Subject: Re: [PATCH 2.6.19-rc3] VFS: per-sb dentry lru list
Posted by vaverin on Tue, 31 Oct 2006 13:08:03 GMT
View Forum Message <> Reply to Message

Hello Neil,

Neil Brown wrote:
>>> We've been down this path before:
>>>
>>> http://marc.theaimsgroup.com/?l=linux-kernel&m=114861109 717260&w=2
>>>
>>> A lot of comments on per-sb unused dentry lists were made in
>>> the threads associated with the above. other solutions were
>>> found to the problem that the above patch addressed, but I don't
>>> think any of them have made it to mainline yet. You might want
>>> to have a bit of a read of these threads first...
>> The major difference between our patch and the one discussed in the link
>> it that we keep both global and per-sb dentry LRU lists.
>> Thus, when needed normal LRU is used and prune logic is unchanged,
>> while umount/remount use per-sb list and do its job faster.
>
> Yes, we have been down this path before - several times I think.
> Below is the patch that I like (not tested recently - just rediffed
> and reviewed).

I've reviewed your patch, in general it looks great and I'm very like an idea to
have only one place to free the unused dentries.
I have several minor notes and one big question:

```
> @@ -485,68 +483,25 @@ static void prune_dcache(int count, stru
>      list_add(&dentry->d_lru, &dentry_unused);
>     dentry_stat.nr_unused++;
>   }
> + /* split any remaining entries back onto dentry_unused */
> + if (have_list)
> +  list_splice(list, dentry_unused.prev);
>   spin_unlock(&dcache_lock);
>  }
```

- It seems for me this hunk is not required here. You are knows how many entries
are present in the private list and nobody can add new dentries to it. Therefore
I think your list should be empty on exit.

- I do not like idea to split dentry_unused list, IMHO it broke LRU slightly (if
somebody on the other CPU's shrinks dcache). However on the other hand this
split is temporal and therefore it should not lead to any problems.

Unfortunately I'm not sure that it can help us against long remount. As far as I

understand the work time of shrink_dcache_sb() function depends on the subtree size, and you need to check whole tree several (at least 3) times:
1) to remove all the unused dentries from dentry_unused to the private list
2) to collect all the non-freed dentries (prune_dcache will insert DCACHE_REFERENCED dentries back to the global LRU list)
3) to check that there is not any unused dentries.

If the tree is large we will handle it too long and can be rescheduled. In these cases steps 1) and 2) may be restarted several times. In the worst case we will prune only one dentry per cycle. And it is without any filesystem activity.

And now let's assume that we do the same on the live filesystem. I'm afraid in this case shrink_dcache_sb() may be executed unpredictable long time with taken s_umount semaphore. :(I would note that it is real issue, and some of our users have complained about this behaviour.

In case of per-sb LRU we will not have any additional overheads, we will be able to traverse over the unused dentries list without any additional overheads. Heavy activity on the filesystem will delay us too, but we will not depends on filesystem size.

As far as I understand nobody is against per-sb LRU, but nobody wants to add the new fields to the dentry struct.

Let's see on the possible alternatives:

1) per-sb LRU + global LRU (OpenVZ patch): it is fastest and (IMHO) performance-optimal but requires 2 additional pointers on dentry struct.

2) per-sb LRU + per-dentry time stamp (Eric Dumazet idea):
shirnk_dcache_sb() and per-sb prune_dcache(,sb) are optimal (like in case 1), but shrink_dcache(,NULL) called from shrink_dcache_memory() will be executed slower than in current version(). It requires only one additional field on dentry struct for time stamp, but probably we can use some of current fields on the dentry struct (d_count? d_time?) for this purpose?

3) we can remove both LRU from struct dentry to the some dynamic-allocated structure. It will be allocated for each new unused dentry and it can contain both LRU fields and probably some other fields. Also by this way we can decrease size of dentry. However ratio used/unused dentries is too low and probably it will not have any advantages.

4) think how we can modify the other proposed patches (Neil Brown, Dave Chinner, David Howells)

5) any new ideas?

Thank you,

Vasily Averin