
Subject: Re: [PATCH] memory.min_usage
Posted by [Balbir Singh](#) on Tue, 04 Dec 2007 13:30:25 GMT
[View Forum Message](#) <> [Reply to Message](#)

YAMAMOTO Takashi wrote:

> hi,
>
> here's a patch to implement memory.min_usage,
> which controls the minimum memory usage for a cgroup.
>
> it works similarly to mlock;
> global memory reclamation doesn't reclaim memory from
> cgroups whose memory usage is below the value.
> setting it too high is a dangerous operation.
>
> it's against 2.6.24-rc3-mm2 + memory.swappiness patch i posted here yesterday.
> but it's logically independent from the swappiness patch.
>

Hi,

I think it's a good approach, but we need to make the reclaim dependent on priority. I think that if we stop reclaiming from a cgroup even though there is high global pressure, it might be a bad idea.

> todo:
> - restrict non-root user's operation regardless of owner of cgroupfs files?

Sorry, I don't understand this and I think using file permissions is a good idea.

> - make oom killer aware of this?
>

Yes, we might also need a memory controller version of

1. overcommit
2. oom_adj* parameters

> YAMAMOTO Takashi
>
>
> Signed-off-by: YAMAMOTO Takashi <yamamoto@valinux.co.jp>
> ---
>
> --- linux-2.6.24-rc3-mm2-swappiness/include/linux/memcontrol.h.BACKUP2 2007-12-03
13:52:37.000000000 +0900

```

> +++ linux-2.6.24-rc3-mm2-swappiness/include/linux/memcontrol.h 2007-12-03
14:07:45.000000000 +0900
> @@ -47,6 +47,7 @@ extern int mem_cgroup_cache_charge(struct
>     gfp_t gfp_mask);
> int task_in_mem_cgroup(struct task_struct *task, const struct mem_cgroup *mem);
> extern int mem_cgroup_swappiness(struct mem_cgroup *mem);
> +extern int mem_cgroup_canreclaim(struct page *page, struct mem_cgroup *mem);
>
> static inline struct mem_cgroup *mm_cgroup(const struct mm_struct *mm)
> {
> --- linux-2.6.24-rc3-mm2-swappiness/mm/memcontrol.c.BACKUP2 2007-12-03
13:52:13.000000000 +0900
> +++ linux-2.6.24-rc3-mm2-swappiness/mm/memcontrol.c 2007-12-04 11:42:07.000000000
+0900
> @@ -134,6 +134,7 @@ struct mem_cgroup {
>     unsigned long control_type; /* control RSS or RSS+Pagecache */
>     int prev_priority; /* for recording reclaim priority */
>     unsigned int swappiness; /* swappiness */
> + unsigned long long min_usage; /* XXX should be a part of res_counter? */
> /*
>  * statistics.
>  */
> @@ -1096,6 +1097,22 @@ static u64 mem_cgroup_swappiness_read(st
>     return mem->swappiness;
> }
>
> +static int mem_cgroup_min_usage_write(struct cgroup *cont, struct cftype *cft,
> +     u64 val)
> +{
> + struct mem_cgroup *mem = mem_cgroup_from_cont(cont);
> +
> + mem->min_usage = val;
> + return 0;
> +}
> +
> +static u64 mem_cgroup_min_usage_read(struct cgroup *cont, struct cftype *cft)
> +{
> + struct mem_cgroup *mem = mem_cgroup_from_cont(cont);
> +
> + return mem->min_usage;
> +}
> +
> static struct cftype mem_cgroup_files[] = {
> {
>     .name = "usage_in_bytes",
> @@ -1132,6 +1149,11 @@ static struct cftype mem_cgroup_files[]
>     .write_uint = mem_cgroup_swappiness_write,
>     .read_uint = mem_cgroup_swappiness_read,

```

```

> },
> +{
> + .name = "min_usage_in_bytes",
> + .write_uint = mem_cgroup_min_usage_write,
> + .read_uint = mem_cgroup_min_usage_read,
> + },
> };
>
> /* XXX probably it's better to move try_to_free_mem_cgroup_pages to
> @@ -1142,6 +1164,36 @@ int mem_cgroup_swappiness(struct mem_cgr
> return mem->swappiness;
> }
>
> +int mem_cgroup_canreclaim(struct page *page, struct mem_cgroup *mem1)
> +{
> + struct page_cgroup *pc;
> + int result = 1;
> +
> + if (mem1 != NULL)
> + return 1;
> +
> + lock_page_cgroup(page);
> + pc = page_get_page_cgroup(page);
> + if (pc) {
> + struct mem_cgroup *mem2 = pc->mem_cgroup;
> + unsigned long long min_usage;
> +
> + BUG_ON(mem2 == NULL);
> + min_usage = mem2->min_usage;
> + if (min_usage != 0) {
> + unsigned long flags;
> +
> + spin_lock_irqsave(&mem2->res.lock, flags);
> + if (mem2->res.usage <= min_usage)
> + result = 0;
> + spin_unlock_irqrestore(&mem2->res.lock, flags);
> + }
> + }
> + unlock_page_cgroup(page);
> +
> + return result;
> +}
> +
> static int alloc_mem_cgroup_per_zone_info(struct mem_cgroup *mem, int node)
> {
> struct mem_cgroup_per_node *pn;
> --- linux-2.6.24-rc3-mm2-swappiness/mm/vmscan.c.BACKUP2 2007-12-03 13:52:22.000000000
+0900

```

```

> +++ linux-2.6.24-rc3-mm2-swappiness/mm/vmscan.c 2007-12-03 14:11:42.000000000 +0900
> @@ -531,6 +531,11 @@ static unsigned long shrink_page_list(st
>     referenced && page_mapping_inuse(page))
>     goto activate_locked;
>
> +#ifdef CONFIG_CGROUP_MEM_CONT
> + if (!mem_cgroup_canreclaim(page, sc->mem_cgroup))
> +     goto activate_locked;
> +#endif /* CONFIG_CGROUP_MEM_CONT */
> +
> #ifdef CONFIG_SWAP
>     /*
>      * Anonymous process memory has backing store?
> @@ -1122,6 +1127,12 @@ static void shrink_active_list(unsigned
>     list_add(&page->lru, &l_active);
>     continue;
> }
> +#ifdef CONFIG_CGROUP_MEM_CONT
> + if (!mem_cgroup_canreclaim(page, sc->mem_cgroup)) {
> +     list_add(&page->lru, &l_active);
> +     continue;
> + }
> +#endif /* CONFIG_CGROUP_MEM_CONT */
>     list_add(&page->lru, &l_inactive);
> }
>
>
> _____
> Containers mailing list
> Containers@lists.linux-foundation.org
> https://lists.linux-foundation.org/mailman/listinfo/containers

```

--

Warm Regards,
Balbir Singh
Linux Technology Center
IBM, ISTL

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