Subject: Re: [RFC PATCH 0/4] Container Freezer: Reuse Suspend Freezer
Posted by Matt Helsley on Sat, 05 Apr 2008 00:54:03 GMT
View Forum Message <> Reply to Message

On Fri, 2008-04-04 at 20:30 -0400, Oren Laadan wrote:
>
> Matt Helsley wrote:

<snip>

> > I don't have an upper limit on how many more states we will need and I
> > think that number impacts the interface significantly. Can you give us
> > an estimate?
>
> In Zap there are (using the current terminology):
> RUNNING - running
> FREEZING - transition to FROZEN
> FROZEN - frozen
> THAWING - transition to RUNNING
> CKPTING - being checkpointed (needs special semantics* for some ops)
> RSTRTING - being restarted (may need special semantics* for some ops)
> ABORTING - transition to DEAD (mainly when aborting a failed restart)
> DEAD - dead (but not fully cleaned up yet)
>
> There is also "SPECIAL" in which some operations are not allowed;
> this simplifies dealing with a bunch of races related to checkpoint/
> restart, but I'm not sure it's a must. If anything, it only stays
> for very short times (like an uninterruptible sleep) saying "don't
> mess with this container now, it's busy".
>
> I have very good justifications for almost all the states, a good
> reasoning for DEAD, and a case for SPECIAL (although there may be
> a way to do without it).
>
> Despite the "many" states, there are very few transitions: CKPTING
> can only be reached from- and changed to- FROZEN. A similar rule holds
> for RSTRTING. ABORTING is reached from RSTRTING, and leads to DEAD.
> The only one I don't cover is reaching DEAD from any other state
> (except SPECIAL) but I never saw a reason to explicitly encode that.
> Something like this (without SPECIAL):
>
> -> FREEZING -> <-> CKPTING
> RUNNING FROZEN
> <- THAWING <- <-> RSTRTING -> ABORTING -> DEAD
>
> Out of curiosity - why does the number of states impact the interface
> so much ?

Only that, if you're trying to keep the interface human readable, then having too many names for things (states and transitions) can make the interface less than intuitive. Doesn't look like that's an issue for the above state machine though so I think you're right -- a single file with transitions encoded as states seems best.

> >
> >> Second, my gut feeling is that a single, atomic operation to get the
> >> status is preferred over multiple (non-atomic) operations. In other
> >> words, I suggest a single state file instead of two. You can encode
> >> every possible transition in a single state. It's not that the kernel
> >
> > If the transitions are to be human-readable and there are more than a
> > small number of states it may not be desirable to encode transitions as
> > states. Paul's reason for suggesting the additional file(s), as best I
> > could tell, was to keep the interface human-readable.
>
> The scheme above has very few transitions. Whatever be the final scheme,
> I would prefer not to have many possible transition (the full matrix).
> It probably isn't necessary either.
>
> The main idea behind limiting the transitions above, is that checkpoint
> requires to first freeze the container to be able to capture a consistent
> view of the state of its processes; that means, for example, that we also
> would like to prevent signals from being delivered to tasks in a frozen
> state (if you do want to signal - thaw it first).
>
> There is also the issue of a pre-checkpoint (a.k.a live migration) where
> significant state (mainly memory) of the container is recorded while the
> container is still running, and when it's finally frozen little state
> remains to be saved, reducing the application downtime. I didn't see a
> need for a special state for this case; instead Zap uses a status flag
> that belongs to the container.
>
> Oren.

Thanks for the explanation of the states.

Cheers,
-Matt Helsley

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers