Selectively dropping pages from buffer cache

Maksym Planeta

July 16, 2014

Contents

1 Project Idea

The project idea arose when Thomas Knauth was benchmarking. He was repeatedly measuring the timing when things were read from the disk. Dropping everything from the cache, also dropped useful things, not just the few files the benchmark intended to measure.

Thus there was proposed an idea to drop pages from buffer cache selectively. And since the interface of `drop_caches` was convenient enough it was proposed to extend it with leveraging which parts of the buffer cache should be dropped.

The idea is that the user writes a string to the file that represents selective drop page cache interface. This string is a path, i.e., can be either a file or a directory. In case of a directory, the contents is dropped recursively.

2 Related facilities

There exist alternative facilities that make it possible to achieve similar effect as `sdrop_caches`.

The simplest one, that was used as basis for this project is `drop_caches`. This function allows to drop the whole page cache or drop slab buffers or do both at the same time. To use this function one has to write appropriate number to `/proc/sys/vm/drop_caches` file. Number 1 is written to drop page cache, number 2 is written to drop slab buffers and number 3 is written to do both. This function is accessible as part of sysctl interface and can be compiled into the kernel optionally. But usually this interface is compiled in. Advantage of this method is that it is very simple to use. But disadvantage is that the user can't control what exactly to drop at fine grained level and sometimes using this function can lead to performance degradation.

Another option is just to unmount and than mount again the partition with the file which page buffer has to be cleaned. This method is simple to use, but as the previous one, it does not provide fine grained control too. Sometimes it could be impossible to unmount the partition because it contains some very important data (like root partition). Additionally this method requires root privileges to unmount and mount partition. Or partition to be remounted has to be configured for user mount beforehand. The advantage of this method is that it always exists and does not require any changes in the kernel code.

Fine grained control can be provided by `madvise` system call. It advises the kernel about how to handle paging input/output in the specified address range. This system call accepts several types of the advice. And the call with an advice parameter `MADV_DONTNEED` is the one that tells the kernel that the application is not going to access the given range in the near future. Thus, the kernel is free to flush the page buffer for the given range. But the nature of the advise is such that the kernel is free to ignore the advise. Similar behavior is implemented by a call to `posix_fadvise` with the advice value `POSIX_FADV_DONTNEED`. The difference is that the range is specified not in the process virtual memory map, but as region in a file opened by the process.

On the contrary to `madvise` that is not obligated to fulfill the request, `msync` system call with flag `MS_INVALIDATE` is obligated to be executed. This call causes the system to free clean pages and flush dirty pages. Unfortunately, calls to `madvise`, `posix_fadvise` and `msync` are not available from a command line directly.

3 Implementation details

Implementation of `sdrop_caches` sysctl function is based on `drop_caches` function. The difference is that instead of integer number it accepts string with a path to file or directory that has to be cleaned.

The path name string is converted to `struct path` which contains appropriate `struct dentry` for this path. This structure contains pointer to the memory map of the file which page cache has to be cleaned. After cleaning the page cache of the directory node it is checked if this directory node has children. And if it the case than children are traversed and cleaned too.

There is implementation of `sdrop_caches` both as a separate module and as patch for vanilla kernel.

The interface can be used as follows:

```
echo /path/to/the/file/or/directory > /proc/sys/vm/sdrop_cache
```

4 Discussion

After first implementation was finished the patch with `sdrop_caches` was proposed on `linux-mm` [1] mailing list. In the discussion of the patch there were mentioned several shortcomings and contradictory points.

The major shortcoming is that the proposed patch does not limit the depth of the recursion that performs lookup in directory entry's children.

There was a discussion regarding access control. At the moment possibility to clean the page cache is restricted by access control mode to the file `sdrop_caches`. As an alternative there was a proposition to make access control based on access mode of the file which cache is to be cleaned. So that the user can clean the page cache only for the files that he has the right to write to, read from or if he owns the file. This feature could be easily implemented, because credentials of current users are known for the kernel and access mode to the directory entries can be also determined.

Also, it is questionable what should happen when the user tries to drop cache for the symbolic link. There are several options. First, the cache for the link can be dropped. Second, the link can be followed and the cache for the linked file can be dropped. Third, page cache for both link and linked file can be dropped. There is no obvious answer which option is better. And the sysctl interface does not allow to provide additional parameters easily,

[1] The thread topic is "[PATCH] sysctl: Add a feature to drop caches selectively"

so that the user can make the choice which behavior is more appropriate for him.

One possible extension of the `sdrop_caches` interface is adding support for ranges. Current implementations drops the whole cache for the file. But with some changes to the patch, user will be able to tell witch range he wants to drop exactly. The request in this case can look as follows:

```
echo "my-file 0-1000,8000-10000" > /proc/sys/vm/sdrop_cache
```

Here, `0-1000` and `8000-10000` are the ranges which cache has to be dropped. The problem, is that such extension of the interface was not planned by sysctl developers and it requires many changes in the kernel code.

Because sysctl interface has limited extensibility it was proposed to use another interfaces, like ioctl or new syscall. The final decision on which interface to use can be done when the features of the `sdrop_caches` facility will be negotiated completely.

5 Conclusion

During this project there was proposed new interface to clean drop caches that can be convenient to use, for example, for benchmarking. Work on this project showed some alternatives for the proposed approach. Additionally there were discovered some discussion points. Some of the make the approach less eligible to use in production systems (like unbound recursion problem). Others show space for further improvement.