
Subject: Re: [RFC 4/4] memcg: NUMA background reclaim
Posted by [KAMEZAWA Hiroyuki](#) on Wed, 28 May 2008 00:44:23 GMT
[View Forum Message](#) <> [Reply to Message](#)

On Tue, 27 May 2008 22:56:43 +0530
Balbir Singh <balbir@linux.vnet.ibm.com> wrote:

```
> KAMEZAWA Hiroyuki wrote:
> > One aspect of difference in reclaim logic between global lru and memcg is
> > * global LRU triggers memory reclaim at memory shortage.
> > * memcg LRU triggers memory reclaim at excess of usage.
> >
> > Then, global LRU _know_ which node we should start reclaim from.
> > * start from a node at memory shortage or
> > * start from a node where memory allocation is waiting
> >
> > WRT memcg, it's difficult to find where we should start because
> > there is no memory shortage and LRU is splitted.
> > (But per-zone-LRU is definitely necessary for scalability.)
> >
> > This patch tries to determine a node for starting reclaim by checking
> > ratio of inactive pages/active pages in a node. And trying to avoid starting
> > from a node with relatively small usage.
> > Better algorithm is welcome.
> >
> > Signed-off-by: KAMEZAWA Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com>
> >
> > Index: mm-2.6.26-rc2-mm1/mm/memcontrol.c
> > =====
> > --- mm-2.6.26-rc2-mm1.orig/mm/memcontrol.c
> > +++ mm-2.6.26-rc2-mm1/mm/memcontrol.c
> > @@ -578,7 +578,7 @@ retry:
> >  if (state == RES_OVER_LIMIT) {
> >    if (!gfp_mask & __GFP_WAIT)
> >      goto out;
> > - if (try_to_free_mem_cgroup_pages(mem, gfp_mask))
> > + if (try_to_free_mem_cgroup_pages(mem, -1, gfp_mask))
> >    goto retry;
> >  /*
> >   * try_to_free_mem_cgroup_pages() might not give us a
> > @@ -801,7 +801,7 @@ int mem_cgroup_shrink_usage(struct mm_st
> >  rcu_read_unlock();
> >
> >  do {
> > - progress = try_to_free_mem_cgroup_pages(mem, gfp_mask);
> > + progress = try_to_free_mem_cgroup_pages(mem, -1, gfp_mask);
> >  } while (!progress && --retry);
> >
```

```

>> if (!retry)
>> @@ -814,7 +814,7 @@ static void mem_cgroup_drop_all_pages(st
>> {
>> int progress;
>> while (!res_counter_empty(&mem->res)) {
>> - progress = try_to_free_mem_cgroup_pages(mem,
>> + progress = try_to_free_mem_cgroup_pages(mem, -1,
>>     GFP_HIGHUSER_MOVABLE);
>> if (!progress) /* we did as much as possible */
>> break;
>> @@ -912,6 +912,62 @@ out:
>> /*
>>  * background reclaim daemon.
>> */
>> +
>> + #ifdef CONFIG_NUMA
>> + /*
>> +  * Because memory controller's memory reclaim doesn't come from memory shortage,
>> +  * we cannot know which node should be reclaimed in an easy way.
>> +  * This routine select a node with inactive pages to be a node for starting
>> +  * scanning.
>> +  */
>> + int __select_best_node(struct mem_cgroup *mem)
>> + {
>> + int nid;
>> + int best_node = -1;
>> + unsigned long highest_inactive_ratio = 0;
>> + unsigned long active, inactive, inactive_ratio, total, threshold, flags;
>> + struct mem_cgroup_per_zone *mz;
>> + int zid;
>> +
>> + /*
>> +  * When a node's memory usage is smaller than
>> +  * total_usage/num_of_node * 75%, we don't select the node
>> +  */
>> + total = mem->res.usage >> PAGE_SHIFT;
>> + threshold = (total / num_node_state(N_HIGH_MEMORY)) * 3 / 4;
>> +
>> + /*
>> +  * See nodemask.h, N_HIGH_MEMORY means that a node has memory
>> +  * can be used for user's memory.(i.e. not means HIGHMEM).
>> +  */
>> + for_each_node_state(nid, N_HIGH_MEMORY) {
>> + active = 0;
>> + inactive = 0;
>> +
>> + for (zid = 0; zid < MAX_NR_ZONES; zid++) {
>> + mz = mem_cgroup_zoneinfo(mem, nid, zid);

```

```
> > + spin_lock_irqsave(&mz->lru_lock, flags);
> > + active += MEM_CGROUP_ZSTAT(mz, MEM_CGROUP_ZSTAT_ACTIVE);
> > + inactive +=
> > + MEM_CGROUP_ZSTAT(mz, MEM_CGROUP_ZSTAT_INACTIVE);
> > + spin_unlock_irqrestore(&mz->lru_lock, flags);
> > + }
> > +
> > + if (active + inactive < threshold)
> > + continue;
> > + inactive_ratio = (inactive * 100) / (active + 1);
> > + if (inactive_ratio > highest_inactive_ratio)
> > + best_node = nid;
>
> Shouldn't we update highest_inactive_ratio here?
>
AH, yes. blame me :( Thanks!
```

Thanks,
-Kame

Containers mailing list
Containers@lists.linux-foundation.org
<https://lists.linux-foundation.org/mailman/listinfo/containers>
