Subject: Re: [-mm PATCH] Memory controller improve user interface
Posted by Paul Menage on Wed, 29 Aug 2007 15:28:56 GMT
View Forum Message <> Reply to Message

On 8/29/07, Balbir Singh <balbir@linux.vnet.ibm.com> wrote:
>
> Change the interface to use kilobytes instead of pages. Page sizes can vary
> across platforms and configurations. A new strategy routine has been added
> to the resource counters infrastructure to format the data as desired.
>
> Suggested by David Rientjes, Andrew Morton and Herbert Poetzl
>
> Signed-off-by: Balbir Singh <balbir@linux.vnet.ibm.com>
> ---
>
> Documentation/controllers/memory.txt | 7 +++--
> include/linux/res_counter.h | 6 ++--
> kernel/res_counter.c | 24 +++++++++++++----
> mm/memcontrol.c | 47 +++++++++++++++++++++++++--------
> 4 files changed, 64 insertions(+), 20 deletions(-)
>
> diff -puN mm/memcontrol.c~mem-control-make-ui-use-kilobytes mm/memcontrol.c
> --- linux-2.6.23-rc3/mm/memcontrol.c~mem-control-make-ui-use-kilobytes 2007-08-28
13:20:44.000000000 +0530
> +++ linux-2.6.23-rc3-balbir/mm/memcontrol.c   2007-08-29 14:36:07.000000000 +0530
> @@ -32,6 +32,7 @@
>
> struct container_subsys mem_container_subsys;
> static const int MEM_CONTAINER_RECLAIM_RETRIES = 5;
> +static const int MEM_CONTAINER_CHARGE_KB = (PAGE_SIZE >> 10);
>
> /*
>   * The memory controller data structure. The memory controller controls both
> @@ -312,7 +313,7 @@ int mem_container_charge(struct page *pa
>   * If we created the page_container, we should free it on exceeding
>   * the container limit.
>   */
> -   while (res_counter_charge(&mem->res, 1)) {
> +   while (res_counter_charge(&mem->res, MEM_CONTAINER_CHARGE_KB)) {
>   if (try_to_free_mem_container_pages(mem))
>   continue;
>
> @@ -352,7 +353,7 @@ int mem_container_charge(struct page *pa
>   kfree(pc);
>   pc = race_pc;
>   atomic_inc(&pc->ref_cnt);
> -   res_counter_uncharge(&mem->res, 1);
> +   res_counter_uncharge(&mem->res, MEM_CONTAINER_CHARGE_KB);

```
>                 css_put(&mem->css);
>                 goto done;
>         }
> @@ -417,7 +418,7 @@ void mem_container_uncharge(struct page_
>                 css_put(&mem->css);
>                 page_assign_page_container(page, NULL);
>                 unlock_page_container(page);
> -               res_counter_uncharge(&mem->res, 1);
> +                res_counter_uncharge(&mem->res, MEM_CONTAINER_CHARGE_KB);
>
>                 spin_lock_irqsave(&mem->lru_lock, flags);
>                 list_del_init(&pc->lru);
> @@ -426,12 +427,37 @@ void mem_container_uncharge(struct page_
>         }
>  }
>
> -static ssize_t mem_container_read(struct container *cont, struct cftype *cft,
> -                       struct file *file, char __user *userbuf, size_t nbytes,
> -                       loff_t *ppos)
> +int mem_container_read_strategy(unsigned long val, char *buf)
> +{
> +       return sprintf(buf, "%lu (kB)\n", val);
> +}
> +
> +int mem_container_write_strategy(char *buf, unsigned long *tmp)
> +{
> +       *tmp = memparse(buf, &buf);
> +       if (*buf != '\0')
> +               return -EINVAL;
> +
> +       *tmp = *tmp >> 10;              /* convert to kilobytes */
> +       return 0;
> +}
```

This seems a bit inconsistent - if you write a value to a limit file,
then the value that you read back is reduced by a factor of 1024?
Having the "(kB)" suffix isn't really a big help to automated
middleware.

I'd still be in favour of just reading/writing 64-bit values
representing bytes - simple, and unambiguous for programmatic use, and
not really any less user-friendly than kilobytes for manual use
(since the numbers involved are going to be unwieldly for manual use
whether they're in bytes or kB).

Paul

_____
Containers mailing list

Subject: Re: [-mm PATCH] Memory controller improve user interface
Posted by Balbir Singh on Wed, 29 Aug 2007 16:07:38 GMT
View Forum Message <> Reply to Message

Paul Menage wrote:
> On 8/29/07, Balbir Singh <balbir@linux.vnet.ibm.com> wrote:
>> Change the interface to use kilobytes instead of pages. Page sizes can vary
>> across platforms and configurations. A new strategy routine has been added
>> to the resource counters infrastructure to format the data as desired.
>>
>> Suggested by David Rientjes, Andrew Morton and Herbert Poetzl
>>
>> Signed-off-by: Balbir Singh <balbir@linux.vnet.ibm.com>
>> ---
>>
>> Documentation/controllers/memory.txt | 7 +++--
>> include/linux/res_counter.h | 6 ++--
>> kernel/res_counter.c | 24 +++++++++++++----
>> mm/memcontrol.c | 47 +++++++++++++++++++++++++--------
>> 4 files changed, 64 insertions(+), 20 deletions(-)
>>
>> diff -puN mm/memcontrol.c~mem-control-make-ui-use-kilobytes mm/memcontrol.c
>> --- linux-2.6.23-rc3/mm/memcontrol.c~mem-control-make-ui-use-kilobytes 2007-08-28
13:20:44.000000000 +0530
>> +++ linux-2.6.23-rc3-balbir/mm/memcontrol.c   2007-08-29 14:36:07.000000000 +0530
>> @@ -32,6 +32,7 @@
>>
>> struct container_subsys mem_container_subsys;
>> static const int MEM_CONTAINER_RECLAIM_RETRIES = 5;
>> +static const int MEM_CONTAINER_CHARGE_KB = (PAGE_SIZE >> 10);
>>
>> /*
>>   * The memory controller data structure. The memory controller controls both
>> @@ -312,7 +313,7 @@ int mem_container_charge(struct page *pa
>>   * If we created the page_container, we should free it on exceeding
>>   * the container limit.
>>   */
>> -   while (res_counter_charge(&mem->res, 1)) {
>> +   while (res_counter_charge(&mem->res, MEM_CONTAINER_CHARGE_KB)) {
>>   if (try_to_free_mem_container_pages(mem))
>>   continue;
>>
>> @@ -352,7 +353,7 @@ int mem_container_charge(struct page *pa
>>   kfree(pc);

```
>>                pc = race_pc;
>>                atomic_inc(&pc->ref_cnt);
>> -              res_counter_uncharge(&mem->res, 1);
>> +               res_counter_uncharge(&mem->res, MEM_CONTAINER_CHARGE_KB);
>>                css_put(&mem->css);
>>                goto done;
>>        }
>> @@ -417,7 +418,7 @@ void mem_container_uncharge(struct page_
>>                css_put(&mem->css);
>>                page_assign_page_container(page, NULL);
>>                unlock_page_container(page);
>> -              res_counter_uncharge(&mem->res, 1);
>> +               res_counter_uncharge(&mem->res, MEM_CONTAINER_CHARGE_KB);
>>
>>                spin_lock_irqsave(&mem->lru_lock, flags);
>>                list_del_init(&pc->lru);
>> @@ -426,12 +427,37 @@ void mem_container_uncharge(struct page_
>>        }
>>  }
>>
>> -static ssize_t mem_container_read(struct container *cont, struct cftype *cft,
>> -                    struct file *file, char __user *userbuf, size_t nbytes,
>> -                    loff_t *ppos)
>> +int mem_container_read_strategy(unsigned long val, char *buf)
>> +{
>> +      return sprintf(buf, "%lu (kB)\n", val);
>> +}
>> +
>> +int mem_container_write_strategy(char *buf, unsigned long *tmp)
>> +{
>> +      *tmp = memparse(buf, &buf);
>> +      if (*buf != '\0')
>> +              return -EINVAL;
>> +
>> +      *tmp = *tmp >> 10;              /* convert to kilobytes */
>> +      return 0;
>> +}
>
> This seems a bit inconsistent - if you write a value to a limit file,
> then the value that you read back is reduced by a factor of 1024?
> Having the "(kB)" suffix isn't really a big help to automated
> middleware.
>
```

Why is that? Is it because you could write 4M and see it show up
as 4096 kilobytes? We'll that can be fixed with another variant
of the memparse() utility.

> I'd still be in favour of just reading/writing 64-bit values
> representing bytes - simple, and unambiguous for programmatic use, and
> not really any less user-friendly than kilobytes for manual use
> (since the numbers involved are going to be unwieldly for manual use
> whether they're in bytes or kB).
>

64 bit might be an overkill for 32 bit machines. 32 bit machines with
PAE cannot use 32 bit values, they need 64 bits. I think KiloBytes
is an acceptable metric these days, everybody understands them.

> Paul
> _____
> 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