

When Memory Serves You: Using ramfs and tmpfs

Ken Hess

If your read/write performance isn't keeping up with your needs, the least expensive and least time-consuming fix is to place those heavily used files into RAM. Reading and writing to RAM is significantly faster than when using disk-based filesystems. Disk I/O-sensitive data transfers, like those involving databases, reap extreme benefits from moving to RAM-based filesystems.

Why RAM? RAM is fast. It operates in nanosecond-level access times whereas the fastest disk operates in millisecond-level access times. RAM doesn't spin. Mechanical drives spin, which means their read/write and seek speeds are significantly slower than their RAM-based equivalents. DDR3 RAM, for example, moves data in and out at solder-melting peak rates exceeding 10GB/s. Even Hitachi's hottest 15,000 RPM UltraStar disk, transfers data at a sluggish 119MB/s to 198MB/s sustained and 600MB/s max. RAM has longer MTBF (Mean Time Between Failures). Since RAM isn't mechanical, it doesn't enjoy the high failure rates of spinning disks, therefore giving it a life expectancy well beyond that of a typical disk drive.

You have two performance-boosting options for RAM-based filesystems: tmpfs and ramfs. Let's learn how to set up a RAM-based filesystem, some general usage tips and how best to avoid common trouble spots along the way.

ramfs

Tmpfs and ramfs handle their jobs very differently. Ramfs can only use system memory, doesn't reveal itself in a `df -h` listing, has no size limits and provides no error messages when any optional limits are set. It might seem contradictory to say that ramfs has no size limitations and you don't see errors messages when optional limits are set, but it isn't. You can set a limit on a ramfs filesystem but you won't receive a warning when you exceed that limit nor will the system prevent you from exceeding that limit.

For example, the standard syntax for mounting a new filesystem is as follows:

```
# mount -t fs_type device mount_dir
```

The syntax for setting up a 200MB ramfs filesystem for a database in the directory, `/opt/data`:

```
# mount -t ramfs -o size=200m ramfs /opt/data
```

As previously stated, this filesystem won't show up in a `df -h` listing. The only way to see the ramfs is to use the `mount` command.

```
# mount
```

```
/dev/mapper/VolGroup00-LogVol100 on / type ext3 (rw)
proc on /proc type proc (rw)
sysfs on /sys type sysfs (rw)
devpts on /dev/pts type devpts (rw,gid=5,mode=620)
/dev/sdal on /boot type ext3 (rw)
none on /proc/sys/fs/binfmt_misc type binfmt_misc (rw)
sunrpc on /var/lib/nfs/rpc_pipefs type rpc_pipefs (rw)
ramfs on /opt/db type ramfs (rw,size=200m)
```

If you attempt to write more than 200MB of data to this filesystem, the write will proceed and you will receive no warnings about the filesystem's size limits. The 200MB limit is superfluous and has no effect on the actual size of the ramfs or how much data you can write to it. This is a major disadvantage of ramfs.

When Memory Serves You: Using ramfs and tmpfs

Ken Hess

Since system administrators tend to be an overworked group, they sometimes leave stray files on a filesystem after installing patches, after installing software or after the infinite amount of testing that goes on for some systems. A ramfs would answer this problem effectively. All files deemed non-essential for long-term storage could reside in a temporary storage area (ramfs) until they're no longer needed. Unmount the ramfs and the system returns to normal without the untidiness that follows these periodic actions.

tmpfs

System administrators find that for tasks involving real data, tmpfs is a superior choice over ramfs. Ramfs has a definite fixed size, it can use RAM or swap for storage and displays errors when you exceed its specified size. The syntax resembles that of the ramfs.

```
# mount -t tmpfs -o size=200m tmpfs /opt/data
```

A `df -h` shows the mounted tmpfs as it would any other mounted filesystem.

```
# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/mapper/VolGroup00-LogVol100
                360G  225G  117G   66% /
/dev/sda1       99M   25M   70M   27% /boot
tmpfs           200M    0   200M    0% /opt/data
```

And the mount command yields:

```
# mount
/dev/mapper/VolGroup00-LogVol100 on / type ext3 (rw)
proc on /proc type proc (rw)
sysfs on /sys type sysfs (rw)
devpts on /dev/pts type devpts (rw,gid=5,mode=620)
/dev/sda1 on /boot type ext3 (rw)
none on /proc/sys/fs/binfmt_misc type binfmt_misc (rw)
sunrpc on /var/lib/nfs/rpc_pipefs type rpc_pipefs (rw)
tmpfs on /opt/db type tmpfs (rw,size=200m)
```

When you exceed the limits of a mounted tmpfs, you'll receive a "No space left on device" nastygram from the system informing you that your filesystem is full. The tmpfs behaves like a disk-based filesystem except for its volatile nature. You may insert an entry into the `/etc/fstab` for either of these filesystem types to re-establish the volatile mount after a reboot.

Both ramfs and tmpfs are volatile storage. In other words, if your system crashes, reboots or shuts down for any reason, the data contained in either filesystem type is deleted. For this reason alone, you should make periodic dumps of the data from such a volatile filesystem to a permanent storage location. Remember the adage, "Safe, fast, cheap; choose any two." Using a RAM-based filesystem is fast and cheap but not safe.