

|  |
| --- |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| [**ETS**](https://wiki.ucern.com/x/EUDzTw) |

General Swap space discussion

Swap space is a common aspect of computing today, regardless of operating system. Linux uses swap space to increase the amount of virtual memory available to a host. It can use one or more dedicated swap partitions or a swap file on a regular filesystem or logical volume.

There are two basic types of memory in a typical computer. The first type, random access memory (RAM), is used to store data and programs while they are being actively used by the computer. Programs and data cannot be used by the computer unless they are stored in RAM. RAM is volatile memory; that is, the data stored in RAM is lost if the computer is turned off.

Hard drives are magnetic media used for long-term storage of data and programs. Magnetic media is nonvolatile; the data stored on a disk remains even when power is removed from the computer. The CPU (central processing unit) cannot directly access the programs and data on the hard drive; it must be copied into RAM first, and that is where the CPU can access its programming instructions and the data to be operated on by those instructions. During the boot process, a computer copies specific operating system programs, such as the kernel and init or systemd, and data from the hard drive into RAM, where it is accessed directly by the computer’s processor, the CPU.

Swap space

Swap space is the second type of memory in modern Linux systems. The primary function of swap space is to substitute disk space for RAM memory when real RAM fills up and more space is needed.

For example, assume you have a computer system with 8GB of RAM. If you start up programs that don’t fill that RAM, everything is fine and no swapping is required. But suppose the spreadsheet you are working on grows when you add more rows, and that, plus everything else that's running, now fills all of RAM. Without swap space available, you would have to stop working on the spreadsheet until you could free up some of your limited RAM by closing down some other programs.

The kernel uses a memory management program that detects blocks, aka pages, of memory in which the contents have not been used recently. The memory management program swaps enough of these relatively infrequently used pages of memory out to a special partition on the hard drive specifically designated for “paging,” or swapping. This frees up RAM and makes room for more data to be entered into your spreadsheet. Those pages of memory swapped out to the hard drive are tracked by the kernel’s memory management code and can be paged back into RAM if they are needed.

The total amount of memory in a Linux computer is the RAM plus swap space and is referred to as virtual memory.

Types of Linux swap

Linux provides for two types of swap space. By default, most Linux installations create a swap partition, but it is also possible to use a specially configured file as a swap file. A swap partition is just what its name implies—a standard disk partition that is designated as swap space by the mkswap command.

A swap file can be used if there is no free disk space in which to create a new swap partition or space in a volume group where a logical volume can be created for swap space. This is just a regular file that is created and preallocated to a specified size. Then the mkswap command is run to configure it as swap space. I don’t recommend using a file for swap space unless absolutely necessary.

Thrashing

Thrashing can occur when total virtual memory, both RAM and swap space, become nearly full. The system spends so much time paging blocks of memory between swap space and RAM and back that little time is left for real work. The typical symptoms of this are obvious: The system becomes slow or completely unresponsive, and the hard drive activity light is on almost constantly.

If you can manage to issue a command like free that shows CPU load and memory usage, you will see that the CPU load is very high, perhaps as much as 30 to 40 times the number of CPU cores in the system. Another symptom is that both RAM and swap space are almost completely allocated.

After the fact, looking at SAR (system activity report) data can also show these symptoms. I install SAR on every system I work on and use it for post-repair forensic analysis.

What is the right amount of swap space?

Many years ago, the rule of thumb for the amount of swap space that should be allocated on the hard drive was 2X the amount of RAM installed in the computer (of course, that was when most computers' RAM was measured in KB or MB). So if a computer had 64KB of RAM, a swap partition of 128KB would be an optimum size. This rule took into account the facts that RAM sizes were typically quite small at that time and that allocating more than 2X RAM for swap space did not improve performance. With more than twice RAM for swap, most systems spent more time thrashing than actually performing useful work.

RAM has become an inexpensive commodity and most computers these days have amounts of RAM that extend into tens of gigabytes. Most newer computers have at least 8GB of RAM and up to 64GB. Older computers or mini’s like Raspberry Pi’s have  .25 to 1GB of RAM.

When dealing with computers having huge amounts of RAM, the limiting performance factor for swap space is far lower than the 2X multiplier. The Fedora 28 online Installation Guide, which can be found online at Fedora Installation Guide, defines current thinking about swap space allocation. I have included below some discussion and the table of recommendations from that document.

*Table 1: Recommended system swap space in Fedora 28 documentation*

|  |  |
| --- | --- |
| Amount of system RAM | Recommended swap space |
| less than 2 GB | 2 times the amount of RAM |
| 2 GB - 8 GB | Equal to the amount of RAM |
| 8 GB - 64 GB | 0.5 times the amount of RAM |
| more than 64 GB | workload dependent |
| Cerner Systems | Workload dependent, set by cookbook. |

One consideration is that as the amount of RAM increases, beyond a certain point adding more swap space simply leads to thrashing well before the swap space even comes close to being filled. If you have too little virtual memory while following these recommendations, you should add more RAM, if possible, rather than more swap space. As with all recommendations that affect system performance, use what works best for your specific environment. This will take time and effort to experiment and make changes based on the conditions in your Linux environment.

Adding swap to an LVM disk environment

If your disk setup uses LVM, changing swap space will be fairly easy. Again, this assumes that space is available in the volume group in which the current swap volume is located. By default, the installation procedures for RedHat based Linux in an LVM environment create the swap partition as a logical volume. This makes it easy because you can simply increase the size of the swap volume.

Here are the steps required to increase the amount of swap space in an LVM environment:

* Turn off all swap.
* Increase the size of the logical volume designated for swap.
* Configure the resized volume as swap space.
* Turn on swap.

First, let’s verify that swap exists and is a logical volume using the lvs command (list logical volume).

You can see that the current swap size is 8GB. In this case, we want to add 2GB to this swap volume. First, stop existing swap. You may have to terminate running programs if swap space is in use.

swapoff -a

Now increase the size of the logical volume.

[root@studentvm1 ~]# lvextend -L +2G /dev/mapper/fedora\_studentvm1-swap

  Size of logical volume fedora\_studentvm1/swap changed from 8.00 GiB (2048 extents) to 10.00 GiB (2560 extents).

  Logical volume fedora\_studentvm1/swap successfully resized.

[root@studentvm1 ~]#

Run the mkswap command to make this entire 10GB partition into swap space.

[root@studentvm1 ~]# mkswap /dev/mapper/fedora\_studentvm1-swap

mkswap: /dev/mapper/fedora\_studentvm1-swap: warning: wiping old swap signature.

Setting up swapspace version 1, size = 10 GiB (10737414144 bytes)

no label, UUID=3cc2bee0-e746-4b66-aa2d-1ea15ef1574a

[root@studentvm1 ~]#

Then turn swap back on.

[root@studentvm1 ~]# swapon -a

[root@studentvm1 ~]#

Now verify the new swap space is present with the list block devices command. Again, a reboot is not required.

[root@studentvm1 ~]# lsblk

NAME                                 MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT

sda                                    8:0    0   60G  0 disk

|-sda1                                 8:1    0    1G  0 part /boot

`-sda2                                 8:2    0   59G  0 part

  |-fedora\_studentvm1-pool00\_tmeta   253:0    0    4M  0 lvm

  | `-fedora\_studentvm1-pool00-tpool 253:2    0    2G  0 lvm

  |   |-fedora\_studentvm1-root       253:3    0    2G  0 lvm  /

  |   `-fedora\_studentvm1-pool00     253:6    0    2G  0 lvm

  |-fedora\_studentvm1-pool00\_tdata   253:1    0    2G  0 lvm

  | `-fedora\_studentvm1-pool00-tpool 253:2    0    2G  0 lvm

  |   |-fedora\_studentvm1-root       253:3    0    2G  0 lvm  /

  |   `-fedora\_studentvm1-pool00     253:6    0    2G  0 lvm

  |-fedora\_studentvm1-swap           253:4    0   10G  0 lvm  [SWAP]

  |-fedora\_studentvm1-usr            253:5    0   15G  0 lvm  /usr

  |-fedora\_studentvm1-home           253:7    0    2G  0 lvm  /home

  |-fedora\_studentvm1-var            253:8    0   10G  0 lvm  /var

  `-fedora\_studentvm1-tmp            253:9    0    5G  0 lvm  /tmp

sr0                                   11:0    1 1024M  0 rom

[root@studentvm1 ~]#

You can also use the swapon -s command, or top, free, or any of several other commands to verify this.

[root@studentvm1 ~]# free

              total        used        free      shared  buff/cache   available

Mem:        4038808      382404     2754072        4152      902332     3404184

Swap:      10485756           0    10485756

[root@studentvm1 ~]#

Note that the different commands display or require as input the device special file in different forms. There are a number of ways in which specific devices are accessed in the /dev directory.

Cerner specific variations to the “generic” knowledge

Swap space at Cerner tends to be set correctly for a given application. For those rare instances where it is not we will tweak the swap space or better yet, reduce the work load of the server so it does not swap.

**I’ll reiterate this point. It is best to reduce the workload before increasing swap.**

Typically, Java is the culprit when a system is taxed on memory. There are too many jvm’s running or they are set with too high of a memory limit. If the system is busy and running out of swap try to figure out a way to horizontally scale the application, if that is not possible then increasing the swap size it in order. – remember, once you cross the ~8GB threshold of swap the system may spend too much time swapping to do any work. That said, M+ Chart search has 12GB swap space by default for a 128GB Ram system.

What to do if a system is alarming for swap space

I’ll use Legacy Chart Search since they tend to produce the most alarms in the Millennium Cloud Services space. Again, this is almost always because the system has too many java processes on it or one or more of those processes are not working correctly. Occasionally, one can find Altiris or some other process consuming a lot of swap space if you do then you typically cycle that process to release the memory. When a process like Altiris is consuming swap it is because of two reasons 1. The process has a memory leak (which means it is allocating memory for a sub-process and then not returning it to the free pool) or 2. It is mis-behaving like it gets caught in an endless loop such as it is following a symbolic link that points to itself or points to a parent directory which means if it is trying to find all files it keeps going back to a parent directory then goes to the subdirectory that has a link back to the parent – you get the idea. NOTE: this is a made-up example, not that it is reality.

To find the process that is using swap you can look at top and the “VIRT” column (you may need to change the sort column) or you can run this nifty script in the Lunr documentation.

<https://github.cerner.com/lunr/lunr_tools/blob/master/lunr_scripts/find_swap_usage.sh>

run it with this command:

curl -sSL -k <https://raw.github.cerner.com/lunr/lunr_tools/master/lunr_scripts/find_swap_usage.sh> | bash | sort -nk3

look through it and see which process is using the most swap. Use caution and common sense when cycling processes. Such as “PID=1 swapped 116 KB (init)” Not a good one to cycle. J

Example output:

Overall swap used: 103256 KB

PID=3056 swapped 12 KB (java)

PID=2807 swapped 20 KB (java)

PID=2679 swapped 24 KB (lldpd)

PID=2686 swapped 32 KB (lldpd)

PID=3147 swapped 32 KB (java)

PID=2662 swapped 52 KB (ntpd)

PID=2356 swapped 56 KB (audispd)

PID=2378 swapped 88 KB (portreserve)

PID=2547 swapped 92 KB (nscd)

PID=1 swapped 116 KB (init)

PID=2354 swapped 132 KB (auditd)

PID=2459 swapped 144 KB (irqbalance)

PID=2236 swapped 412 KB (udevd)

PID=2410 swapped 412 KB (udevd)

PID=1041 swapped 416 KB (udevd)

PID=2388 swapped 416 KB (rsyslogd)

PID=2878 swapped 580 KB (java)

PID=2774 swapped 808 KB (qmgr)

PID=2567 swapped 1120 KB (dnsmasq)

PID=3223 swapped 5972 KB (java)

PID=2967 swapped 35484 KB (java)

PID=2580 swapped 56836 KB (java)

This shows two java processes that have done a lot of swapping. One of them is a Solr shard the other is Zabbix.

11:19:41 # ps -ef | grep 2967

tomcat    2967     1  0 Feb06 ?        01:51:35 /usr/lib/jvm/java/bin/java -Djava.util.logging.config.file=/solr/brahe-tomcat/instance\_11/conf/logging.properties -Djava.util.logging.manager=org.apache.juli.ClassLoaderLogManager -Djdk.tls.ephemeralDHKeySize=2048 -Xms512m -Xmx3g -Dcom.sun.management.jmxremote -Dcom.sun.management.jmxremote.ssl=false -Dcom.sun.management.jmxremote.authenticate=false -Dorg.apache.catalina.startup.EXIT\_ON\_INIT\_FAILURE=true -XX:+UseG1GC -XX:+AggressiveOpts -XX:G1HeapWastePercent=2 -XX:G1HeapRegionSize=16m -XX:+ExplicitGCInvokesConcurrent -XX:+UseCompressedOops -XX:+UseCompressedClassPointers -XX:MaxGCPauseMillis=200 -XX:ParallelGCThreads=10 -XX:ConcGCThreads=3 -XX:+UnlockExperimentalVMOptions -XX:-ResizePLAB -XX:+ParallelRefProcEnabled -XX:+AlwaysPreTouch -Djava.security.egd=[file:/dev/urandom](http://file/dev/urandom) -Dcom.sun.management.jmxremote.port=41041 -Dcom.sun.management.jmxremote.rmi.port=41061 -Dignore.endorsed.dirs= -classpath /solr/brahe-tomcat/instance\_11/bin/bootstrap.[jar:/solr/brahe-tomcat/instance\_11/bin/tomcat-juli.jar](http://jar/solr/brahe-tomcat/instance_11/bin/tomcat-juli.jar) -Dcatalina.base=/solr/brahe-tomcat/instance\_11 -Dcatalina.home=/solr/brahe-tomcat/instance\_11 -[Djava.io](http://djava.io/).tmpdir=/solr/brahe-tomcat/instance\_11/temp org.apache.catalina.startup.Bootstrap start

11:20:43 # ps -ef | grep 2580

zabbix    2580     1  0 Feb06 ?        00:39:09 /usr/bin/java -Djdk.home= -Djruby.home=/home/zabbix/ruby/jruby-9.1.9.0 -Djruby.script=jruby -Djruby.shell=/bin/sh -Djffi.boot.library.path=/home/zabbix/ruby/jruby-9.1.9.0/lib/[jni:/home/zabbix/ruby/jruby-9.1.9.0/lib/jni/i386-Linux:/home/zabbix/ruby/jruby-9.1.9.0/lib/jni/ppc64le-Linux:/home/zabbix/ruby/jruby-9.1.9.0/lib/jni/aarch64-Linux:/home/zabbix/ruby/jruby-9.1.9.0/lib/jni/ppc64-Linux:/home/zabbix/ruby/jruby-9.1.9.0/lib/jni/x86\_64-Linux:/home/zabbix/ruby/jruby-9.1.9.0/lib/jni/arm-Linux](http://jni/home/zabbix/ruby/jruby-9.1.9.0/lib/jni/i386-Linux:/home/zabbix/ruby/jruby-9.1.9.0/lib/jni/ppc64le-Linux:/home/zabbix/ruby/jruby-9.1.9.0/lib/jni/aarch64-Linux:/home/zabbix/ruby/jruby-9.1.9.0/lib/jni/ppc64-Linux:/home/zabbix/ruby/jruby-9.1.9.0/lib/jni/x86_64-Linux:/home/zabbix/ruby/jruby-9.1.9.0/lib/jni/arm-Linux) -Xmx500m -Xss2048k -Dsun.java.command=org.jruby.Main -cp  -Xbootclasspath/a:/home/zabbix/ruby/jruby-9.1.9.0/lib/jruby.jar -Djava.security.egd=[file:/dev/urandom](http://file/dev/urandom) org/jruby/Main /usr/local/etc/zabbix/jmx-get2.rb -p /var/run/zabbix/jmx-get.pid -s /var/run/zabbix/jmx-get

If swap on this server was excessive then cycling Zabbix would be in order. If further action is warranted then we look at cycling the solr shard.

These two examples come from a “M+ Chart Search” server and cycling the process can only be done after ensuring there is a second copy of the shard in a “healthy” state.