Subject: Re: [RFC] kernel/pid.c pid allocation wierdness
Posted by ebiederm on Wed, 14 Mar 2007 14:12:35 GMT
View Forum Message <> Reply to Message

Pavel Emelianov <xemul@sw.ru> writes:

> Hi.
>
> I'm looking at how alloc_pid() works and can't understand
> one (simple/stupid) thing.
>
> It first kmem_cache_alloc()-s a strct pid, then calls
> alloc_pidmap() and at the end it taks a global pidmap_lock()
> to add new pid to hash.
>
> The question is - why does alloc_pidmap() use at least
> two atomic ops and potentially loop to find a zero bit
> in pidmap? Why not call alloc_pidmap() under pidmap_lock
> and find zero pid in pidmap w/o any loops and atomics?
>
> The same is for free_pid(). Do I miss something?

Well as far as I can tell that is just the way the code
evolved.

Looking at the history. At the time I started messing with it
alloc_pidmap was the function and it behaved pretty much as it
does today with locking (except it didn't disable irqs).

To add the allocation of struct pid. I added alloc_pid
as a wrapper. Left alloc_pidmap alone, and added the hash
table manipulation code. I know this results is fairly
short hold times which is moderately important for a global lock.

We loop in alloc_pidnmap because of what we are trying to do. Simply
returning the first free pid number would have bad effects on user
space, so we have the simple requirement that we don't reuse pid
numbers for as long as is practical. We achieve that doing full walks
through the pid space before we consider a pid again. So we have to
start from the last place we looked. In addition we may have
multiple pages of bitmap to traverse (when our pid limit is high) and
those pages are not physically contiguous.

So while I wouldn't call alloc_pidmap perfect it does seem to be
reasonable.

>From what I can tell for the low number of pids that we usually have
the pid hash table seems near optimal.

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.

Eric

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

Subject: Re: [RFC] kernel/pid.c pid allocation wierdness
Posted by Oleg Nesterov on Wed, 14 Mar 2007 15:33:41 GMT
View Forum Message <> Reply to Message

On 03/14, Eric W. Biederman wrote:
> Pavel Emelianov <xemul@sw.ru> writes:
>
> > Hi.
> >
> > I'm looking at how alloc_pid() works and can't understand
> > one (simple/stupid) thing.
> >
> > It first kmem_cache_alloc()-s a strct pid, then calls
> > alloc_pidmap() and at the end it taks a global pidmap_lock()
> > to add new pid to hash.

We need some global lock. pidmap_lock is already here, and it is
only used to protect pidmap->page allocation. Iow, it is almost
unused. So it was very natural to re-use it while implementing
pidrefs.

> > The question is - why does alloc_pidmap() use at least
> > two atomic ops and potentially loop to find a zero bit
> > in pidmap? Why not call alloc_pidmap() under pidmap_lock
> > and find zero pid in pidmap w/o any loops and atomics?

Currently we search for zero bit lockless, why do you want
to do it under spin_lock ?

Oleg.

Subject: Re: [RFC] kernel/pid.c pid allocation wierdness
Posted by xemul on Fri, 16 Mar 2007 10:57:39 GMT
View Forum Message <> Reply to Message

Oleg Nesterov wrote:
> On 03/14, Eric W. Biederman wrote:
>> Pavel Emelianov <xemul@sw.ru> writes:
>>
>>> Hi.
>>>
>>> I'm looking at how alloc_pid() works and can't understand
>>> one (simple/stupid) thing.
>>>
>>> It first kmem_cache_alloc()-s a strct pid, then calls
>>> alloc_pidmap() and at the end it taks a global pidmap_lock()
>>> to add new pid to hash.
>
> We need some global lock. pidmap_lock is already here, and it is
> only used to protect pidmap->page allocation. Iow, it is almost
> unused. So it was very natural to re-use it while implementing
> pidrefs.
>
>>> The question is - why does alloc_pidmap() use at least
>>> two atomic ops and potentially loop to find a zero bit
>>> in pidmap? Why not call alloc_pidmap() under pidmap_lock
>>> and find zero pid in pidmap w/o any loops and atomics?
>
> Currently we search for zero bit lockless, why do you want
> to do it under spin_lock ?

Search isn't lockless. Look:

```
while (1) {
  if (!test_and_set_bit(...)) {
     atomic_dec(&nr_free);
     return pid;
  }
  ...
}
```

we use two atomic operations to find and set a bit in a map.

> Oleg.
>
>

_____

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