Subject: Re: [RFC] kernel/pid.c pid allocation wierdness
Posted by ebiederm on Wed, 14 Mar 2007 16:54:07 GMT
View Forum Message <> Reply to Message

William Lee Irwin III <wli@holomorphy.com> writes:

> On Wed, Mar 14, 2007 at 08:12:35AM -0600, Eric W. Biederman wrote:
>> If we do dig into this more we need to consider a radix_tree to hold
>> the pid values. That could replace both the pid map and the hash
>> table, gracefully handle but large and small pid counts, might
>> be a smidgin simpler, possibly be more space efficient, and it would
>> more easily handle multiple pid namespaces. The downside to using a
>> radix tree is that is looks like it will have more cache misses for
>> the normal pid map size, and it is yet another change that we would
>> need to validate.
>
> Radix trees' space behavior is extremely poor in sparsely-populated
> index spaces. There is no way they would save space or even come close
> to the current space footprint.

Possibly. We aren't that sparsely populated when it comes to pids.
Hash tables aren't good at saving space either, and when they are space
efficient they are on the edge of long hash chains so they are on
the edge of performance problems. There is at least one variant of the
fib tree that is as space efficient as any binary tree but works by
looking at bits. Not that I think that one makes much sense.

> Lock contention here would be a severe functional regression (note
> "functional," not "performance;" the lock contention surrounding these
> affairs takes down systems vs. mere slowdown nonsense), so it would
> necessarily depend on lockless radix tree code for correctness.

I don't know about the existing in kernel implementations but there is no
reason we could not have an rcu protected radix tree. At which point the
challenges are about the same but we have indexes that would help us find
the next free bit, which could reduce cpu time.

The current pid implementation does not scale to larger process counts
particularly well. The hash table size is fixed so we get a lot of hash
collisions etc. Several other things go wonky as well. The one time
I actually had 64K pids on a system (most of them zombies) it was not
a very pleasant situation.

It isn't common that we push the pid count up, and with the normal pid
counts the current data structures seem very well suited to the
problem. I have been looking at the data structures though in case it
ever changes because the current good behavior seems quite fragile.
Not that I am advocating changing anything yet, but I'm thinking about

it so when we do come to the point where it matters we can make a
reasonable change.

> The comment block describing the hashtable locking is stale and should
> have been updated in tandem with the RCU changes.

Feel free to submit the patch. I didn't make the RCU changes just took
advantage of them.

Eric

_____

Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers