

Troubleshoot High Memory Utilization Issues with CPS

Contents

[Introduction](#)

[Prerequisites](#)

[Requirements](#)

[Components Used](#)

[Background Information](#)

[Problem](#)

[Procedure to Resolve High Memory Utilization Issues with CPS](#)

Introduction

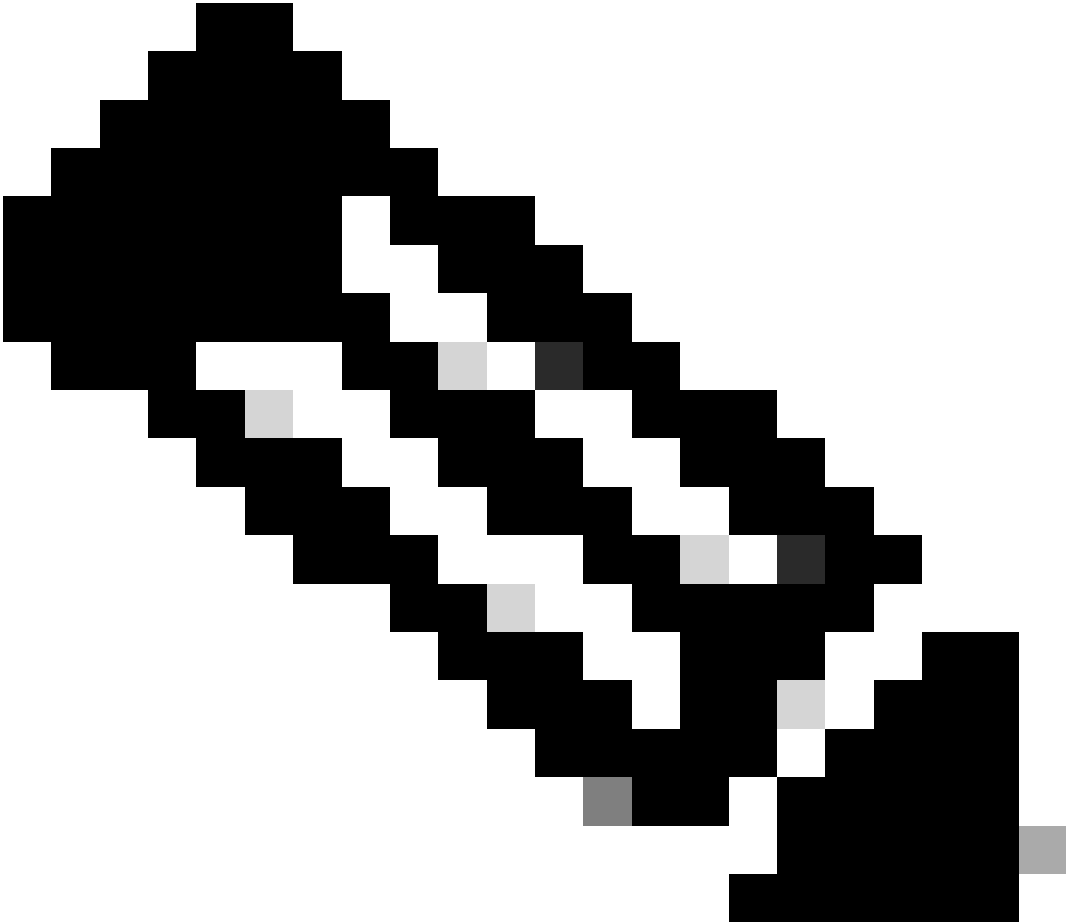
This document describes the procedure to troubleshoot High Memory Utilization Issues with Cisco Policy Suite (CPS).

Prerequisites

Requirements

Cisco recommends that you have knowledge of these topics:

- Linux
- CPS
- MongoDB



Note: Cisco recommends that have privilege root access to CPS CLI.

Components Used

The information in this document is based on these software and hardware versions:

- CPS 20.2
- Unified Computing System (UCS)-B
- MongoDB v3.6.17

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If your network is live, ensure that you understand the potential impact of any command.

Background Information

Linux has a wide range of tools to support, manage, monitor, and deploy software applications.

Services and features added to the product application can consume considerable memory. Memory

optimization for Linux servers not only makes applications run smoother and faster, it also reduces the risk of data loss and server crashes.

To optimize memory for Linux machines, you first need to understand how memory works in Linux. You start with some memory terms, discuss how Linux handles memory, and then learn how to troubleshoot and prevent memory issues.

The total amount of memory that one machine can contain is based on the operating system's architecture.

The entire memory in Linux is called virtual memory—it includes physical memory (often called RAM - Random Access Memory) and swap space. The physical memory of a system cannot be increased unless we add more RAM. However, the virtual memory can be increased by the use of swap space from the hard disk.

RAM determines whether your machine can handle high memory-consumption processes.

Data from the user, computer processes, and Hard Disk Drive (HDD) is sent to RAM. If needed, RAM stores and sends it back to the user or HDD. If the data needs to be persistent, RAM sends it to the Central Processing Unit (CPU).

To check for available free space in your machine, you can use the free command.

```
[root@installer ~]# free -h
total used free shared buff/cache available
Mem: 11Gi 1.3Gi 2.9Gi 105Mi 7.4Gi 10Gi
Swap: 0B 0B 0B
[root@installer ~]#
```

Problem

A Linux server can consume a considerable amount of memory for various reasons. For quick troubleshooting effectively, first, you need to rule out the most likely reasons.

Java process:

There are several applications implemented by the use of Java, and their incorrect implementation or configuration can lead to high memory usage in the server. The two most common causes are wrong configuration in caching and session caching anti-pattern.

Caching is a common way to achieve high performance for applications but when applied incorrectly, it can end up hurting system performance instead. The wrong configuration could make the cache grow too quickly, and leave less memory for other processes running in the system.

Session caching is often used when storing the intermediate state of the application. It allows developers to store users per session and makes it easy to save or get data object value. However, developers tend to forget to clean up session caching data afterwards.

When working with databases in Java, a hibernate session is commonly used to create connections and manage the session between the server and the database. But there's an error that frequently occurs when developers work with hibernate sessions. Instead of being isolated for thread safety, the hibernate session is included in the same Hypertext Transfer Protocol (HTTP) session. This makes the application store more states than necessary, and with only a few users, memory usage greatly increases.

Database:

When discussing high memory–consumption processes, you must mention databases. With many reads and writes to the database while the application handles user requests, our database can consume considerable memory.

Take a MongoDB database as a reference: To achieve high performance, it applies a buffer mechanism for caching and indexing data. If you configure the database to use maximum memory when you have several requests to the database, the memory in your Linux server can soon be overwhelmed.

CPS memory consumption can be monitored by the use of appropriate KPIs in Grafana graphs or other tools to monitor. If memory consumption increases beyond the default threshold of 90% on any CPS Virtual Machine (VM), CPS can generate a Low Memory alarm for that VM. This threshold is configurable in the CPS Deployment Template by the use of the `free_mem_per` settings.

Identify the process/utility which causes high memory usage:

1. Log in to the VM that has thrown a Low Memory Alarm.
2. Navigate to `/var/log` directory and check in `top_memory_consuming_processes` file to identify the Process ID (PID) with a high % memory consumption.

```
***** Date: Tue May 16 05:06:01 UTC 2023 *****
PID PPID CMD %MEM %CPU RSS PRI STAT PSR WCHAN NI P
9435 1 /usr/bin/java -XX:OnOutOfMe 26.7 77.9 4353796 5 S<l 2 - -15 *
24139 1 /usr/java/default/bin/java 1.0 0.0 174636 20 S'l 3 - 0 *
2905 2862 /usr/sbin/collectd -C /etc/ 1.0 0.2 169104 20 S'l 1 hrtimer_nanosl 0 *
913 1 /usr/lib/systemd/systemd-jo 0.4 0.1 69364 20 Ss 5 do_epoll_wait 0 *
1513 1 /usr/libexec/platform-pytho 0.1 0.0 27912 20 Ss'l 5 - 0 *
3379 2905 /usr/sbin/collectd -C /etc/ 0.1 0.0 23716 20 S'l 3 - 0 *
3377 2905 /usr/sbin/collectd -C /etc/ 0.1 0.0 23712 20 S'l 4 - 0 *
3378 2905 /usr/sbin/collectd -C /etc/ 0.1 0.0 23712 20 S'l 5 - 0 *
3380 2905 /usr/sbin/collectd -C /etc/ 0.1 0.0 23712 20 S'l 5 - 0 *
***** END *****
```

3. Validate the process by this command, whether it is an application or database process.

```
<#root>
```

```
#ps -ef | grep <PID>
```

Procedure to Resolve High Memory Utilization Issues with CPS

Optimizing memory in Linux is complex, and fixing an overloaded memory requires significant effort.

Approach 1.

Detect and Reclaim Cached Memory:

In some cases, a Low Memory Alarm can be a result of Linux memory management allocating objects in

cache.

To evaluate how much memory a VM has cached and to trigger Linux to free some of the cached memory.

1. Compare the amount of memory cached on two or more CPS VMs, to do so run the `free -m` command on each VM.

```
[root@dc1-qns01 ~]# free -m
total used free shared buff/cache available
Mem: 15876 5262 4396 808 6217 9628
Swap: 4095 0 4095
[root@dc1-qns01 ~]#
```

2. To reclaim some of the inactive cached memory, run this command.

```
#free && sync && echo 3 > /proc/sys/vm/drop_caches && echo "" && free
```

```
[root@dc1-qns01 ~]# free -m
total used free shared buff/cache available
Mem: 15876 5016 8782 872 2076 9809
Swap: 4095 0 4095
[root@dc1-qns01 ~]#
```

Please note:

1. This command discards cache objects which can cause a temporary increase in Input Output (IO) and Central Processing Unit (CPU) usage, so it is recommended to run this command at off-peak hours/maintenance window.
2. This is a non-destructive command and only free memory that is not in use.

If the low memory alarm is still unresolved, then proceed with Approach 2.

Approach 2.

If high memory consumption is due to any of the application processes such as QNS and so on.

1. Restart the process.

<#root>

Command Syntax:

```
#monit restart <process name>
```

2. Verify reduction in memory usage by `free-m` command.

If the low memory alarm is still unresolved, then proceed with Approach 3.

Approach 3.

Restart the VM for which alarms have been generated, as VM restart is typically done to increase resources to the VM (disk memory CPU).

If high memory utilization has been noticed for sessionmgr VM, then proceed with Approach 4.

Approach 4.

1. Log in to the VM for which high memory usage has been noticed.

2. Navigate to `/var/log` directory and check in `mongodb-<xxxx>.log` file for warn/messages related to memory consumption and `writeConcernMajorityJournalDefault` parameter.

```
2022-12-13T00:30:39.012+0200 I REPL [replexec-0] ** WARNING: This replica set node is running without j
2022-12-13T00:30:39.012+0200 I REPL [replexec-0] ** writeConcernMajorityJournalDefault option to the re
2022-12-13T00:30:39.012+0200 I REPL [replexec-0] ** is set to true. The writeConcernMajorityJournalDefa
2022-12-13T00:30:39.012+0200 I REPL [replexec-0] ** option to the replica set config must be set to fal
2022-12-13T00:30:39.012+0200 I REPL [replexec-0] ** or w:majority write concerns will never complete.
2022-12-13T00:30:39.012+0200 I REPL [replexec-0] ** In addition, this node's memory consumption may inc
2022-12-13T00:30:39.012+0200 I REPL [replexec-0] ** available free RAM is exhausted.
```

3. Login to respective mongoShell and verify current values of `mongo protocolVersion` and `writeConcernMajorityJournalDefault` .

```
set04:PRIMARY> rs.status().optimes.lastCommittedOpTime.t
NumberLong(0)
set04:PRIMARY>
```



Note: It's always a negative value in NumberLong o/p with mongo protocol version 0.

```
set04:PRIMARY> rs.conf().writeConcernMajorityJournalDefault
set04:PRIMARY>
```

Note: If the output returns none, then you must consider that `writeConcernMajorityJournalDefault` value is set as true by default.

4. If `protocolVersion` is 1 and `writeConcernMajorityJournalDefault` value is true , then run these commands from respective mongoShell to modify `writeConcernMajorityJournalDefault` value to false.

```
#cfg=rs.conf()
#cfg.writeConcernMajorityJournalDefault=false
#rs.reconfig(cfg)
```

5. Verify that `writeConcernMajorityJournalDefault` value has changed to false .

```
set03:PRIMARY> rs.conf().writeConcernMajorityJournalDefault
false
set03:PRIMARY>
```


6. Verify reduction in memory usage by `free-m` command.