Linux SWAP

How Linux uses RAM (very simplified)

Each application can use some of your memory. Linux uses all otherwise unoccupied memory (except for the last few Mb) as “cache”. This includes the page cache, inode caches, etc. This is a good thing - it helps speed things up heaps. Both writing to disk and reading from disk can be sped up immensely by cache.

Ideally, you have enough memory for all your applications, and you still have several hundred Mb left for cache. In this situation, as long as your applications don't increase their memory use and the system isn't putting too much pressure on the cache, there is no need for any swap.

Once applications claim more RAM, it simply goes into some of the space that was used by cache, shrinking the cache. De-allocating cache is cheap and easy enough that it is simply done in real time - everything that sits in the cache is either just a second copy of something that's already on disk, so can just be deallocated instantly, or it's something that we would have had to flush to disk within the next few seconds anyway, thus there is zero performance hit in re-allocating cache to applications.

So, when someone refers to “free” RAM, this may or may not include cache, since cache will only occupy “free” RAM. This is not a situation that is specific to Linux - all modern operating systems work this way. The different operating systems might just report free RAM differently: some include the cache as part of what they consider “free” and some may not.

When you talk about free RAM, it's a lot more meaningful to include cache, because it practically is free - it's available should any application request it. On Linux, the free command reports it both ways - the first line includes cache in the used RAM column, and the second line includes cache (and buffers) in the free column.

How Linux uses swap (even more simplified)

Once you have used up enough memory that there is not enough left for a smooth-running cache, Linux may re-allocate some unused application memory from RAM to swap in order to regain some memory for cache.

It doesn't do this according to a definite cutoff though. It's not like you reach a certain percentage of allocation then Linux starts swapping. It has a rather “fuzzy” algorithm. It takes a lot of things into account, which can best be described by “how much pressure is there for memory allocation”. If there is a lot of “pressure” to allocate new memory, then it will increase the chances some will be swapped to make more room. If there is less “pressure” then it will decrease these chances.

Your system has a “swappiness” setting which helps you tweak how this “pressure” is calculated. It's normally not recommended to alter this at all, and I would certainly never recommend you alter it. Swapping is overall a very good thing - any occasional performance penalties are intended to be offset by a gain in overall system responsiveness and stability for a wide range of tasks. If you reduce the swappiness, you let the amount of cache memory shrink a little bit more than it would otherwise, even when it may really be useful. You therefore risk slowing down your computer in general, because

there is less cache, while memory is being taken up by applications that aren't even using it. Whether this is a good enough trade-off for whatever specific problem you're having with swapping is up to you. If you go further than this and actually disable swapping, you risk system instability in the event that the system runs out of memory for processes.

What is happening when the system is bogged down and swapping heavily?

A lot of the time people will look at their system that is thrashing the disk heavily and using a lot of swap space and blame swapping for it. That's the wrong approach to take. If swapping ever reaches this extreme, it means that your system is extremely low in memory and the swapping is the only thing keeping it from crashing or killing processes randomly. Without swapping, in this situation, processes will crash and die. The swapping is a symptom of a deeper problem. In a system with enough memory for all its tasks, swapping only ensures that memory is utilised in efficient ways, dealing out memory to the cache over dormant processes where it sees it will be worth it. In a system where swapping is relentlessly thrashing the disk, it's not swapping's fault.

When choosing what is to be swapped to disk, the system tries to pick memory that is not actually being used - read to or written from. It has a pretty simple algorithm for calculating this that chooses well most of the time.

In a desktop system that is used sporadically, there is a well-known situation in which suddenly trying to use process that has been sitting idle for quite some time results in a delay where the memory for the idle process has to be swapped back from disk to memory. This can result in a disk based bottleneck while it happens that slows down the system in general. It's typical when, for example, you've let your computer do tasks such as backups, torrents etc overnight, then suddenly in the morning you re-open your minimised browser window and try to use it again. This is a result of the system being unable to predict that a process that's been idle for a long period will suddenly be required once more. It's a side effect of the swapping system. Turn off swap and this initial sluggishness after the backup/virus scan may not happen, but the system may run a little bit slower all day long as there is more pressure on the cache size, and you lose the added protection against out-of-memory conditions which can cause processes to crash or be killed.

Note: none of this is a situation or problem that's limited to Linux. All modern operating system will operate this way.

The most effective way to solve sluggishness associated with memory swapping is to increase the physical memory size (or run less memory-hungry applications). Turning off swap or dialling it back is one of the least effective ways to solve it because swap is only a coping mechanism for the low memory situation.

Can/should swap be disabled on a system that has lots of RAM anyway?

If you have a system where you have a huge amount of RAM (at time of writing, 8GB is a huge amount for a typical Linux distro), then you will very rarely ever hit a situation where swap is needed at all. There is no need to explicitly disable swap at all or adjust its swappiness - your system will

realise that there is plenty of RAM for both applications and cache and will never make the decision to swap.

You could disable swap or decrease swappiness, and it will very probably work just fine, but in all normal situations it will make absolutely zero difference because your system would not have decided to swap anyway due to having plenty of memory. The only situations in which it would make a difference would be in the unlikely situation the system finds itself running out of memory and consequently the cache system is getting hampered, and it's in this type of situation where you would want swap most. So you can safely leave swap on its normal settings for added peace of mind without it having a negative effect when you have plenty of memory.

But how can swap speed up my system? Doesn't swapping slow things down?

The act of transferring data from RAM to swap is a slow operation, but it's only taken when the kernel is pretty sure the overall benefit will outweigh this. For example, if your application memory has risen to the point that you have almost no cache left and your I/O is very inefficient because of this, you can actually get a lot more speed out of your system by freeing up some memory, even after the initial expense of swapping data in order to free it up.

It's also a last resort should your applications actually request more memory than you actually have. In this case, swapping is necessary to prevent an out-of-memory situation which will often result in an application crashing or having to be forcibly killed. There's no getting around the fact that your system will probably become dog-slow when this happens, but console yourself by realising that were it not for swap, your application would likely have crashed instead, potentially resulting in data loss.

Swapping is only associated with times where your system is performing poorly because it happens at times when you are running out of usable RAM, which would have slowed your system down (maybe even crashed applications) even if you didn't have swap. So to simplify things, swapping happens because your system is becoming bogged down, rather than the other way around - and in some cases it can save the day.

Once data is in swap, when does it come out again?

Transferring data out of swap is (for traditional hard disks, at least) just as time-consuming as putting it in there. So understandably, your kernel will be just as reluctant to remove data from swap, especially if it's not actually being used (ie read from or written to). If you have data in swap and it's not being used, then it's actually a good thing that it remains in swap, since it leaves more memory for other things that are being used, potentially speeding up your system.

Server vs desktop difference

On usual desktop, you have 4-5 active tasks that consume 50-60% of memory. If you set swappiness to 60, then about 1/4-1/3 of the ACTIVE task pages will be swapped out. That means, for every task change, for every new tab you opened, for every JS execution, there will be a swapping process.

The solution is to set swappiness to 10. By practical observations, this causes the system to give up disk io cache (that plays little to no role on desktop, as read/write cache is virtually not used at all. Unless you constantly copying LARGE files) instead of pushing anything into swap. In practice, that means system will refuse to swap pages, cutting io cache instead, unless it hits 90% used memory. And that in turn means a smooth, swapless, fast desktop experience.

On the file server, however, I would set swappiness to 60 or even more, because server does not have huge active foreground tasks that must be kept in the memory as whole, but rather a lot of smaller processes that are either working or sleeping, and not really changing their state immediately. Instead, server often serves (pardon) the exact same data to clients, making disk io caches much more valueable. So on the server, it is much better to swap out the sleeping processes, freeing memory space for disk cache requests.

On desktops, however, this exact setting leads to swapping out blocks of memory of REAL applications, that near constantly modify or access this data.

Oddly enough, browsers often reserve large chunks of memory, that they constantly modify. When such chunks are swapped out, it takes a while if they are requested back - and at the same time, browser goes forth updating its caches. Which causes huge latencies. In practice, you will be sitting 2 minutes waiting for the single web page in a new tab to load.

Desktop does not really care about disk io, because desktop rarely reads and writes cacheable repeating big portions of data. Cutting on disk io in order to just prevent swaping so much as possible is much more favorible for desktop, than to have 30% of memory reserved for disk cache with 30% of RAM (full of blocks belonging to actively used applications) swapped out.