Subject: Re: [Lxc-devel] Q: Do systems using containers user more process ids?
Posted by dev on Mon, 14 Aug 2006 09:12:12 GMT
View Forum Message <> Reply to Message

We have not seen any degradation here in real cases,
but probably you are right and pid hash can be allocated taking into account
physical memory as it is done for TCP/ip/other hashes?

But not sure, it is worth bothering right now... Maybe it worth first to make some
simple test, say:

1. run 50,000 tasks.
2. run some benchmark

and compare benchmark results with different hash sizes?
What do you think?

Kirill

> When looking at the linux pid hash table it is clearly tuned
> to a small number of processes < 4096. If there are more then
> that it starts to degrade with a worst case of 1024 entries
> on each hash chain when pid_max is pushed up to 4*1024*1024.
>
> If more process ids are common then it may be the case that
> we need to fix that data structure so it scales more gracefully.
>
> Eric
>
>
> ---------- ----------
> Using Tomcat but need to do more? Need to support web services, security?
> Get stuff done quickly with pre-integrated technology to make your job easier
> Download IBM WebSphere Application Server v.1.0.1 based on Apache Geronimo
> http://sel.as-us.falkag.net/sel?cmd=lnk&kid=120709&b id=263057&dat=121642
> _____
> Lxc-devel mailing list
> Lxc-devel@lists.sourceforge.net
> https://lists.sourceforge.net/lists/listinfo/lxc-devel
>

Subject: Re: Q: Do systems using containers user more process ids?
Posted by ebiederm on Mon, 14 Aug 2006 21:01:49 GMT
View Forum Message <> Reply to Message

Kirill Korotaev <dev@sw.ru> writes:

> We have not seen any degradation here in real cases,
> but probably you are right and pid hash can be allocated taking into account
> physical memory as it is done for TCP/ip/other hashes?

It is but it is currently capped at 4K entries.
With 4K entries and 32K pids our worst case is usage is a hash chain
9 entries long. At 4M pids our hash chains are 1000 entries long, which
sucks.

> But not sure, it is worth bothering right now... Maybe it worth first to make
> some
> simple test, say:
>
> 1. run 50,000 tasks.
> 2. run some benchmark
>
> and compare benchmark results with different hash sizes?
> What do you think?

If it is easy sure. The real point of where things degrade is
past 50K processes though.

The practical question is if systems using containers are using noticeably
more pids than anyone else. So far the responses I have gotten indicate
that users aren't. So at least until we descend into multi-core madness
it sounds like the current structures are fine, but it might be worth moving
the cap on the number of pid hash table entries at some point in the future.

Eric

Subject: Re: [Containers] Q: Do systems using containers user more process ids?
Posted by Dave Hansen on Mon, 14 Aug 2006 21:18:34 GMT
View Forum Message <> Reply to Message

On Mon, 2006-08-14 at 15:01 -0600, Eric W. Biederman wrote:
> The practical question is if systems using containers are using noticeably
> more pids than anyone else. So far the responses I have gotten indicate
> that users aren't. So at least until we descend into multi-core madness
> it sounds like the current structures are fine, but it might be worth moving
> the cap on the number of pid hash table entries at some point in the future.

Since it is already resized at boot-time, I can't imagine this be a real
problem to fix. I assume you're just trying to see if anybody has run
into it as of yet.

Perhaps a one-time-per-boot warning in find_pid() if the chains get too
long would be nice to have. It wouldn't give us detailed performance

measurements, but it would be a nice canary in the mine in case
something goes horribly wrong.

What about something like this?

---

 lxc-dave/kernel/pid.c | 9 +++++++++
 1 files changed, 9 insertions(+)

```
diff -puN Makefile~warn-on-long-pidhash-chains Makefile
diff -puN kernel/pid.c~warn-on-long-pidhash-chains kernel/pid.c
--- lxc/kernel/pid.c~warn-on-long-pidhash-chains 2006-08-14 14:13:39.000000000 -0700
+++ lxc-dave/kernel/pid.c 2006-08-14 14:17:49.000000000 -0700
@@ -209,9 +209,18 @@ struct pid * fastcall find_pid(int nr)
 {
  struct hlist_node *elem;
  struct pid *pid;
+ int chain_length = 0;
+ static int chain_length_limit = 5;
+ static int issued_warning = 0;

  hlist_for_each_entry_rcu(pid, elem,
    &pid_hash[pid_hashfn(nr)], pid_chain) {
+  if (!issued_warning && (chain_length++ > chain_length_limit)) {
+   issued_warning = 1;
+   printk(KERN_WARN "%s() pid hash chain length "
+      "exceeded %d elements\n",
+      __FUNCTION__, chain_length);
+  }
   WARN_ON(!pid->nr); /* to be removed soon */
   if (pid->nr == nr)
    return pid;
_
```

-- Dave

Subject: Re: Q: Do systems using containers user more process ids?
Posted by ebiederm on Mon, 14 Aug 2006 22:07:02 GMT

View Forum Message <> Reply to Message

Dave Hansen <haveblue@us.ibm.com> writes:

> On Mon, 2006-08-14 at 15:01 -0600, Eric W. Biederman wrote:
>> The practical question is if systems using containers are using noticeably
>> more pids than anyone else. So far the responses I have gotten indicate

>> that users aren't. So at least until we descend into multi-core madness
>> it sounds like the current structures are fine, but it might be worth moving
>> the cap on the number of pid hash table entries at some point in the future.
>
> Since it is already resized at boot-time, I can't imagine this be a real
> problem to fix. I assume you're just trying to see if anybody has run
> into it as of yet.

More or less. There is some other work that needs to be done and I'm
trying to see if reworking the data structure make sense. Not having
a hash table would be nice in the container case.

> Perhaps a one-time-per-boot warning in find_pid() if the chains get too
> long would be nice to have. It wouldn't give us detailed performance
> measurements, but it would be a nice canary in the mine in case
> something goes horribly wrong.
>
> What about something like this?

If we put the canary on pushing pid_max up I'm all for it.
Our current hash is even enough and our set of pids small enough
that just knowing pid_max we can easily calculate the worst case hash
chain length, by brute force.

Eric

Subject: Re: Re: Q: Do systems using containers user more process ids?
Posted by dev on Tue, 15 Aug 2006 08:13:03 GMT
View Forum Message <> Reply to Message

>>We have not seen any degradation here in real cases,
>>but probably you are right and pid hash can be allocated taking into account
>>physical memory as it is done for TCP/ip/other hashes?
>
>
> It is but it is currently capped at 4K entries.
> With 4K entries and 32K pids our worst case is usage is a hash chain
> 9 entries long. At 4M pids our hash chains are 1000 entries long, which
> sucks.
4M pids are almost unreal in production systems.
(only if you spawn these tasks to make them sleep forever :)))).
we usually have no more than 20,000 tasks (which is 200VEs with 100 tasks in each)

>>But not sure, it is worth bothering right now... Maybe it worth first to make
>>some
>>simple test, say:

>>
>>1. run 50,000 tasks.
>>2. run some benchmark
>>
>>and compare benchmark results with different hash sizes?
>>What do you think?
>
>
> If it is easy sure. The real point of where things degrade is
> past 50K processes though.
>
> The practical question is if systems using containers are using noticeably
> more pids than anyone else. So far the responses I have gotten indicate
> that users aren't. So at least until we descend into multi-core madness
> it sounds like the current structures are fine, but it might be worth moving
> the cap on the number of pid hash table entries at some point in the future.
containers are using noticeably more pids, I think it is not a doubt...
the question is whether it is worth doing something here _now_...

Kirill

Subject: Re: Re: [Containers] Q: Do systems using containers user more process ids?
Posted by dev on Tue, 15 Aug 2006 08:15:20 GMT
View Forum Message <> Reply to Message

> Perhaps a one-time-per-boot warning in find_pid() if the chains get too
> long would be nice to have. It wouldn't give us detailed performance
> measurements, but it would be a nice canary in the mine in case
> something goes horribly wrong.
>
> What about something like this?
I think it will be hit from time to time just because
of the hash function and pid distribution :)))

Kirill

Subject: Re: Q: Do systems using containers user more process ids?
Posted by ebiederm on Tue, 15 Aug 2006 18:32:48 GMT
View Forum Message <> Reply to Message

Kirill Korotaev <dev@sw.ru> writes:

>>>We have not seen any degradation here in real cases,
>>>but probably you are right and pid hash can be allocated taking into account

>>>physical memory as it is done for TCP/ip/other hashes?
>>
>>
>> It is but it is currently capped at 4K entries.
>> With 4K entries and 32K pids our worst case is usage is a hash chain
>> 9 entries long. At 4M pids our hash chains are 1000 entries long, which
>> sucks.
> 4M pids are almost unreal in production systems.
> (only if you spawn these tasks to make them sleep forever :)))).
> we usually have no more than 20,000 tasks (which is 200VEs with 100 tasks in
> each)

Ok. That is a reasonable upper bound. I need to look and see what a heavily
loaded sever looks like.

>> The practical question is if systems using containers are using noticeably
>> more pids than anyone else. So far the responses I have gotten indicate
>> that users aren't. So at least until we descend into multi-core madness
>> it sounds like the current structures are fine, but it might be worth moving
>> the cap on the number of pid hash table entries at some point in the future.
> containers are using noticeably more pids, I think it is not a doubt...
> the question is whether it is worth doing something here _now_...

True. The break even point in terms of cache misses between a hash chain
and a binary tree is between 8 and 16 levels deep. Currently it looks like
we are close to that point but not over on a heavily loaded system.

So until we cross that line it probably doesn't make sense to change
the implementation, unless it makes multiple pid namespaces easier to
implement, and it doesn't look like it will significantly help that
case.

Eric