Subject: Re: [PATCH v3 3/6] expose fine-grained per-cpu data for cpuacct stats
Posted by Paul Turner on Wed, 30 May 2012 12:48:40 GMT
View Forum Message <> Reply to Message

On Wed, May 30, 2012 at 5:20 AM, Glauber Costa <glommer@parallels.com> wrote:
> On 05/30/2012 03:24 PM, Paul Turner wrote:

```
>>>
>>> +static int cpuacct_stats_percpu_show(struct cgroup *cgrp, struct cftype
>>> *cft,
>>> >  +                               struct cgroup_map_cb *cb)
>>> >  +{
>>> >  +       struct cpuacct *ca = cgroup_ca(cgrp);
>>> >  +       int cpu;
>>> >  +
>>> >  +       for_each_online_cpu(cpu) {
>>> >  +               do_fill_cb(cb, ca, "user", cpu, CPUTIME_USER);
>>> >  +               do_fill_cb(cb, ca, "nice", cpu, CPUTIME_NICE);
>>> >  +               do_fill_cb(cb, ca, "system", cpu, CPUTIME_SYSTEM);
>>> >  +               do_fill_cb(cb, ca, "irq", cpu, CPUTIME_IRQ);
>>> >  +               do_fill_cb(cb, ca, "softirq", cpu, CPUTIME_SOFTIRQ);
>>> >  +               do_fill_cb(cb, ca, "guest", cpu, CPUTIME_GUEST);
>>> >  +               do_fill_cb(cb, ca, "guest_nice", cpu,
>>> > CPUTIME_GUEST_NICE);
>>> >  +       }
>>> >  +
```

>>
>> I don't know if there's much that can be trivially done about it but I
>> suspect these are a bit of a memory allocation time-bomb on a many-CPU
>> machine. The cgroup:seq_file mating (via read_map) treats everything
>> as/one/ record. This means that seq_printf is going to end up
>> eventually allocating a buffer that can fit_everything_ (as well as
>>
>> every power-of-2 on the way there). Adding insult to injury is that
>> that the backing buffer is kmalloc() not vmalloc().
>>
>> 200+ bytes per-cpu above really is not unreasonable (46 bytes just for
>> the text, plus a byte per base 10 digit we end up reporting), but that
>> then leaves us looking at order-12/13 allocations just to print this
>> thing when there are O(many) cpus.
>>
>
> And how's /proc/stat different ?
> It will suffer from the very same problems, since it also have this very
> same information (actually more, since I am skipping some), per-cpu.

So,

a) the information in /proc/stat is actually much denser since it's

"cpu VAL VAL VAL VAL" as opposed to "cpuX.FIELD VAL"
b) If it became a problem the /proc/stat case is actually fairly
trivially fixable by defining each cpu as a record and "everything
else" as a magic im-out-of-cpus value.

>
> Now, if you guys are okay with a file per-cpu, I can do it as well.
> It pollutes the filesystem, but at least protects against the fact that this
> is kmalloc-backed.
>

As I prefaced, I'm not sure there's much that can be trivially done
about it. This is really a fundamental limitation of how read_map()
works.

What we really need is a proper seq_file exposed through cftypes.