Linux中的Page cache和Buffer cache详解

1、内存情况

在讲解Linux内存管理时已经提到，**当你在Linux下频繁存取文件后**，即使系统上没有运行许多程序，也**会占用大量的物理内存。**这是因为当你读写文件的时候，Linux内核为了提高读写的性能和速度，会**将文件在内存中进行缓存，这部分内存就是Cache Memory(缓存内存)。**即使你的程序运行结束后，**Cache Memory 也不会自动释放**，这就会导致你的Linux系统在频繁读写文件后，可用物理内存会很少。

可用 free 命令查看cache的大小：

1.1 第一行Men：

1. Mem：表示物理内存统计
2. total： 表示物理内存总量(total = used + free)
3. used：表示总计分配给缓存（包含buffers 与cache ）使用的数量，但其中可能部分缓存并未实际使用。
4. free： 未被分配的内存。
5. shared：共享内存。
6. buffers：系统分配但未被使用的buffers 数量。
7. cached：系统分配但未被使用的cache 数量。

1.2 第二行-/+ buffers/cache：表示物理内存的缓存统计

1. used2：也就是第一行中的used – buffers-cached 也是实际使用的内存总量。 //used2为第二行
2. free2= buffers1 + cached1 + free1 //free2为第二行、buffers1等为第一行
3. free2：未被使用的buffers 与cache 和未被分配的内存之和，这就是系统当前实际可用内存。

1.3 第三行Swap 表示硬盘上交换分区的使用情况。

1. used——已使用
   free——未使用

为了提高磁盘存取效率, Linux做了一些精心的设计, 除了对dentry进行缓存（用于VFS，加速文件路径名到inode的转换）， 还采取了两种主要Cache方式：Buffer Cache和Page Cache。前者针对磁盘块的读写，后者针对文件 inode 的读写。这些Cache有效缩短了 I/O系统调用（比如read,write,getdents）的时间。

2 内存类别

2.1 Buffer cache（块缓存）

块缓冲，通常1K，对应于一个磁盘块，用于减少磁盘IO

由物理内存分配，通常空闲内存全是bufferCache

应用层面，不直接与BufferCache交互，而是与PageCache交互（见下）

读文件：

直接从bufferCache中读取

写文件：

方法一，写bufferCache，后写磁盘

方法二，写bufferCache，后台程序合并写磁盘

Buffer cache 也叫块缓冲，**是对物理磁盘上的一个磁盘块进行的缓冲**，其大小为通常为1k，磁盘块也是磁盘的组织单位。**设立buffer cache的目的是为在程序多次访问同一磁盘块时，减少访问时间**。系统将磁盘块首先读入buffer cache 如果cache空间不够时，会通过一定的策略将一些过时或多次未被访问的buffer cache清空。程序在下一次访问磁盘时首先查看是否在buffer cache找到所需块，命中可减少访问磁盘时间。不命中时需重新读入buffer cache。对buffer cache 的写分为两种，一是直接写，这是程序在写buffer cache后也写磁盘，要读时从buffer cache 上读，二是后台写，程序在写完buffer cache 后并不立即写磁盘，因为有可能程序在很短时间内又需要写文件，如

果直接写，就需多次写磁盘了。这样效率很低，而是过一段时间后由后台写，减少了多次访磁盘 的时间。

Buffer cache 是由物理内存分配，linux系统为提高内存使用率，会将空闲内存全分给buffer cache ，当其他程序需要更多内存时，系统会减少cahce大小。

2.2 Page cache（页面缓存）

页缓冲/文件缓冲，通常4K，**由若干个磁盘块组成（物理上不一定连续，HY：Page cache在内存中，别误解了），也即由若干个bufferCache组成**

读文件：

可能不连续的几个磁盘块--->bufferCache--->pageCache--->应用程序进程空间

写文件：

pageCache--->bufferCache--->磁盘

Page cache 也叫页缓冲或文件缓冲，是由好几个磁盘块构成（HY:别误解了），大小通常为4k，在64位系统上为8k，构成的几个磁盘块在物理磁盘上不一定连续，文件的组织单位为一页，也就是一个page cache大小，文件读取是由外存上不连续的几个磁盘块，到buffer cache，然后组成page cache，然后供给应用程序。

Page cache在linux读写文件时，它用于缓存文件的逻辑内容，从而加快对磁盘上映像和数据的访问。具体说是加速对文件内容的访问，buffer cache缓存文件的具体内容——物理磁盘上的磁盘块，这是加速对磁盘的访问。

2.3 Buffer page（缓冲页）

如果内核需要单独访问一个块，就会涉及到buffer page，并会检查对应的buffer head。

2.4 Swap space（交换空间）

Swap space 交换空间，是**虚拟内存的表现形式。**系统为了应付一些需要大量内存的应用，而**将磁盘上的空间做内存使用**，当物理内存不够用时，将其中一些暂时不需的数据交换 到交换空间，也叫交换文件或页面文件中。做虚拟内存的好处是让进程以为好像可以访问整个系统物理内存。因为在一个进程访问数据时，其他进程的数据会被交换 到交换空间中。

2.5 Swap cache（交换缓存）

swapcached，它表示交换缓存的大小。**Page cache是磁盘数据在内存中的缓存，而swap cache则是交换分区在内存中的临时缓存。**

2.6 Memory mapping（内存映射）

内核有两种类型的内存映射：**共享型**(shared)和**私有型**(private)。

私有型是当进程为了只读文件，而不写文件时使用，这时，私有映射更加高效。 但是，任何对私有映射页的写操作都会导致内核停止映射该文件中的页。所以，写操作既不会改变磁盘上的文件，对访问该文件的其它进程也是不可见的。

共享内存中的页通常都位于page cache，私有内存映射只要没有修改，也位于page cache。当进程试图修改一个私有映射内存页时，内核就把该页进行复制，并在页表中用复制的页替换原来的页。由于修改了页表，尽管原来的页仍然在 page cache，但是已经不再属于该内存映射。而新复制的页也不会插入page cache，而是添加到匿名页反向映射数据结构。

3 区别

3.1 Page cache和Buffer cache的区别

磁盘的操作有逻辑级（文件系统）和物理级（磁盘块），这两种Cache就是分别缓存逻辑和物理级数据的。

假设我们通过文件系统操作文件，那么文件将被缓存到Page Cache，如果需要刷新文件的时候，Page Cache将交给Buffer Cache去完成，因为Buffer Cache就是缓存磁盘块的。

也就是说，直接去操作文件，那就是Page Cache区缓存，用dd等命令直接操作磁盘块，就是Buffer Cache缓存的东西。

Page cache实际上是针对文件系统的，是文件的缓存，在文件层面上的数据会缓存到page cache。文件的逻辑层需要映射到实际的物理磁盘，这种映射关系由文件系统来完成。当page cache的数据需要刷新时，page cache中的数据交给buffer cache，但是这种处理在2.6版本的内核之后就变的很简单了，没有真正意义上的cache操作。

Buffer cache是针对磁盘块的缓存，也就是在没有文件系统的情况下，直接对磁盘进行操作的数据会缓存到buffer cache中，例如，文件系统的元数据都会缓存到buffer cache中。

简单说来，**page cache用来缓存文件数据，buffer cache用来缓存磁盘数据。在有文件系统的情况下，对文件操作，那么数据会缓存到page cache，如果直接采用dd等工具对磁盘进行读写，那么数据会缓存到buffer cache。**

Buffer(Buffer Cache)以块形式缓冲了块设备的操作，定时或手动的同步到硬盘，它是为了缓冲写操作然后一次性将很多改动写入硬盘，避免频繁写硬盘，提高写入效率。

Cache(Page Cache)以页面形式缓存了文件系统的文件，给需要使用的程序读取，它是为了给读操作提供缓冲，避免频繁读硬盘，提高读取效率。

buffer cache与page cache

1、page cache用于优化文件系统的I/O，buffer cache用于优化磁盘的I/O。

一般来说 page cache缓存的是file，很重要。buffer cache缓存的是 inode，dentry等内容，相对不那么重要。
page cache常用于读操作的时候，将常常读取的file缓存起来；buffer cache则是将要写入磁盘的内容缓冲（零存整取）。

以下文章引自Quora：
the page cache caches pages of files to optimize file I/O.The buffer cache caches disk blocks to optimize block I/O.

page cache 缓存了file，以优化文件的I/O，buffer cache 缓冲了磁盘 block，以优化磁盘I/O。

Prior to Linux kernel version 2.4,the two caches were distinct : File were in the page cache, disk block were in the buffer cache. Given that most files are represented by a filesystem on a disk ,data was represented twice, once in each of the caches. Many Unix systems follow a similar pattern.

在Linux kernel 2.4之前，两个cache是有区别的：文件存在page cache内，磁盘块存在buffer cache内。大部分磁盘里的文件呈现到文件系统过程中，数据一般呈现两次，在每个cache呈现一次。很多unix系统都是类似模式。

This is simple to implement, but with an obvious inelegance and inefficiency. Starting with Linux kernel version 2.4, the contents of two caches were unified. The VM subsystem now drives I/O and it does so out of the page cache.If cached data has both a file and a block representation -- as most data does -- the buffer cache will simply point into the page cache; thus only one instance of the data is cached in memory. The page cache is what you picture when you think of a disk cache: It caches file data from a disk to make subsequent I/O faster.

这是很容易实现的，但是不够优雅和高效。Linux 2.4 开始，两个cache里的内容统一了。虚拟机子系统现在可以驱动I/O，也可以在页面缓存中进行。如果数据既以文件又以块的形式缓存 -- 这是大部分数据的缓存方式 -- 那么buffer cache将简单地指向page cache；因此仅是一个数据的实例存在内存里（另一个地方存指针）。page cache缓存了你看到的磁盘文件：它从磁盘里缓存file数据优化I/O的速度。

The buffer cache remains,however,as the kernel still needs to perform block I/O in terms of blocks, not pages. As most block represent file data, most of the buffer cache is represented by the page cache. But a small amount of block data isn't file backed-- metadata and raw block I/O for example -- and thus is solely represented by the buffer cache.

这里buffer cache还有一点，内核需要优化块I/O，而不是页存取。因为file数据是用块数据呈现的，大部分buffer cache 呈现为page cache。但是一小部分块数据不是文件备份 -- metadata 和 raw block I/O -- 这些数据用buffer cache存储（比如inode cache和dentry cache）。

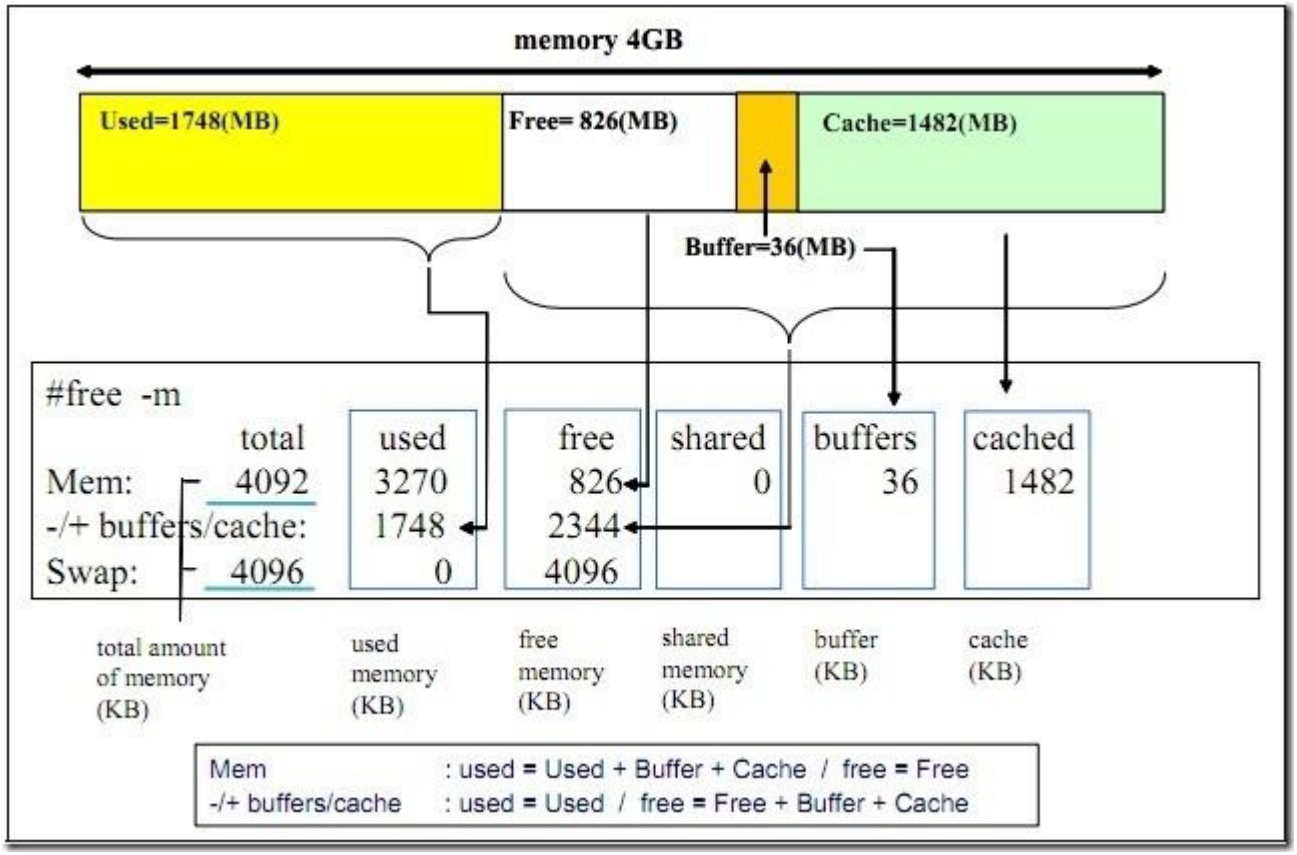
2、在操作系统中释放cache的方式：
echo 1> /proc/sys/vm/drop_caches : 释放page cache
echo 2> /proc/sys/vm/drop_caches : 释放 inode cache和dentry cache
echo 3> /proc/sys/vm/drop_caches : 释放全部

https://www.jianshu.com/p/57fdc61a7294
///

Linux系统中的Page cache和Buffer cache

Free命令显示内存

首先，我们来了解下内存的使用情况



- Mem：表示物理内存统计。
- total：表示物理内存总量(total = used + free)。
- used：表示总计分配给缓存（包含buffers 与cache ）使用的数量，但其中可能部分缓存并未实际使用。
- free：未被分配的内存。
- shared：共享内存。
- buffers：系统分配但未被使用的buffers数量。
- cached：系统分配但未被使用的cache数量。
- -/+ buffers/cache：表示物理内存的缓存统计。
- used2：也就是第一行中的used – buffers - cached也是实际使用的内存总量。 // used2为第二行
- free2 = buffers1 + cached1 + free1 // free2为第二行，buffers1等为第一行
- free2：未被使用的buffers与cache和未被分配的内存之和，这就是系统当前实际可用内存。

- Swap：表示硬盘上交换分区的使用情况。

在Free命令中显示的buffer和cache，它们都是占用内存：

buffer : 作为buffer cache的内存，是块设备的读写缓冲区，更靠近存储设备，或者直接就是disk的缓冲区。

cache: 作为page cache的内存, 文件系统的cache，是memory的缓冲区 。

如果cache 的值很大，说明cache住的文件数很多。如果频繁访问到的文件都能被cache住，那么磁盘的读IO 必会非常小 。

Page cache（页面缓存）

Page cache 也叫页缓冲或文件缓冲，是由好几个磁盘块构成，大小通常为4k，在64位系统上为8k，构成的几个磁盘块在物理磁盘上不一定连续，文件的组织单位为一页， 也就是一个page cache大小，文件读取是由外存上不连续的几个磁盘块，到buffer cache，然后组成page cache，然后供给应用程序。

Page cache在linux读写文件时，它用于缓存文件的逻辑内容，从而加快对磁盘上映像和数据的访问。具体说是加速对文件内容的访问，buffer cache缓存文件的具体内容——物理磁盘上的磁盘块，这是加速对磁盘的访问。

Buffer cache（块缓存）

Buffer cache 也叫块缓冲，是对物理磁盘上的一个磁盘块进行的缓冲，其大小为通常为1k，磁盘块也是磁盘的组织单位。设立buffer cache的目的是为在程序多次访问同一磁盘块时，减少访问时间。系统将磁盘块首先读入buffer cache，如果cache空间不够时，会通过一定的策略将一些过时或多次未被访问的buffer cache清空。程序在下一次访问磁盘时首先查看是否在buffer cache找到所需块，命中可减少访问磁盘时间。不命中时需重新读入buffer cache。对buffer cache的写分为两种，一是直接写，这是程序在写buffer cache后也写磁盘，要读时从buffer cache上读，二是后台写，程序在写完buffer cache后并不立即写磁盘，因为有可能程序在很短时间内又需要写文件，如果直接写，就需多次写磁盘了。这样效率很低，而是过一段时间后由后台写，减少了多次访磁盘的时间。

Buffer cache是由物理内存分配，Linux系统为提高内存使用率，会将空闲内存全分给buffer cache，当其他程序需要更多内存时，系统会减少cache大小。

Buffer page（缓冲页）

如果内核需要单独访问一个块，就会涉及到buffer page，并会检查对应的buffer head。

Swap space（交换空间）

Swap space交换空间，是虚拟内存的表现形式。系统为了应付一些需要大量内存的应用，而将磁盘上的空间做内存使用，当物理内存不够用时，将其中一些暂时不需的数据交换到交换空间，也叫交换文件或页面文件中。做虚拟内存的好处是让进程以为好像可以访问整个系统物理内存。因为在一个进程访问数据时，其他进程的数据会被交换到交换空间中。

Swap cache（交换缓存）

swap cache，它表示交换缓存的大小。Page cache是磁盘数据在内存中的缓存，而swap cache则是交换分区在内存中的临时缓存。

Memory mapping（内存映射）

内核有两种类型的内存映射：共享型(shared)和私有型(private)。私有型是当进程为了只读文件，而不写文件时使用，这时，私有映射更加高效。 但是，任何对私有映射页的写操作都会导致内核停止映射该文件中的页。所以，写操作既不会改变磁盘上的文件，对访问该文件的其它进程也是不可见的。

共享内存中的页通常都位于page cache，私有内存映射只要没有修改，也位于page cache。当进程试图修改一个私有映射内存页时，内核就把该页进行复制，并在页表中用复制的页替换原来的页。由于修改了页表，尽管原来的页仍然在 page cache，但是已经不再属于该内存映射。而新复制的页也不会插入page cache，而是添加到匿名页反向映射数据结构。

Page cache和Buffer cache的区别

磁盘的操作有逻辑级（文件系统）和物理级（磁盘块），这两种Cache就是分别缓存逻辑和物理级数据的。

假设我们通过文件系统操作文件，那么文件将被缓存到Page Cache，如果需要刷新文件的时候，Page Cache将交给Buffer Cache去完成，因为Buffer Cache就是缓存磁盘块的。

也就是说，直接去操作文件，那就是Page Cache区缓存，用dd等命令直接操作磁盘块，就是Buffer Cache缓存的东西。

Page cache实际上是针对文件系统的，是文件的缓存，在文件层面上的数据会缓存到page cache。文件的逻辑层需要映射到实际的物理磁盘，这种映射关系由文件系统来完成。当page cache的数据需要刷新时，page cache中的数据交给buffer cache，但是这种处理在2.6版本的内核之后就变的很简单了，没有真正意义上的cache操作。

Buffer cache是针对磁盘块的缓存，也就是在没有文件系统的情况下，直接对磁盘进行操作的数据会缓存到buffer cache中，例如，文件系统的元数据都会缓存到buffer cache中。

简单说来，page cache用来缓存文件数据，buffer cache用来缓存磁盘数据。在有文件系统的情况下，对文件操作，那么数据会缓存到page cache，如果直接采用dd等工具对磁盘进行读写，那么数据会缓存到buffer cache。

Buffer(Buffer Cache)以块形式缓冲了块设备的操作，定时或手动的同步到硬盘，它是为了缓冲写操作然后一次性将很多改动写入硬盘，避免频繁写硬盘，提高写入效率。

Cache(Page Cache)以页面形式缓存了文件系统的文件，给需要使用的程序读取，它是为了给读操作提供缓冲，避免频繁读硬盘，提高读取效率。

更多相关推荐

Linux中的Page cache和Buffer cache详解

Linux

网上查了查，这里做一下记录。 1、内存情况 在讲解Linux内存管理时已经提到，当你在Linux下频繁存取文件后，即使系统上没有运行许多程序，也会占用大量的物理内存。这是因为当你读写文件的时候，Linux内核为了提高...