Subject: containers development plans (July 20 version)
Posted by serge on Fri, 20 Jul 2007 17:36:15 GMT

View Forum Message <> Reply to Message

(If you missed earlier parts of this thread, you can catch earlier parts of
this thread starting at
 https://lists.linux-foundation.org/pipermail/containers/2007 -July/005860.html)

```
======================  Section 0  ======================
=Status of this document
======================  Section 0  ======================
```

I've added a 'use cases' section. That is where we attempt to
explain to people not familiar with containers work why it is
worth integrating upstream.

Srivatsa Vaddagiri is independently gathering additional information
on specific task container subsystems. That will eventually be
incorporated into the final version of this roadmap.

```
======================  Section 1  ======================
=Introduction
======================  Section 1  ======================
```

We are trying to create a roadmap for the next year of
'container' development, to be reported to the upcoming kernel
summit. Containers here is a bit of an ambiguous term, so we are
taking it to mean all of:

 1. namespaces
        kernel resource namespaces to support resource isolation
        and virtualization for virtual servers and application
        checkpoint/restart.
 2. task containers framework
        the task containers (or, as Paul Jackson suggests, resource
        containers) framework by Paul Menage which especially
        provides a framework for subsystems which perform resource
        accounting and limits.
 3. checkpoint/restart

```
======================  Section 2  ======================
=Detailed development plans
======================  Section 2  ======================
```

A (still under construction) list of features we expect to be worked on
next year looks like this:

    1. completion of ongoing namespaces

        - pid namespace
            - push merged patchset upstream
            - kthread cleanup
                - especially nfs
                - autofs
            - af_unix credentials (stores pid_t?)
        - net namespace
        - ro bind mounts

2. continuation with new namespaces
    - devpts, console, and ttydrivers
    - user
    - time
    - namespace management tools
    - namespace entering (using one of:)
        - bind_ns()
        - ns container subsystem
        - (vs refuse this functionality)
    - multiple /sys mounts
        - break /sys into smaller chunks?
        - shadow dirs vs namespaces
    - multiple proc mounts
        - likely need to extend on the work done for pid namespaces
        - i.e. other /proc files will need some care

virtualization of statistics for 'top', etc

3. any additional work needed for virtual servers?
    - i.e. in-kernel keyring usage for cross-usernamespace permissions, etc
        - nfs and rpc updates needed?
        - general security fixes
            - per-container capabilities?
        - device access controls
            - e.g. root in container should not have access to /dev/sda by default)
        - filesystems access controls

4. task containers functionality
    - base features
        - virtualized continerfs mounts
            - to support vserver mgmnt of sub-containers
        - locking cleanup
        - control file API simplification
        - control file prefixing with subsystem name

userpace RBCE to provide controls for

users

groups

pgrp

executable

- specific containers
    - split cpusets into
        - cpuset

memset
network
connect/bind/accept controller using iptables
network flow id control
userspace per-container OOM handler
per-container swap
per-container disk I/O scheduling

5. checkpoint/restart
memory c/r
(there are a few designs and prototypes)
(though this may be ironed out by then)
per-container swapfile?
overall checkpoint strategy (one of:)
in-kernel
userspace-driven
hybrid
overall restart strategy
use freezer API
use suspend-to-disk?
sysvipc
"set identifier" syscall
pid namespace
clone_with_pid()

===================== Section 3 =====================
=Use cases
===================== Section 3 =====================

1, Namespaces:

The most commonly listed uses for namespaces are virtual servers and checkpoint restart. Other uses are debugging (running tests in not-quite-virtual-servers) and resource isolation, such as the use of mounts namespaces to simulate multi-level directories for LSPP.

2. Task Containers:

(Vatsa to fill in)

3. Checkpoint/restart

load balancing:
applications can be migrated from high-load systems to ones with a lower load. Long-running applications can be checkpointed (or migrated) to start a short-running high-load job, then

restarted.

kernel upgrades:
A long-running application - or whole virtual server - can
be migrated or checkpointed so that the system can be
rebooted, and the application can continue to run

===================== Section 4 =====================
=Involved parties
===================== Section 4 =====================

In the list of stakeholders, I try to guess based on past comments and
contributions what *general* area they are most likely to contribute in.
I may try to narrow those down later, but am just trying to get something
out the door right now before my next computer breaks.

Stakeholders:
Eric Biederman
everything
google
task containers
ibm (serge, dave, cedric, daniel)
namespaces
checkpoint/restart
bull (benjamin, pierre)
namespaces
checkpoint/restart
ibm (balbir, vatsa)
task containers
kerlabs
checkpoint/restart
openvz
everything
NEC Japan (Masahiko Takahashi)
checkpoint/restart
Linux-VServer
namespaces+containers
zap project
checkpoint/restart
planetlab
everything
hp
(i must have lost an email - what are they
interested in working on?)
XtreemOS
checkpoint/restart
Fujitsu/VA Linux Japan

resource control

Is anyone else still missing from the list?

thanks,
-serge