Linux Page Cache Basics

Under Linux, the **Page Cache** accelerates many accesses to files on non volatile storage. This happens because, when it first reads from or writes to data media like hard drives, Linux also stores data in **unused areas of memory**, which acts as a cache. If this data is read again later, it can be quickly read from this cache in memory. This article will supply valuable background information about this page cache.

Contents [hide]

Page Cache or Buffer Cache

The term, Buffer Cache, is often used for the Page Cache. Linux kernels up to version 2.2 had both a Page Cache as well as a Buffer Cache. As of the 2.4 kernel, these two caches have been combined. Today, there is only one cache, the Page Cache.[1]

Functional Approach

Memory Usage

Under Linux, the number of megabytes of main memory currently used for the page cache is indicated in the **Cached** column of the report produced by the `free -m` command.

```
[root@testserver ~]# free -m
             total       used       free     shared    buffers     cached
Mem:         15976      15195        781          0        167       9153
-/+ buffers/cache:       5874      10102
Swap:         2000          0       1999
[root@testserver ~]#
```

Writing

If data is written, it is first written to the Page Cache and managed as one of its *dirty pages*. *Dirty* means that the data is stored in the Page Cache, but needs to be written to the underlying storage device first. The content of these *dirty pages* is periodically transferred (as well as with the system calls sync or fsync) to the underlying storage device. The system may, in this last instance, be a RAID controller or the hard disk directly.

The following example will show the creation of a 10-megabyte file, which will first be written to the Page Cache. The number of dirty pages will be increased by doing so, until they have been written to the underlying solid-state drive (SSD), in this case manually in response to the sync command.

```
wfischer@pc:~$ dd if=/dev/zero of=testfile.txt bs=1M count=10
10+0 records in
10+0 records out
10485760 bytes (10 MB) copied, 0,0121043 s, 866 MB/s
wfischer@pc:~$ cat /proc/meminfo | grep Dirty
Dirty:             10260 kB
wfischer@pc:~$ sync
```

```
wfischer@pc:~$ cat /proc/meminfo | grep Dirty
Dirty:                 0 kB
```

Up to Version 2.6.31 of the Kernel: pdflush

Up to and including the 2.6.31 version of the Linux kernel, the pdflush threads ensured that dirty pages were periodically written to the underlying storage device.

As of Version 2.6.32: per-backing-device based writeback

Since pdflush had several performance disadvantages, Jens Axboe developed a new, more effective writeback mechanism for Linux Kernel version 2.6.32. [2]

This approach provides threads for each device, as the following example of a computer with an SSD (/dev/sda) and a hard disk (/dev/sdb) shows.

```
root@pc:~# ls -l /dev/sda
brw-rw---- 1 root disk 8, 0 2011-09-01 10:36 /dev/sda
root@pc:~# ls -l /dev/sdb
brw-rw---- 1 root disk 8, 16 2011-09-01 10:36 /dev/sdb
root@pc:~# ps -eaf | grep -i flush
root       935     2  0 10:36 ?        00:00:00 [flush-8:0]
root       936     2  0 10:36 ?        00:00:00 [flush-8:16]
```

Reading

File blocks are written to the Page Cache not just during writing, but also when reading files. For example, when you read a 100-megabyte file twice, once after the other, the second access will be quicker, because the file blocks come directly from the Page Cache in memory and do not have to be read from the hard disk again. The following example shows that the size of the Page Cache has increased after a good, 200-megabytes video has been played.

```
user@adminpc:~$ free -m
             total       used       free     shared    buffers     cached
Mem:          3884       1812       2071          0         60       1328
-/+ buffers/cache:        424       3459
Swap:         1956          0       1956
user@adminpc:~$ vlc video.avi
[...]
```

```
user@adminpc:~$ free -m
             total       used       free     shared    buffers     cached
Mem:          3884       2056       1827          0         60       1566
-/+ buffers/cache:        429       3454
Swap:         1956          0       1956
user@adminpc:~$
```

If Linux needs more memory for normal applications than is currently available, areas of the Page Cache that are no longer in use will be automatically deleted.

Optimizing the Page Cache

Automatically storing file blocks in the Page Cache is generally quite advantageous. Some data, such as log files or MySQL dump files, are often no longer needed once they have been written. Therefore, such file blocks often use space in the Page Cache unnecessarily. Other file blocks, whose storage would be more advantageous, might be prematurely deleted from the Page Cache by newer log files or MySQL.[3]

Periodic execution of logrotate with gzip compression can help with log files, somewhat. When a 500-megabyte log file is compressed into 10 megabytes by logrotate and gzip, the original log file becomes invalid along with its cache space. 490 megabytes in the Page Cache will then become available by doing so. The danger of a continuously growing log file displacing more useful file blocks from the Page Cache is reduced thereby.

Therefore, it is completely reasonable, if some applications would normally not store certain files and file blocks in the cache. There is already such a patch available for rsync.[4]

References

1. ↑ The Buffer Cache (Section 15.3) page 348, Linux-Kernel Manual: Guidelines for the Design and Implementation of Kernel 2.6, Robert Love, Addison-Wesley, 2005
2. ↑ Linux 2.6. – 32 Per-backing-device based writeback (kernelnewbies.org)
3. ↑ Clearing The Linux Buffer Cache (blog.straylightrun.net, 03.12.2009)
4. ↑ Improving Linux performance by preserving Buffer Cache State (insights.oetiker.ch)

Additional Information

- Page Cache, the Affair Between Memory and Files (Blog)
- Linux buffer cache state (Blog)
- The Linux Page Cache and pdflush: Theory of Operation and Tuning for Write-Heavy Loads (Last update 8/08/2007)
- drop_caches (linuxinsight.com)
- Examining Linux 2.6 Page-Cache Performance (linuxinsight.com)
- Page cache (en.wikipedia.org)

Your feedback is welcome Printable version

Category: Linux Basics