Subject: Re: [PATCH v6 02/10] ipc: "use key as id" functionality for resource get system ca
Posted by ebiederm on Mon, 15 Oct 2012 19:47:40 GMT
View Forum Message <> Reply to Message

ebiederm@xmission.com (Eric W. Biederman) writes:

> Stanislav Kinsbursky <skinsbursky@parallels.com> writes:
>
>> This patch introduces new IPC resource get request flag IPC_PRESET, which
>> should be interpreted as a request to try to allocate IPC slot with number,
>> starting from value resented by key. IOW, kernel will try
>> allocate new segment in specified slot.
>>
>> Note: if desired slot is not emply, then next free slot will be used.
>
> This way of handling things is pretty nasty.
>
> - You don't fail if the requested id is not available.
> - You don't allow assigning the key (which leads to the need to change
>   the key in later patches). Changing the creator uid and creator
>   gid and key is semantically ugly.
>
> It would be much cleaner if you could instead add IPC_PRESET and then
> extend the definition of the creation functions all by one argument.
>
> aka
> int msgget(key_t key, int msgflg, int id);
> int semget(key_t key, int nsems, int semflg, int id);
> int shmget(key_t key, size_t size, int shmflg, int id);
>
> Where the extra id argument is ignored unless IPC_PRESET is specified.
>
> Also msgget, semget, and shmget should fail if unregconized flags are
> passed in. That ipcget doesn't do that today is bizarre.

Hmm. Come to think of it I don't see why you need to set the id at all.
We are using an idr allocator which effectively offers the semantics
that the lowest available id will be allocated. The same semantics we
have for file descriptors.

So it should be possible at least for the first pass at
checkpoint/restart to implement the restoration of sysv ipc without
IPC_PRESET at all.

So IPC_PRESET should just be an optimization, not a necessary feature.

That makes all of your code go away except the message queue

peeking, which seems much less intrusive for the first pass.

Eric

Subject: Re: [PATCH v6 02/10] ipc: "use key as id" functionality for resource get system ca
Posted by Stanislav Kinsbursky on Tue, 16 Oct 2012 07:55:09 GMT
View Forum Message <> Reply to Message

> ebiederm@xmission.com (Eric W. Biederman) writes:
>
>> Stanislav Kinsbursky <skinsbursky@parallels.com> writes:
>>
>>> This patch introduces new IPC resource get request flag IPC_PRESET, which
>>> should be interpreted as a request to try to allocate IPC slot with number,
>>> starting from value resented by key. IOW, kernel will try
>>> allocate new segment in specified slot.
>>>
>>> Note: if desired slot is not emply, then next free slot will be used.
>>
>> This way of handling things is pretty nasty.
>>
>> - You don't fail if the requested id is not available.
>> - You don't allow assigning the key (which leads to the need to change
>>   the key in later patches). Changing the creator uid and creator
>>   gid and key is semantically ugly.
>>
>> It would be much cleaner if you could instead add IPC_PRESET and then
>> extend the definition of the creation functions all by one argument.
>>
>> aka
>> int msgget(key_t key, int msgflg, int id);
>> int semget(key_t key, int nsems, int semflg, int id);
>> int shmget(key_t key, size_t size, int shmflg, int id);
>>
>> Where the extra id argument is ignored unless IPC_PRESET is specified.
>>
>> Also msgget, semget, and shmget should fail if unregconized flags are
>> passed in. That ipcget doesn't do that today is bizarre.
>
> Hmm. Come to think of it I don't see why you need to set the id at all.
> We are using an idr allocator which effectively offers the semantics
> that the lowest available id will be allocated. The same semantics we
> have for file descriptors.
>
> So it should be possible at least for the first pass at

> checkpoint/restart to implement the restoration of sysv ipc without
> IPC_PRESET at all.
>
> So IPC_PRESET should just be an optimization, not a necessary feature.
>

CRIU was designed to suspend/restore not only containers with it's own IPC
namespace, but also for single process and process tree.
So we have to restore IPC objects with proper id and key.

> That makes all of your code go away except the message queue
> peeking, which seems much less intrusive for the first pass.
>
> Eric
>

--
Best regards,
Stanislav Kinsbursky

Subject: Re: [PATCH v6 02/10] ipc: "use key as id" functionality for resource get system ca
Posted by ebiederm on Tue, 16 Oct 2012 09:03:29 GMT
View Forum Message <> Reply to Message

Stanislav Kinsbursky <skinsbursky@parallels.com> writes:

>> Hmm. Come to think of it I don't see why you need to set the id at all.
>> We are using an idr allocator which effectively offers the semantics
>> that the lowest available id will be allocated. The same semantics we
>> have for file descriptors.
>>
>> So it should be possible at least for the first pass at
>> checkpoint/restart to implement the restoration of sysv ipc without
>> IPC_PRESET at all.
>>
>> So IPC_PRESET should just be an optimization, not a necessary feature.
>>
>
> CRIU was designed to suspend/restore not only containers with it's own IPC
> namespace, but also for single process and process tree.
> So we have to restore IPC objects with proper id and key.

I was not suggesting restoring IPC objects without the proper id and

key.

I was pointing that since the algorithm for id assignment is known, that by simply allocating objects with care you can guarantee that you allocate objects with the proper id and key.

As for the case when you don't have your own ipc namespace, in general that case will fail. Especially with the current id assignment algorithm chances that your id is already taken are very high. So since in general and most of the time it will fail to restore an ipc namespace that is not shared outside of the container there needs to be a compelling reason to support to even consider it.

I am not comfortable with the key<->id association changing on active objects. That gives rise to new and scary races in userspace, and a much more complex set of interactions to analyze to see if the both userspace and the kernel code is correct.

I am not particularly comfortable adding a new allocation pathway for sysv ipc objects when that new pathway appears completely unnecessary.

Stanislav please concentrate on something that works for the easy case of an ipc namespace per container today. Once there is a working checkpoint/restart and people can use it and see what the limitations it becomes much easier to see if extra optimizations like your IPC_PRESET code are worth it.

Eric

Subject: Re: [PATCH v6 02/10] ipc: "use key as id" functionality for resource get system ca
Posted by Stanislav Kinsbursky on Tue, 16 Oct 2012 11:33:50 GMT
View Forum Message <> Reply to Message

> Stanislav Kinsbursky <skinsbursky@parallels.com> writes:
>

>
>>> Hmm. Come to think of it I don't see why you need to set the id at all.
>>> We are using an idr allocator which effectively offers the semantics
>>> that the lowest available id will be allocated. The same semantics we
>>> have for file descriptors.
>>>
>>> So it should be possible at least for the first pass at
>>> checkpoint/restart to implement the restoration of sysv ipc without
>>> IPC_PRESET at all.

>>>
>>> So IPC_PRESET should just be an optimization, not a necessary feature.
>>>
>>
>> CRIU was designed to suspend/restore not only containers with it's own IPC
>> namespace, but also for single process and process tree.
>> So we have to restore IPC objects with proper id and key.
>
> I was not suggesting restoring IPC objects without the proper id and
> key.
>
> I was pointing that since the algorithm for id assignment is known, that
> by simply allocating objects with care you can guarantee that you
> allocate objects with the proper id and key.
>

Does it means, that you suggest to allocate as many objects, as required to get
desired id and then release all, that not needed?

> As for the case when you don't have your own ipc namespace, in general
> that case will fail. Especially with the current id assignment
> algorithm chances that your id is already taken are very high. So since
> in general and most of the time it will fail to restore an ipc namespace
> that is not shared outside of the container there needs to be a
> compelling reason to support to even consider it.
>

The reason is simple: we want at least to have a chance to restore such
processes with IPC objects.
Yes, in general, resource id will be close to 0, and it would be most probably
impossible to restore such resource on another machine, if any software on it
uses IPC objects.
But this doesn't mean, that the implementation is useless.
Imagine, that you have a process tree, sharing some IPC resource. And it's
actually all additional software, that you have running on a hardware node
(except processes, started during boot by default).
Then you have another node with the same configuration. And you want to use them
for load balancing or something like this. I.e. you want to migrate this process
tree from one to another.
In this particular case (which is valid and used widely) this migration will be
always successful.

> I am not comfortable with the key<->id association changing on active
> objects. That gives rise to new and scary races in userspace, and a
> much more complex set of interactions to analyze to see if the both
> userspace and the kernel code is correct.
>

Could you, please, give an example, what scares you?
The case, when some software can change IPC object key, thus making some other
task unable to attach this object by key, is not significant, because task can
destroy object by key already.

> I am not particularly comfortable adding a new allocation pathway for
> sysv ipc objects when that new pathway appears completely unnecessary.
>
> Stanislav please concentrate on something that works for the easy case
> of an ipc namespace per container today. Once there is a working
> checkpoint/restart and people can use it and see what the limitations
> it becomes much easier to see if extra optimizations like your
> IPC_PRESET code are worth it.
>

Sorry, but I don't understand your objection.

> Eric
>

--
Best regards,
Stanislav Kinsbursky