Subject: Re: Container mini-summit notes
Posted by serue on Wed, 05 Sep 2007 14:38:49 GMT
View Forum Message <> Reply to Message

Quoting Cedric Le Goater (clg@fr.ibm.com):
...

> Container mini-summit notes v0.01
>
> ==
>
> Held at Universty Arm's, Cambridge, UK
>
> on 3rd of September, 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>
> Cedric Le Goater <clg@fr.ibm.com>
> James ??? (Google)
> ??? (NTT)
>
> On the phone (skype with very high noise level)
>
> Srivatsa Vaddagiri <vatsa@in.ibm.com>
> Paul Menage <menage@google.com>
>
> on 4th of September, 15h00 to 18h00 :
>
> Pavel Emelianov <xemul@openvz.org>
> Cedric Le Goater <clg@fr.ibm.com>
> Paul Menage <menage@google.com>
> Eric W. Biederman <ebiederm@xmission.com>

Boy I really missed out didn't I...

> = Namespace status
> ==
>
> * ipc
>
> extend to posix mqueue.
> check that /dev/mqueue can be mounted multiple times
> mqueue sysctls need a fix
> fs.mqueue.queues_max
> fs.mqueue.msg_max
> fs.mqueue.msgsize_max

>
>
> * uname namespace
>
> considered complete but what about setting the kernel version ?

I still say yuck. The kernel won't properly emulate the other kernel version, so lying seems a horrible thing to do. If you want a vm with another kernel, use qemu/kvm/xen.

> * user (to complete)
>
> useful today to current container technologies (userns, vserver)
> (uid, userns) checks must be completed to be integrate
>   with filesystems
> security needs to be looked at
> so is signal delivery
>
> * pid namespace (in dev)
>
> stable.
>
> Current tasks are :
>   . signal handling
>   . pid_t cleanups
>   remove any reference to task->pid
>   keep ->pid in task struct only for performance
>   complex ones:
>   af_unix credentials
>   file locks
>   timer stat
>   . kthread cleanup
>   replace kernel_thread() by the kthread API
>   change core kthread API to support signals
>   nfs needs extra love. is someone working on it ?

REALLY would like to hear if anyone is.

>   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

And has by several people :)

> * 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 a device namespace ?

How the hell does a separate device ns protect you from mknod?

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

I previously sent a patch to implement CLONE_NEWPTS, which just
intercepts /dev/pts/ access without multiple mounts. I think that,
plus access to child containers' devpts mounts through the nsproxy
container subsystem, would be a very simple and useful way to
implement this.

Then there is also the array of console devices to turn into a
namespace.

And finally there is the issue of actualy virtual devices. So I want to
start up a virtual server, and I want it's serial port to be connected
to a virtual tty which buffers the last page or few pages of data so I
can connect to the server and see what's going on. I'm not too savvy
here - can we do that now, maybe just through a fake serial device?

> * time (to do)
>
> keep the monotonic timers from expiring when you restart
> required for C/R but will only make sense in "close" environment
>
> * other possible namespace ?
>
> rtc ? which an isolation problem and also a sysctl issue

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

I'd like to talk to Eric and find out more precisely what he is
thinking.

Note that one issue with using lsm is that it will purely control
access, not resource ids. So container 1 may not have access to
/dev/pts/2, which at least is safe, but it'll see everyone's devices
under devpts.

But yes, this is why I've been saying that userns is useful at the
moment. You should be able to write simple selinux policies to fully
container a userns.

> * namespace management
>
> . entering
>   no consensus on how this should be 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)

>
> . 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

Just might work...

It kind of looks like having a 'initial' userns which is special-cased,
but I assume the idea is that /proc/ files would be created at boot in
the initial userns, so straightforward checks for userns equivalence
between the writing process and the file would suffice.

> * Console
>
> . a running getty should be covered by tty namespace

I'm not understanding this line.

Is there a fully tty namespace design that was discussed, which this is
summarizing?

> . 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
>
> * 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 (see detailbelow)
>   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

The data transfer method is pretty important to sort out, so that people
can start writing patches to c/r various resources.

I know you've discussed having some sort of 'crfs'. I gather you would
then do a checkpoint by doing something like
 cp /mnt/crfs/vserver1 /opt/checkpoints/vserver1/`date +%F`/`date +%R`

I actually like that, so long as it's that, and not actually having the
cr subsystem write the data straight to a regular file.

Another option would be to use relayfs for the checkpoint data dump,
since checkpoint and restart will be assymetrical in nature anyway.

>   * 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 ?

Absolutely. per-container swapfile, and memory checkpoint forces write to swap and writes the pte data out to relayfs or a crfs file ready for copy.

One tricky issue then is that we really want to do a copy-on-write copy of the swapfile. We don't want to wait for the swapfile to be fully copied before restarting the process. Are any of the fancy new filesystems implementing that? Or are we gonna be using zfs over fuse for our per-container swapfiles? :)

>   * 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

It would be great if we could provide a data write/read api so these subsystems can all be written without worrying about which method we'll use for dumping checkpoint data and pushing restart data.

>   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

Hmm.

>   * 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