Subject: Container mini-summit notes
Posted by Cedric Le Goater on Wed, 05 Sep 2007 14:07:07 GMT
View Forum Message <> Reply to Message

Held at De Vere Universty Arms Hotel, Cambridge, UK

* Monday, Sept 3, 9h00 to 16h00 :

Kir Kolyshkin <kir@openvz.org>
Pavel Emelianov <xemul@openvz.org>
Masahiko Takahashi <masahiko@linux-foundation.org>
Oren Laadan <orenl@cs.columbia.edu>
James Youngman <youngman@google.com>
??? (NTT)
Cedric Le Goater <clg@fr.ibm.com>

On the phone (skype with very high noise level)

Paul Menage <menage@google.com>
Srivatsa Vaddagiri <vatsa@in.ibm.com>
Dhaval Giani <dhaval@linux.vnet.ibm.com>
Vaidyanathan Srinivasan <svaidy@in.ibm.com>

* Tuesday, Sept 4, 15h00 to 18h00 :

Pavel Emelianov <xemul@openvz.org>
Paul Menage <menage@google.com>
Eric W. Biederman <ebiederm@xmission.com>
Cedric Le Goater <clg@fr.ibm.com>

= Namespace status
==========

* sysv ipc

extend to posix mqueue.
. check that /dev/mqueue can be mounted multiple times
. mqueue sysctls will need a fix :
fs.mqueue.queues_max
fs.mqueue.msg_max
fs.mqueue.msgsize_max

* uname namespace

considered complete.

what about being able to set the kernel version ?

* user

useful today to current container technologies (openvz, vserver)

uid checks should be replaced by (uid, userns) to complete
integration with filesystems
security needs to be looked at
so is signal delivery

* pid namespace

in dev

signal handling completion underway
pid_t cleanups
 . the purpose is to remove any explicit reference to
    task->pid
. keep ->pid in task struct only for performance
. complex cleanups ones:
    af_unix credentials
    file locks
    timer stat

kthread cleanup
 . replace kernel_thread() by the kthread API
. change core kthread API to support signals
. then nfs needs extra love. is someone working on it ?

 do we need hierarchical levels ?

* net

in dev

veth is in dmiller's tree
sysfs cleanups underway in greg's tree
eric is working on a mininal patchset acceptable for netdev. will
ask dmiller advice on the topic

ip isolation could be done with netfilter or security hooks

* device namespace

to do

we don't want to get rid of mknod() but we also want to limit the

view of the devices in a container. one way to do this is through a
device namespace which would only expose a 'white list' of devices
when unshared. a possible 'white list' is :

    /dev/null
    /dev/full
    /dev/zero
    /dev/rtc
    /dev/random
    /dev/pts/*

do we require a extra namespace for /dev/pts/* to handle its
virtualization or can this be done directly in the device namespace ?

check that /dev/pts can be mounted multiple times.

* time

to do

required for C/R
will only make sense in a "closed" environment
the purpose is to keep the monotonic timers from expiring when
you restart

* other possible namespace ?

rtc ? which is an isolation issue and also a sysctl issue

comment from eric :
  a redesign of lsm, a la netfilter, could cover all
isolation needs.

* namespace management

. entering

  no consensus on how this should be done.

  probably because the need is related to a container and not just
  namespaces. it should be solved with a container object and
  probably a subsystem.

  serge's proposal of sys_hijack() is interesting but will require
  more study because, in UNIX, it's not natural for a child process
  to have 2 parents !

. extending clone to support more flags

new syscall proposal for a clone2(struct clone2_arg_struct* args)

* tests

. ltp for unit
. keep the integration tests in each container framework.

* Filesystems

. unprivilege mounts (not addressed)

merged

. multiple /sys mounts (in dev)

missing some bits (eric working on it) to decouple sysfs and
kobjects

. multiple /proc mounts (to complete)

multiple mount done
to limit access to /proc files, use the user namespace checks ?
for the contents of each file, use the current context to identify
namespace

* Console

. a running getty should be covered by tty namespace
. printk will require some support to be isolated.

= Task Container (from container dev plan)
==========

* base features

hierarchical/virtualized containers
support vserver mgmnt of sub-containers
locking cleanup
control file API simplification
unified container including namespaces

the "container"/"task container" name is ambiguous and it should change to
"control group"

* 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

controllers :

memory controller (see detail below)

cpu controller

Status:
- Extensions required to CFS core for supporting group-scheduling aspects are mostly there (in mainline)

Todo:
- Better SMP group-fairness
- Hard-limit cpu usage
- SCHED_FIFO like policy for groups
- Group priorities (?)

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

= Resource Management (from container dev plan)
==========

* 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 cahce control, can theoratically 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 precis

* 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

= Checkpoint/Restart
==========

* need to unified the freezer to reach a quiescence point

* overall strategy :
. checkpoint: in kernel
. restart : first recreate process tree then let each
process restart itself

* possible direction for C/R user api
. checkpoint/restart syscalls
. C/R file systems
solves the set id issue
elegant but exposes too much the ABI

example :

```
.
|-- 0x00003002
|   |-- 0x00003002
|   |   |-- attr
|   |   |-- signal
|   |   |-- signal.altstack
```

```
|   |   |-- signal.pending
|   |   |-- thread
|   |   |-- thread.frame
|   |   |-- timers
|   |   |-- tls
|   |   `-- wait.zombies
|   |-- aio
|   |-- attr
|   |-- fds
|   |-- ldt
|   |-- mem.segments
|   |-- numa
|   |-- process
|   |-- signal.action
|   |-- signal.pending
|   |-- sysv.semadj
|   |-- sysv.shmcount
|   `-- thread.list
|-- af_inet_listening
|-- af_inet_orphan_count
|-- af_inet_orphan_data
|-- af_inet_orphan_info
|-- files
|   |-- 0
|   |-- 1
|   |-- 10137663680
|   |-- 1014250cdc0
|   |-- 2
|   `-- stdios
|-- sysv.msq
|-- sysv.sem
`-- sysv.shm
```

* memory C/R

  critical for performance
  per-container swapfile ?

* subsystem C/R API.

  keep it on the side for the moment <subsys>_cr.c to identify the
  needs of each subsystem before asking the maintainer's comments

  possible cr_ops in some objects (like for network protocols) but
  also ops 'a la' virt_ops to prepare for different C/R strategy :
  brutal, incremental, live migration

* setting id back to what they where

possible global syscall to set ids of pid,ipc,pts.
else use the C/R fs

* statefile format

no big issues. let's pick one.

* optimization

parallel C/R

_____

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