

Hello All,

Some of us will meet next week for the first mini-summit on containers.
Many thanks to Alasdair Kergon and LCE for the help they provided in making this mini-summit happen !

It will be held on Monday the 3rd of September from 9:00 to 12:45 at LCE in room D. We also might get a phone line for external participants and, if not, we should be able to set up a skype phone.

Here's a first try for the Agenda.

Global items

[let's try to defer discussion after presentation]

- * Pavel Emelianov status update
- * Serge E. Hallyn Container Roadmap including
 - . task containers (Paul Menage)
 - . resource management (Srivatsa Vaddagiri)

Special items

[brainstorm sessions which we would like to focus on]

- * building the global container object ('a la' openvz or vserver)
- * container user space tools
- * container checkpoint/restart

Thanks,

C.

===== 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

task containers provide a framework for subsystems which associate
state with arbitrary groups of processes, for purposes such as
resource control/monitoring.

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-namespace 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
- 'container object'?
- implementation (perhaps largely userspace abstraction)
- container enter
- container list
- container shutdown notification

- 4. task containers functionality
 - base features
- hierarchical/virtualized containers
 - support vserver mgmnt of sub-containers
 - locking cleanup
 - control file API simplification
- userpace RBCE to provide controls for
 - users
 - groups
 - pgrp
 - executable
 - specific containers targeted:
 - split cpusets into
 - cpuset
 - memset
 - network
 - connect/bind/accept controller using iptables
- memory controller (see detail below)
- cpu controller d (see detail below)
- io controller (see detail below)
 - network flow id control
 - per-container OOM handler (userspace)
- per-container swap
- per-container disk I/O scheduling
- per container memory reclaim
- per container dirty page (write throttling) limit.
- network rate limiting (outbound) based on container

misc

User level APIS to identify the resource limits that is allowed to a job, for example, how much physical memory a process can use. This should seamlessly integrated with non-container environment as well (may be with ulimit).

Per container stats, like pages on active list, cpus usage, etc

memory controller

users and requirements:

1. The containers solution would need resource management (including memory control and per container swap files).

Paul Menage, YAMOMOTO Takshi, Peter Zijlstra, Pavel Emelianov have all shown

interest in the memory controller patches.

2. The memory controller can account for page cache as well, all people interested in limiting page cache control, can theoretically put move all page cache hungry applications under the same container.

Planned enhancements to the memory controller

1. Improved shared page accounting
2. Improved statistics
3. Soft-limit memory usage

generic infrastructure work:

1. Enhancing containerstats
 - a. Working on per controller statistics
 - b. Integrating taskstats with containerstats
2. CPU accounting framework
 - a. Migrate the accounting to be more precise

cpu controller

users and requirements:

1. Virtualization solutions like containers and KVM need CPU control. KVM for example would like to have both limits and guarantees supported by a CPU controller, to control CPU allocation to a particular instance.
2. Workload management products would like to exploit this for providing guaranteed cpu bandwidth and also (hard/soft) limiting cpu usage.

work items

1. Fine-grained proportional-share fair-group scheduling.
2. More accurate SMP fairness
3. Hard limit
4. SCHED_FIFO type policy for groups
5. Improved statistics and debug facility for group scheduler

io controller

users and requirements:

1. At a talk presented to the Linux Foundation (OSDL), the attendees showed interest in an IO controller to control IO bandwidth of various filesystem operations (backup, journalling, etc)

work items:

1. Proof of concept IO controller and community discussion/feedback
2. Development and Integration of the IO controller with containers

open issues

1. Automatic tagging/resource classification engine

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()
- live migration

===== 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
 - network namespaces, virtual servers?
- XtreemOS
 - checkpoint/restart
- Fujitsu/VA Linux Japan
 - resource control
- BLCR (Paul H. Hargrove)
 - checkpoint/restart

Is anyone else still missing from the list?

thanks,
-serge

Containers mailing list
Containers@lists.linux-foundation.org

