Subject: Re: [PATCH 3/5] Switch caches notification dynamically
Posted by Pavel Emelianov on Mon, 01 Oct 2007 13:19:39 GMT
View Forum Message <> Reply to Message

Balbir Singh wrote:
> Pavel Emelyanov wrote:
>> The /sys/slab/<name>/cache_notify attribute controls
>> whether the cache <name> is to be accounted or not.
>>
>> By default no caches are accountable. Simply make
>>   # echo -n 1 > /sys/slab/<name>cache_notify
>> to turn notification of this cache on.
>>
>> If we turn accounting on on some cache and this cache
>> is merged with some other, this "other" will be notified
>> as well. One can solve this by disabling of cache merging,
>> i.e. booting with slub_nomerge option to the kernel.
>>
>> Turning the notifications "on" causes all te subsequent
>> allocations use the slow allocation path, so all the
>> per-cpu caches are flushed and all the partial pages
>> are marked as SlabDebug.
>>
>
> Do we know of the overhead of slub notifications? It would
> be nice to know and probably add it to the help text in
> Kconfig.

When the notification is off the overhead is 0. When they are
on it depends on whether we use generic or direct notifications.

Roughly:

The generic notification costs us up to 10% of performance loss.
The direct notification costs us really small, providing the
listener is noop. If the listener is doing something - the performance
loss can be higher, but how much - depends on the listener :)

>
>> ---
>>
>> diff --git a/mm/slub.c b/mm/slub.c
>> index 31d04a3..e066a0e 100644
>> --- a/mm/slub.c
>> +++ b/mm/slub.c
>> @@ -3779,6 +3932,60 @@ static ssize_t defrag_ratio_store(struct
>> SLAB_ATTR(defrag_ratio);
>> #endif

>>
>> +static ssize_t cache_notify_show(struct kmem_cache *s, char *buf)
>> +{
>> + return sprintf(buf, "%d\n", !!(s->flags & SLAB_NOTIFY));
>> +}
>> +
>> +static ssize_t cache_notify_store(struct kmem_cache *s,
>> + const char *buf, size_t length)
>> +{
>> + int err;
>> + int n;
>> +
>> + if ((buf[0] == '1') && !(s->flags & SLAB_NOTIFY)) {
>
> Won't this code break if I pass 10 as input in buf?
>
>> + err = slub_on_notify(s);
>> + if (err < 0)
>> + return err;
>> +
>> + s->flags |= SLAB_NOTIFY;
>> +
>> + flush_all(s);
>
> Sounds like an expensive operation, can we add a comment
> to indicate the same. Also, in the documentation, could
> we warn the user that turning notifications on/off
> can be expensive?
>
>> + for_each_node_state(n, N_NORMAL_MEMORY) {
>> + struct kmem_cache_node *node;
>> + struct page *pg;
>> +
>> + node = get_node(s, n);
>> + spin_lock_irq(&node->list_lock);
>> + list_for_each_entry(pg, &node->partial, lru)
>> + SetSlabDebug(pg);
>> + spin_unlock_irq(&node->list_lock);
>> + }
>> + return length;
>> + }
>> +
>> + if ((buf[0] == '0') && (s->flags & SLAB_NOTIFY)) {
>
> What happens if buf is 01?
>
>> + /*
>> + * TODO: make the notifications-off work

```
>> +   */
>> +  if (any_slab_objects(s))
>> +   return -EBUSY;
>> +
>> +  s->flags &= ~SLAB_NOTIFY;
>> +  err = slub_off_notify(s);
>> +  if (err < 0) {
>> +   s->flags |= SLAB_NOTIFY;
>> +   return err;
>> +  }
>> +
>> +  return length;
>> + }
>> +
>> + return -EINVAL;
>> +}
>> +
>> +SLAB_ATTR(cache_notify);
>> +
>>  static struct attribute * slab_attrs[] = {
>>   &slab_size_attr.attr,
>>   &object_size_attr.attr,
>> @@ -3809,6 +4016,7 @@ static struct attribute * slab_attrs[] =
>>  #ifdef CONFIG_NUMA
>>   &defrag_ratio_attr.attr,
>>  #endif
>> + &cache_notify_attr.attr,
>>   NULL
>>  };
>>
>>
>
>
```