Process (rpd) Memory Usage in the *Junos OS Baseline Network Operations Guide*.

For more information about the show system processes command and the show task memory command, see the *Junos OS System Basics and Services Command Reference*.

I just deleted many routes from the routing protocol process. Why is the routing protocol process still using so much memory?

The **show system processes extensive** command displays a **RES** value measured in kilobytes. This value represents the amount of process memory resident in the physical memory. This is also known as RSS or Resident Set Size. Any amount of memory deactivated by the process might still be considered part of the **RES** value. Generally, the kernel defers the actual freeing of deactivated memory until there is a memory shortage. This can lead to large discrepancies between the values reported by the routing protocol process and the kernel, even after the routing protocol process has deactivated a large amount of memory.

Interpreting Routing Protocol Process-Related Command Outputs FAQs

This section presents frequently asked questions and answers about the routing protocol process-related Junos OS CLI command outputs that are used to display the memory usage of the routing protocol process.

How do I interpret memory numbers displayed in the show system processes extensive command output?

The **show system processes extensive** command displays exhaustive system process information about software processes that are running on the device. This command is equivalent to the UNIX **top** command. However, the UNIX **top** command shows real-time memory usage, with the memory values constantly changing, while the **show system processes extensive** command provides a snapshot of memory usage in a given moment.

To check overall CPU and memory usage, enter the **show system processes extensive** command. Refer to Table 1 on page 4 for information about the **show system processes extensive** command output fields.

```
user@host> show system processes extensive
last pid:   544;  load averages:  0.00,  0.00,  0.00    18:30:33
37 processes:  1 running, 36 sleeping

Mem: 25M Active, 3968K Inact, 19M Wired, 184K Cache, 8346K Buf, 202M Free
Swap: 528M Total, 64K Used, 528M Free
  PID USERNAME PRI NICE SIZE    RES STATE    TIME   WCPU    CPU COMMAND
  544 root      30   0   604K   768K RUN      0:00  0.00%  0.00% top
    3 root      28   0     0K    12K psleep   0:00  0.00%  0.00% vmdaemon
    4 root      28   0     0K    12K update   0:03  0.00%  0.00% update
  528 aviva     18   0   660K   948K pause    0:00  0.00%  0.00% tcsh
  204 root      18   0   300K   544K pause    0:00  0.00%  0.00% csh
  131 root      18   0   332K   532K pause    0:00  0.00%  0.00% cron
  186 root      18   0   196K    68K pause    0:00  0.00%  0.00% watchdog
   27 root      10   0   512M 16288K mfsidl   0:00  0.00%  0.00% mount_mfs
    1 root      10   0   620K   344K wait     0:00  0.00%  0.00% init
  304 root       3   0   884K   900K ttyin    0:00  0.00%  0.00% bash
  200 root       3   0   180K   540K ttyin    0:00  0.00%  0.00% getty
  203 root       3   0   180K   540K ttyin    0:00  0.00%  0.00% getty
  202 root       3   0   180K   540K ttyin    0:00  0.00%  0.00% getty
  201 root       3   0   180K   540K ttyin    0:00  0.00%  0.00% getty
  194 root       2   0  2248K  1640K select   0:11  0.00%  0.00% rpd
  205 root       2   0   964K   800K select   0:12  0.00%  0.00% tnp.chassisd
  189 root       2 -12   352K   740K select   0:03  0.00%  0.00% xntpd
  114 root       2   0   296K   612K select   0:00  0.00%  0.00% amd
```

```
  188 root      2   0   780K   600K select   0:00  0.00%  0.00% dcd
  527 root      2   0   176K   580K select   0:00  0.00%  0.00% rlogind
  195 root      2   0   212K   552K select   0:00  0.00%  0.00% inetd
  187 root      2   0   192K   532K select   0:00  0.00%  0.00% tnetd
   83 root      2   0   188K   520K select   0:00  0.00%  0.00% syslogd
  538 root      2   0  1324K   516K select   0:00  0.00%  0.00% mgd
   99 daemon    2   0   176K   492K select   0:00  0.00%  0.00% portmap
  163 root      2   0   572K   420K select   0:00  0.00%  0.00% nsrexecd
  192 root      2   0   560K   400K select   0:10  0.00%  0.00% snmpd
  191 root      2   0  1284K   376K select   0:00  0.00%  0.00% mgd
  537 aviva     2   0   636K   364K select   0:00  0.00%  0.00% cli
  193 root      2   0   312K   204K select   0:07  0.00%  0.00% mib2d
    5 root      2   0     0K    12K pfesel   0:00  0.00%  0.00% if_pfe
    2 root    -18   0     0K    12K psleep   0:00  0.00%  0.00% pagedaemon
    0 root    -18   0     0K     0K sched    0:00  0.00%  0.00% swapper
```

Table 1 on page 4 describes the output fields that represent the memory values for the **show system processes extensive** command. Output fields are listed in the approximate order in which they appear.

Table 1: show system processes extensive Output Fields

Field Name	Field Description
Mem	Information about physical and virtual memory allocation.
Active	Memory allocated and actively used by the process.
Inact	Memory allocated but not recently used, or memory deactivated by the processes. Inactive memory remains mapped in the address space of one or more processes and, therefore, counts toward the RSS value of those processes.
Wired	Memory that is not eligible to be swapped, usually used for in-kernel memory structure, memory physically locked by a process, or both.
Cache	Freed memory that is no longer associated with any process but still has valid contents that correspond to some file system blocks. Cache pages can be reclaimed as is when the corresponding file system blocks are accessed again. However, when the system is under memory pressure, the contents of Cache pages could be erased by the kernel and the pages reused to service any memory allocation requests.
Buf	Size of the virtual memory buffer used to hold data recently called from the disk.
Free	Free memory that is neither associated with any process nor contains any valid contents.
Swap	Information about swap memory. • Total—Total space on the swap device. • Used—Memory swapped to disk. • Free—Unused space available on the swap device.

The rest of the command output displays information about the memory usage of each process. The **SIZE** field indicates the size of the virtual address space, and the **RES** field indicates the amount of the process in physical memory, which is also known as RSS or Resident Set Size. For more information, see the show system processes command in the *Junos OS System Basics and Services Command Reference*.

What is the difference between Active and Inact memory that is displayed by the show system processes extensive command?

When the system is under memory pressure, the pageout process can free up memory from the **Inact** and, if necessary, **Active** pools after first preserving the contents of those pages on the swap device or backing file systems if necessary. When the pageout process runs, it scans memory to see which pages are good candidates to be unmapped and freed up. Thus, the distinction between **Active** and **Inact** memory is only used by the pageout process to determine which pool of pages to free first at the time of a memory shortage.

The pageout process first scans the **Inact** list and checks whether the pages on this list have been accessed since the time they have been listed here. The pages that have been accessed are moved from the **Inact** list to the **Active** list. On the other hand, pages that have not been accessed become prime candidates to be freed by the pageout process. If the pageout process cannot produce enough free pages from the **Inact** list, pages from the **Active** list are freed up.

Because the pageout process runs only when the system is under memory pressure, the pages on the **Inact** list remain untouched – even if they have not been accessed recently – when the amount of **Free** memory is adequate.

How do I interpret memory numbers displayed in the show task memory command output?

The **show task memory** command provides a comprehensive picture of the memory utilization for routing protocol tasks on the Routing Engine. The routing protocol process is the main task that uses Routing Engine memory.

To check routing process memory usage, enter the **show task memory** command.

```
user@host> show task memory
Memory                 Size (kB)  %Available  When
  Currently In Use:        29417          3%  now
  Maximum Ever Used:       33882          4%  00/02/11 22:07:03
  Available:              756281        100%  now
```

Table 2 on page 5 describes the output fields for the **show task memory** command. Output fields are listed in the approximate order in which they appear.

Table 2: show task memory Output Fields

Field Name	Field Description
Memory Currently In Use	Memory currently in use. Dynamically allocated memory plus the **DATA** segment memory in kilobytes.
Memory Maximum Ever Used	Maximum memory ever used.
Memory Available	Memory currently available.

The **show task memory** command does not display all the memory used by the routing protocol process. This value does not account for the memory used for the **TEXT** and

STACK segments, or the memory used by the routing protocol process's internal memory manager. The **show task memory** command also does not include the memory which has been deactivated by the routing protocol process, although some or all of that deactivated memory has not actually been freed by the kernel.

Why is the Memory Currently In Use value less than the RES value?

The **show task memory** command displays a **Memory Currently In Use** value measured in kilobytes. This value is the dynamically allocated memory plus the **DATA** segment memory. The **show system processes extensive** command displays a **RES** value measured in kilobytes. This value represents the amount of process memory resident in the physical memory. This is also known as RSS or Resident Set Size.

The **Memory Currently In Use** value does not account for all of the memory that the routing protocol process uses. This value does not include the memory used for the **TEXT** and the **STACK** segments, and a small percentage of memory used by the routing protocol process's internal memory manager. The **show task memory** command also does not include the memory which has been deactivated by the routing protocol process, although some or all of that deactivated memory has not actually been freed by the kernel.

Any amount of memory deactivated by the routing protocol process might still be considered part of the **RES** value. Generally, the kernel defers the actual freeing of deactivated memory until there is a memory shortage This can lead to large discrepancies between the **Memory Currently In Use** value and the **RES** value.

Routing Protocol Process Memory Swapping FAQs

This section presents frequently asked questions and answers related to the memory swapping of the routing protocol process from the Routing Engine memory to the hard disk memory.

Why does the system start swapping when I try to perform a core dump using the request system core-dumps command?

The **request system core-dumps** command displays a list of system core files created when the device has failed. This command can be useful for diagnostic purposes. Each list item includes the file permissions, number of links, owner, group, size, modification date, path, and filename. You can use the **core-filename** option and the **core-file-info**, **brief**, and **detail** options to display more information about the specified core dump files.

You can use the **request system core-dumps** command to perform a non-fatal core dump without aborting the routing protocol process. To do this, the routing protocol process is forked, generating a second copy, and then aborted. This process can double the memory consumed by the two copies of the routing protocol process, pushing the system into swap.

Why does the show system processes extensive command show that memory is swapped to disk even though there is plenty of free memory?

Memory can remain swapped out indefinitely if it is not accessed again. Therefore, the **show system processes extensive** command shows that memory is swapped to disk even though there is plenty of free memory. Such a situation is not unusual.

Troubleshooting the Routing Protocol Process FAQs

This section presents frequently asked questions and answers related to a shortage of memory and memory leakage by the routing protocol process.

What does the RPD_OS_MEMHIGH message mean?

The **RPD_OS_MEMHIGH** message is written into the system message file if the routing protocol process is running out of memory. This message alerts you that the routing protocol process is using the indicated amount and percentage of Routing Engine memory, which is considered excessive. This message is generated either because the routing protocol process is leaking memory or the use of system resources is excessive, perhaps because routing filters are not configured properly or the configured network topology is very complex.

When the memory utilization for the routing protocol process is using all available Routing Engine DRAM memory or reaches the maximum memory limit, a message of the following form is written every minute in the syslog message file:

```
RPD_OS_MEMHIGH: Using 188830 KB of memory, 100 percent of available
```

This message includes the amount (in kilobytes), the percentage, or both of the available memory in use.

This message should not appear under normal conditions, as any further memory allocations usually require a portion of existing memory to be written to swap. As a recommended solution, increase the amount of RAM in the Routing Engine. For more information, see http://kb.juniper.net/InfoCenter/index?page=content&id=KB14186 .

What can I do when there is a memory shortage even after a swap?

We do not recommend that the system operate in this state, notwithstanding the existence of swap. The protocols that run in the routing protocol process usually have a real-time requirement that cannot reliably withstand the latency of being swapped to hard disk. If the memory shortage has not resulted from a memory leak, then either a reduction in the memory usage or an upgrade to a higher memory-capacity Routing Engine is required.

What is the task_timer?

The source of a routing protocol process memory leak can usually be identified by dumping the timers for each task. You can use the **show task *task-name*** command to display routing protocol tasks on the Routing Engine. Tasks can be baseline tasks performed regardless of the device's configuration, and other tasks that depend on the device configuration.

For more information, see the show task command in the *Junos OS System Basics and Services Command Reference*.

Related Documentation

- Routing Protocol Process Memory FAQ Overview on page 1