Tales from responsivenessland: why Linux feels slow, and how to fix that

by Rudd-O (https://rudd-o.com/author/RuddO) — published 11 years ago, last modified 5 years ago

Desktop performance on Linux computers has been a hot-button issue of late, and a source of longstanding fights among the Linux developers. Today, I want to show you how I boosted (and you can boost) desktop performance dramatically.

The problem

What's the number one cause of desktop performance problems?

Let me answer that question with another set of questions:

- Have you noticed how slow your computer is after after unzipping a big file?
- Perhaps you can also notice this effect overnight as well -- you return to your computer, and every open application takes a while before they behave snappily again.
- Do you use Amarok? Then you'll notice this effect after Amarok has finished scanning your music collection.

This problem happens all the time. In fact, I would be willing to bet this is the cause most people find Linux to be slow.

The source of the problem

Here's my chief finding: desktop performance has very little to do with schedulers and real performance. It has everything to do with disk caching and *perceived* performance.

Now, bear with me for a second, because what I'm about to explain will let you understand the following. Based on my limited understanding of the Linux kernel (please correct me if I'm wrong, especially about the terminology), there are two distinct cache types:

- a block cache: caches block device contents
- a inode/dentry cache: one layer above the block cache, caches directory entries and other filesystem-related things that cost more to look up than just block device contents

Problem number 1 is that the block cache competes with applications for RAM. Problem number two is that the block cache competes with the inode/dentry cache.

On a normal desktop Linux computer, when you're short of RAM, Linux will aggressively try to free memory to enlarge the caches. It usually does this by paging out unused portions of applications to disk. This is not good.

Why? Since disks are slow, the following consequences follow:

Let's get technical (skip this heading if you're uncomfortable with technicalities)

For desktop performance, where workloads are unpredictable, responsiveness needs dictate that applications' working sets should be in core as much as possible, because the probability of major page faults in desktop loads is much larger than on server loads.

Let me rephrase that: **Screw data. Prioritize code.**

In other words: it's OK if your application stalls a bit while reading non-cached *data* from disk, because that isn't going to make the application seem frozen for a couple of seconds. It's *not* OK if the application freezes because it needs to execute code that was paged out to disk.

All in all, how Linux uses your RAM matters a lot for desktop performance

You can see where this is going: if you have 1 GB of RAM (or less) you unzip a 300 MB archive, and the Linux kernel will swap out portions of your applications to disk in an effort to accomodate both the 300 MB file and the blocks of the uncompressed files about to be written on disk. It will also shrink the inode/dentry cache.

Linux, of course, thinks it's doing a good thing -- the archive uncompresses, in effect, rather quickly because of this. You, of course, want to kick Linux in the nuts because while (and after) the uncompress took place, starting an application or opening a new browser window suddenly takes ages, and your open applications "blank out" for a couple of seconds before reacting to your clicks and keystrokes.

Think about it: Linux is catering to the needs of the uncompression program, and you are actively telling your computer you want to do other stuff while it happens. Linux starts to scramble between reading the archive, writing data to disk, and bringing the portions of the applications you're using "back from the dead" (from swap). Guess who loses? If you guessed "the applications" or "you", you'd have been right.

Even worse: suppose you want to browse your home directory while all of this happens. In addition to the fact that your file manager is probably half-on-disk by now, you've got the problem that Linux needs to re-read your home directory, since the inode/dentry cache has been squashed.

Not good. I posit that the default settings may be very good for workloads with many repeating tasks, but definitely *un*good for desktop situations, when applications (and code paths) are diverse and respond only to the whims of the user.

What we want is perceived performance

Let's take a file manager for example. Let's focus on Konqueror (it's a nice case study, and it's a nice file manager). Suppose I hit the home button on my panel, which shows one of the (usually hidden and prelaunched) Konqueror instances and prompts it to browse my home directory.

If you have lots of RAM, it won't be a problem -- both Konqueror and the contents of your home directory will be in memory, so it'll be blindingly fast. But if you're rather short of memory, it's a different matter -- what happens next determines whether you feel your computer slow or fast.

If Konqueror has been paged out, it will appear to be frozen (or take longer to "start up") for a couple of seconds, until Linux has paged necessary code paths in. If, on the contrary, my home directory has been evicted from the RAM cache, Konqueror will show up instantly and be responsive, while the home directory loads.

I'd much rather wait for the directory display than have to wait for Konq to unfreeze because it was paged out. The difference is that in the first scenario, I can close the window, use the menus, navigate among the window controls, change the URL, abort the operation; in the second case, I'm screwed until Linux decides to fully pagein whatever Konq needs.

That's what I'm arguing for -- perceived performance, not throughput. It matters to me that I can manipulate my file manager half-a-second after I've hit the home button. It doesn't matter to me that, because of this preference, the home directory actually takes one second longer to finish displaying.

Variations of this pattern can be found everywhere: in file open dialogs, in multimedia applications with collection managers, basically everywhere an operation requires some sort of progress report.

The solution

There are two distinct and complementary measures we'll take to solve this problem. Keep reading to find out about them.

Tuning swappiness to prevent impromptu RAM hijacking

Swappiness is the name Linux kernel developers gave to the preference between paging applications out to disk and (in practice) shrinking caches. If it's close to 0, Linux will prefer to keep applications in RAM and not grow the caches. If it's close to 100, Linux will prefer to swap applications out, and enlarge the caches as much as possible. The default is a healthy 60.

The irony of this preference is that, in fact, paging an unused application out generally produces a net performance *increment*, since the cache really helps a lot when it's needed -- but this net performance increment translates to a *net drop in perceived* performance, since you usually don't care whether a file uncompresses a few seconds later, but you do care (a lot) when your applications don't respond instantaneously.

On a desktop computer, you want swappiness to be as close to zero as possible. The reason you want to do this (even though it might hurt actual performance) is because this will immunize your computer to memory shortages caused by temporary big file manipulations (think copying a big video file to another disk). The cache will still be as big as possible, but it won't displace running applications.

Don't believe me? If you have two gigabytes to spare on your / partition, then run the following commands in order:

```
sync
echo 3 > /proc/sys/vm/drop_caches
dd if=/dev/zero of=/tmp/testfile count=1 bs=900M
find / > /dev/null
cp /tmp/testfile /tmp/testfile2
```

Don't do anything else while they run. Yes, they will take a while. Be patient. What this does is rather simple: creates a big 900 MB file, attempts to find all files on your file system, then copies the file one more time.

Finished? Now do something with your computer. Un-minimize your minimized applications, do something relating to your everyday work...

What happened? Excruciatingly slow, right? Did you notice how applications were hung for a while, while the hard disk light blinked frantically?

OK, let's tune swappiness. As root, run:

```
sysctl -w vm.swappiness=1
```

and retry the commands above. Big difference, right?

That's because, with swappiness turned down, the Linux kernel no longer attempts to enlarge the cache by paging applications out. Not unless you're experiencing an extremely high memory shortage.

To make the change permanent, write `vm.swappiness=1` on your `/etc/sysctl.conf` file.

Filesystem caches are more important than other caches

We've already established that the filesystem cache is important because, without it, file browsing goes extremely slowly as well. Now we'll learn how to tell Linux that we want it to prefer inode/dentry cache to other caches.

Try this:

```
sync
echo 3 > /proc/sys/vm/drop_caches
dd if=/dev/zero of=/tmp/testfile count=1 bs=900M

sysctl -w vm.vfs_cache_pressure=100
find / > /dev/null
cp /tmp/testfile /tmp/testfile2
time find / > /dev/null

sysctl -w vm.vfs_cache_pressure=50
find /  > /dev/null
cp /tmp/testfile2 /tmp/testfile3
time find / > /dev/null

rm -f /tmp/testfile /tmp/testfile2 /tmp/testfile3
```

This series of commands does basically what the former did, only this time it attempts to measure the time it takes to find all files on your disk -- after creating big files. It does so twice -- once with the default settings for cache preference, and once with settings heavily tilted to favor inode/dentry cache.

Check the output. Did the second run take less time? It definitely cut times in more than half on my desktop computer. When this knob is closer to 1, the kernel will prefer to protect the inode/dentry cache by freeing block cache first.

To make the change permanent, put `vm.vfs_cache_pressure=50` on your `/etc/sysctl.conf` file.

Please experiment with this value. Values close to 100 provide no gain. Values close to zero can cause huge swap activity during big filesystem scans.

That's it

Desktop Linux is quite good. However, there are certain situations in which it could improve. Fortunately, Linux comes right out of the box with the things you need to tune it for desktop performance.

Don't miss out on the incredible performance gains you'll feel. And feel free to subscribe to this site, if you want to continue learning.

A shameless plug

If you want to find out how your applications are behaving with regard to actual disk usage, let me recommend iogrind (http://live.gnome.org/iogrind) -- one of Michael Meeks' latest tools in the fight against the slowskys, a sort of Filelight for disk accesses. And if you have extensive ReiserFS knowledge, please help!