Subject: Re: Per container statistics (containerstats)
Posted by Balbir Singh on Fri, 08 Jun 2007 02:21:12 GMT
View Forum Message <> Reply to Message

Andrew Morton wrote:
>
> I'd have hoped to see containerstats.c in here.
>

The current statistics code is really small, so it fit into taskstats.c.
May be in the future, we could re-factor it and move it out.

>> +#ifndef _LINUX_CONTAINERSTATS_H
>> +#define _LINUX_CONTAINERSTATS_H
>> +
>> +#include <linux/taskstats.h>
>
> I don't understand the relationship between containerstats and taskstats.
> afacit it's using the same genetlink channel?
>

I've registered containerstats_ops with the same family as taskstats to
reuse the taskstats interface.

>> +enum {
>> + CONTAINERSTATS_CMD_UNSPEC = __TASKSTATS_CMD_MAX, /* Reserved */
>
> This seems to mean that the containerstats commands all get renumbered if
> we add new taskstats commands. That would be bad?
>

As per comment above, since we register containerstats_ops with the taskstats
family, the commands need to be unique, hence we start where taskstats ended
(left off)

>> + */
>> +int containerstats_build(struct containerstats *stats, struct dentry *dentry)
>> +{
>> + int ret = -EINVAL;
>> + struct task_struct *g, *p;
>> + struct container *cont, *root_cont;
>> + struct container *src_cont;
>> + int subsys_id;
>> + struct containerfs_root *root;
>> +
>> + /*
>> + * Validate dentry by checking the superblock operations
>> + */

```
>> + if (dentry->d_sb->s_op != &container_ops)
>> +   goto err;
>> +
>> + ret = 0;
>> + src_cont = (struct container *)dentry->d_fsdata;
>
> Unneeded cast.
>
```

Will remove

```
>> + rcu_read_lock();
>> +
>> + for_each_root(root) {
>> +  if (!root->subsys_bits)
>> +   continue;
>> +  root_cont = &root->top_container;
>> +  get_first_subsys(root_cont, NULL, &subsys_id);
>> +  do_each_thread(g, p) {
>
> this needs tasklist_lock?
>
```

rcu_read_lock() should be fine. From Eric's patch at

2.6.17-mm2 - proc-remove-tasklist_lock-from-proc_pid_readdir.patch

The patch mentions that "We don't need the tasklist_lock to safely
iterate through processes anymore."

```
>> +   cont = task_container(p, subsys_id);
>> +   if (cont == src_cont) {
>> +    switch (p->state) {
>> +    case TASK_RUNNING:
>> +     stats->nr_running++;
>> +     break;
>> +    case TASK_INTERRUPTIBLE:
>> +     stats->nr_sleeping++;
>> +     break;
>> +    case TASK_UNINTERRUPTIBLE:
>> +     stats->nr_uninterruptible++;
>> +     break;
>> +    case TASK_STOPPED:
>> +     stats->nr_stopped++;
>> +     break;
>> +    default:
>> +     if (delayacct_is_task_waiting_on_io(p))
>> +      stats->nr_io_wait++;
```

>> +   break;
>> +   }
>> +   }
>> + } while_each_thread(g, p);
>> + }
>> + rcu_read_unlock();
>> +err:
>> + return ret;
>> +}
>> +
>> static int cmppid(const void *a, const void *b)
>> {
>>   return *(pid_t *)a - *(pid_t *)b;
>> diff -puN kernel/sched.c~containers-taskstats kernel/sched.c
>> --- linux-2.6.22-rc2-mm1/kernel/sched.c~containers-taskstats 2007-06-05 17:21:57.000000000
+0530
>> +++ linux-2.6.22-rc2-mm1-balbir/kernel/sched.c 2007-06-05 17:21:57.000000000 +0530
>> @@ -4280,11 +4280,13 @@ void __sched io_schedule(void)
>> {
>>   struct rq *rq = &__raw_get_cpu_var(runqueues);
>>
>> + delayacct_set_flag(DELAYACCT_PF_BLKIO);
>>   delayacct_blkio_start();
>
> Would it be suitable and appropriate to embed the delayacct_set_flag() call
> inside delayacct_blkio_start()?
>

Yes, I should have done that, will do.

--
Warm Regards,
Balbir Singh
Linux Technology Center
IBM, ISTL