Subject: Re: [RFC][only for review] memory controller bacground reclaim [4/5] high/low watermark for memory
Posted by Pavel Emelianov on Wed, 28 Nov 2007 11:20:33 GMT
View Forum Message <> Reply to Message

KAMEZAWA Hiroyuki wrote:

```
> High Low watermark for page reclaiming in memory cgroup(1)
>
> High-Low value here is implemented for support background reclaim.
>
>  - If USAGE is bigger than high watermark, background reclaim starts.
>  - If USAGE is lower than low watermark, background reclaim stops.
>
> Each value is represented in bytes.
> (Not configurable now, but can be easily.)
> The routine which reclaim is in the next patch. This patch just adds
> high/low watermark value.
>
> Changes from YAMAMOTO-san's version
>   - I like automatic adjustment of high/low watermark.
>     So, I added a code to modify high/low watermark every time the limit
>     changes. A user can custorimize it by hand later.
>     (I think 93%/97% is not so bad value. need investigation.)
>
> TODO:
>  - update documentation.
>
> Signed-off-by: KAMEZAWA Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com>
>
>  mm/memcontrol.c |   63 +++++++++++++++++++++++++++++++++++++++++++++++++++++++++------
>  1 file changed, 57 insertions(+), 6 deletions(-)
>
> Index: linux-2.6.24-rc3-mm1/mm/memcontrol.c
> ===================================================================
> --- linux-2.6.24-rc3-mm1.orig/mm/memcontrol.c 2007-11-28 16:30:44.000000000 +0900
> +++ linux-2.6.24-rc3-mm1/mm/memcontrol.c 2007-11-28 16:44:57.000000000 +0900
> @@ -108,16 +108,12 @@
>   struct mem_cgroup_per_node *nodeinfo[MAX_NUMNODES];
>  };
>
> +
>  /*
>   * The memory controller data structure. The memory controller controls both
>   * page cache and RSS per cgroup. We would eventually like to provide
>   * statistics based on the statistics developed by Rik Van Riel for clock-pro,
>   * to help the administrator determine what knobs to tune.
> - *
> - * TODO: Add a water mark for the memory controller. Reclaim will begin when
```

> - * we hit the water mark. May be even add a low water mark, such that
> - * no reclaim occurs from a cgroup at it's low water mark, this is
> - * a feature that will be implemented much later in the future.
> */
> struct mem_cgroup {
> struct cgroup_subsys_state css;
> @@ -162,6 +158,15 @@
> #define PAGE_CGROUP_FLAG_CACHE (0x1) /* charged as cache */
> #define PAGE_CGROUP_FLAG_ACTIVE (0x2) /* page is active in this cgroup */
>
> +/*
> + * Default value for high-low watermark for start/stop background reclaim.
> + * represented in percent here. can be customized by user later.
> + * This is applied always limit change.
> + */
> +#define DEFAULT_WATERMARK_PERCENT_LOW (93ULL)
> +#define DEFAULT_WATERMARK_PERCENT_HIGH (97ULL)
> +
> +
> static inline int page_cgroup_nid(struct page_cgroup *pc)
> {
> return page_to_nid(pc->page);
> @@ -926,6 +931,29 @@
> return 0;
> }
>
> +static void mem_cgroup_init_watermark(struct cgroup *cont)
> +{
> + struct mem_cgroup *mem;
> + unsigned long long val, high, low;
> + unsigned long flags;
> +
> + mem = mem_cgroup_from_cont(cont);
> + spin_lock_irqsave(&mem->res.lock, flags);
> + val = res_counter_get(&mem->res, RES_LIMIT);
> + if (val == (unsigned long long) LLONG_MAX) {
> + low = (unsigned long long) LLONG_MAX;
> + high = (unsigned long long) LLONG_MAX;
> + } else {
> + low = val * DEFAULT_WATERMARK_PERCENT_LOW / 100ULL;
> + high = val * DEFAULT_WATERMARK_PERCENT_HIGH / 100ULL;
> + }
> + res_counter_set(&mem->res, RES_LOW_WATERMARK, low);
> + res_counter_set(&mem->res, RES_HIGH_WATERMARK, high);
> + spin_unlock_irqrestore(&mem->res.lock, flags);
> + return;
> +}
> +

> +
> static ssize_t mem_cgroup_read(struct cgroup *cont,
>   struct cftype *cft, struct file *file,
>   char __user *userbuf, size_t nbytes, loff_t *ppos)
> @@ -944,6 +972,17 @@
>   mem_cgroup_write_strategy);
> }
>
> +static ssize_t mem_cgroup_write_limit(struct cgroup *cont, struct cftype *cft,
> +   struct file *file, const char __user *userbuf,
> +   size_t nbytes, loff_t *ppos)
> +{
> + ssize_t ret;
> + ret = mem_cgroup_write(cont, cft, file, userbuf, nbytes, ppos);
> + if (ret > 0)
> + mem_cgroup_init_watermark(cont);

No, please, no! I'd be very disappointed if I tune high and low watermarks
carefully and then they are silently re-set after I tune the limit. Better
(see my comment to patch #3) return -EINVAL in case I try to set limit
below hwmark. Please :)

> + return ret;
> +}
> +
> static ssize_t mem_control_type_write(struct cgroup *cont,
>   struct cftype *cft, struct file *file,
>   const char __user *userbuf,
> @@ -1088,7 +1127,7 @@
>   {
>   .name = "limit_in_bytes",
>   .private = RES_LIMIT,
> - .write = mem_cgroup_write,
> + .write = mem_cgroup_write_limit,
>   .read = mem_cgroup_read,
>   },
>   {
> @@ -1102,6 +1141,18 @@
>   .read = mem_control_type_read,
>   },
>   {
> + .name = "low_watermark_in_bytes",
> + .private = RES_LOW_WATERMARK,
> + .write = mem_cgroup_write,
> + .read = mem_cgroup_read,
> + },
> + {
> + .name = "high_watermark_in_bytes",

> + .private = RES_HIGH_WATERMARK,
> + .write = mem_cgroup_write,
> + .read = mem_cgroup_read,
> + },
> + {
>   .name = "force_empty",
>   .write = mem_force_empty_write,
>   .read = mem_force_empty_read,
>
>

_____

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