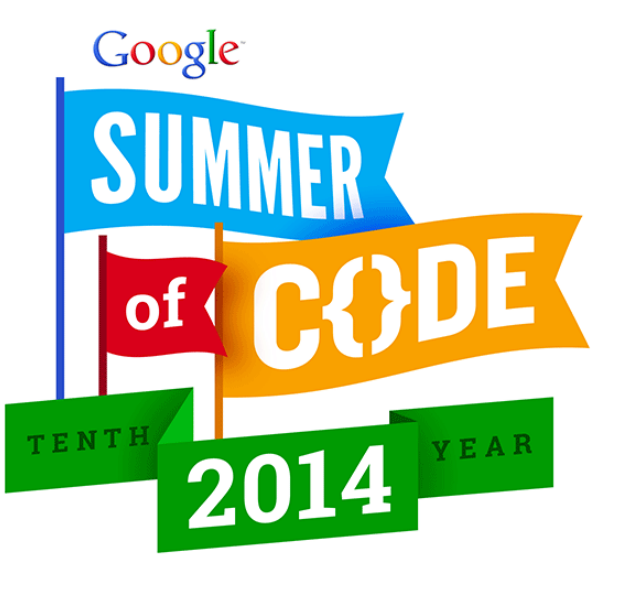
LXC on Ganeti

Yuto Kawamura(kawamuray)
Email: kawamuray.dadada@gmail.com

Google summer of Code

- Global program that offers students stipends to write code for open source projects
- 4420 students
- 6313 proposals
- (don’t know how much of them were accepted)

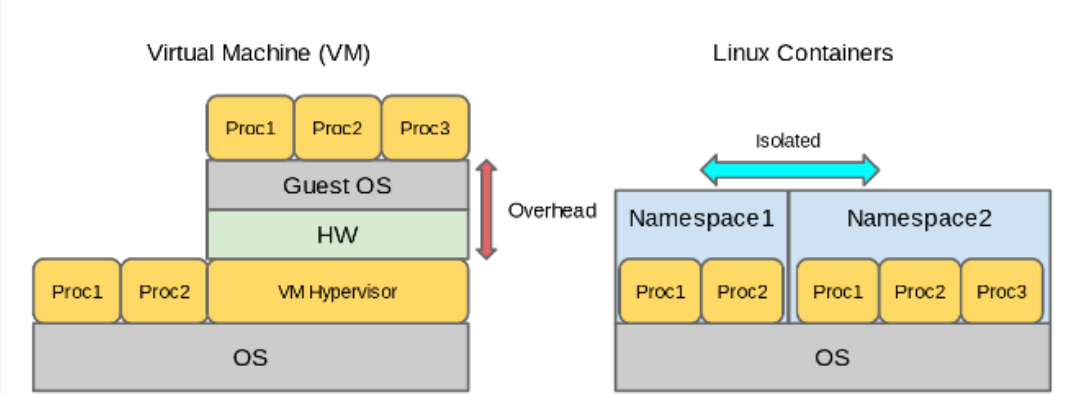


Improve LXC support

- Project accepted by Ganeti on Google Summer of Code 2014
- Objective
  - Improve LXC support on Ganeti
  - Fix existing broken implementation
  - Add unit tests
  - Setup QA
  - Add additional features
- Mentored by Hrvoje Ribicic

LXC

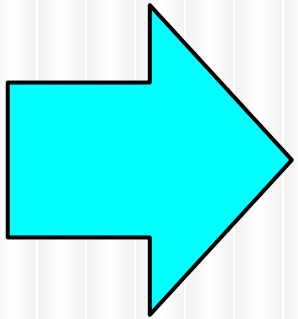
- An userspace interface for the Linux kernel containment features
- Operating System level virtualization
- Composed by two Linux kernel features
  - Namespace - Isolation for pid, ipc, mount, net, user between containers and the host
  - Cgroup - Hardware resource limitation



Before and After

× Console
× Reboot(soft)
× Memory ballooning
✔ Network(veth)
× Migration(stopped)
× Migration(live)
△ Verification

LXC version: any



✔ Console
✔ Reboot(soft)
✔ Memory ballooning
✔ Network(veth)
△ Migration(stopped)
× Migration(live)
✔ Verification

LXC version: >= 1.0.0

Parameters

- cpu_mask: Cpu mask. Same as other hypervisors but no support for vcpus pinning
- lxc_cgroup_use: Cgroup subsystems list. Correspond to lxc.cgroup.use configuration parameter
- lxc_devices: The list of devices ACL.
- lxc_drop_capabilities: List of capabilities which should be dropped for an instance.
- lxc_tty: Number of ttys(ptys) created for an instance.
- lxc_startup_wait: Time to wait until a daemonized instance state changed to RUNNING.

Parameters

- lxc_extra_config: For extra configuration parameters
  - Separating by comma is problem
  - Planning to introduce a functionality to read the value from a file

How to try?

1. Enable cgroup subsystems(if needed)
   a. cpuset, memory, devices, cpuacct
   b. Debian: Need “cgroup_enable=memory” in kernel argument
2. Install LXC >= 1.0.0
3. Check your kernel configuration

```
$ sudo lxc-checkconfig
--- Namespaces ---
Namespaces: enabled
Utsname namespace: enabled
Ipc namespace: enabled
Pid namespace: enabled
User namespace: enabled
Network namespace: enabled
Multiple /dev/pts instances:
enabled ...
```

How to try?

4.
```
gnt-cluster modify \
  --enabled-hypervisors=kvm,lxc \       # enable lxc hv
  --nic-parameters mode=bridge,link=br0 # your brdige interface
```

5.
```
gnt-cluster verify # Checks your environment by LXC hypervisor
```

6.
```
gnt-instance add \
  -H lxc \                              # use LXC hypervisor!
  -o debootstrap+default \
  -t drbd -s 8G lxc-instance1.ganeti
```

↑ will automatically modify the fstab of the instance to make it proper for the LXC container(riba is working for it)

It’s all! Easy right?

Tips - How to debug?

- Log file
  - $PREFIX/var/log/ganeti/lxc/<instance name>.<instance uuid>.log
    - Logs from lxc-start command
    - Mainly used when your instance didn’t started
  - $PREFIX/var/run/ganeti/lxc/<instance name>.console
    - Logs from processes inside the container(init, rc, etc.)
    - See when you cannot use `gnt-instance console …`

Implementations

Basic policy

- LXC containers are managed by command line tools
  - lxc-start, lxc-stop, lxc-ls, lxc-info …
- Not using lxc python library(written in python3)

Instance Start

1. lxc-start -n <instance name> …
   -f $PREFIX/var/run/ganeti/lxc/<instance name>.conf
2. lxc-wait -n <instance name> -s RUNNING
   a. with timeout specified by the lxc_startup_wait hvparam

Instance Stop

- Soft shutdown
  - lxc-stop --nokill -n <instance name>
    - Send SIGPWR to the init
      - Overwritable with lxc.haltsignal
    - Wait until init exits

- Hard(forced) shutdown
  - lxc-stop --kill -n <instance name>
    - Send SIGKILL to the init
    - Container killed without normal shutting down process

Instance Reboot

- Soft reboot
  - lxc-stop --reboot -n <instance name>
    - Send SIGINT to the init
  - Requires CAP_SYS_BOOT
- Hard rebot
  - Same as other hypervisors
    - soft shutdown && start

Memory ballooning

- Use Cgroup memory subsystem
- memory.limit_in_bytes
- memory.memsw.limit_in_bytes
- Shrinking memory will fail if the kernel can’t reclaim memory from processes inside the container

Cgroup filesystem

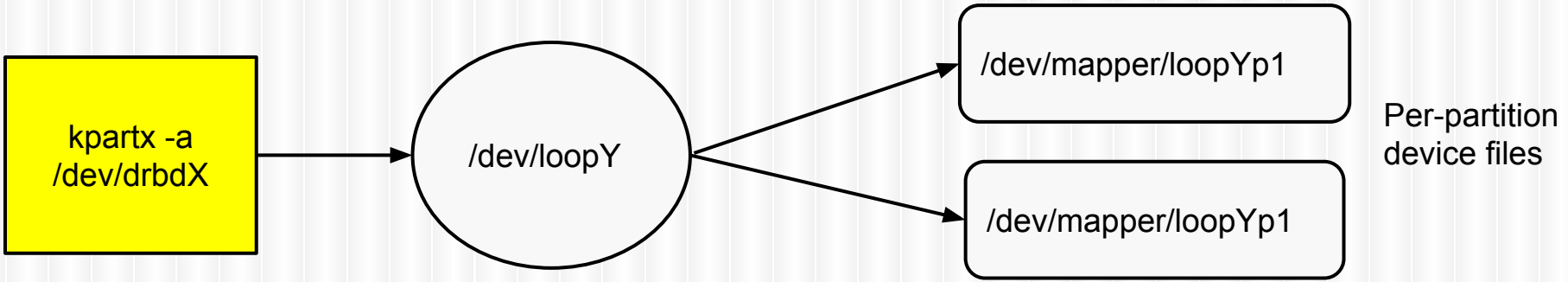
- Differences between distributions/rc system

1. Search /proc/mounts for line having type == cgroup && <subsystem> in options
2. If not found, do: mount -t cgroup -o <subsystem> <subsystem> $PREFIX/var/run/lxc/cgroup/<subsystem>

Storage mount

- LXC can’t handle raw storage device
- Need to prepare per-pertition(mountable) device file for a LXC rootfs

- Use kpartx(1)
  - Create device maps for over partitions segments detected
  - Suppose every environment have it(because OS scripts uses it)



Storage mount

- Use first partition as a rootfs
  - lxc.rootfs = /dev/mapper/loopNp1
- You may have to increase the number of max_loop

Quality

- Tested on Debian Linux/Gentoo Linux with LXC 1.0.0 and 1.0.5
- Passed basic QA

but …

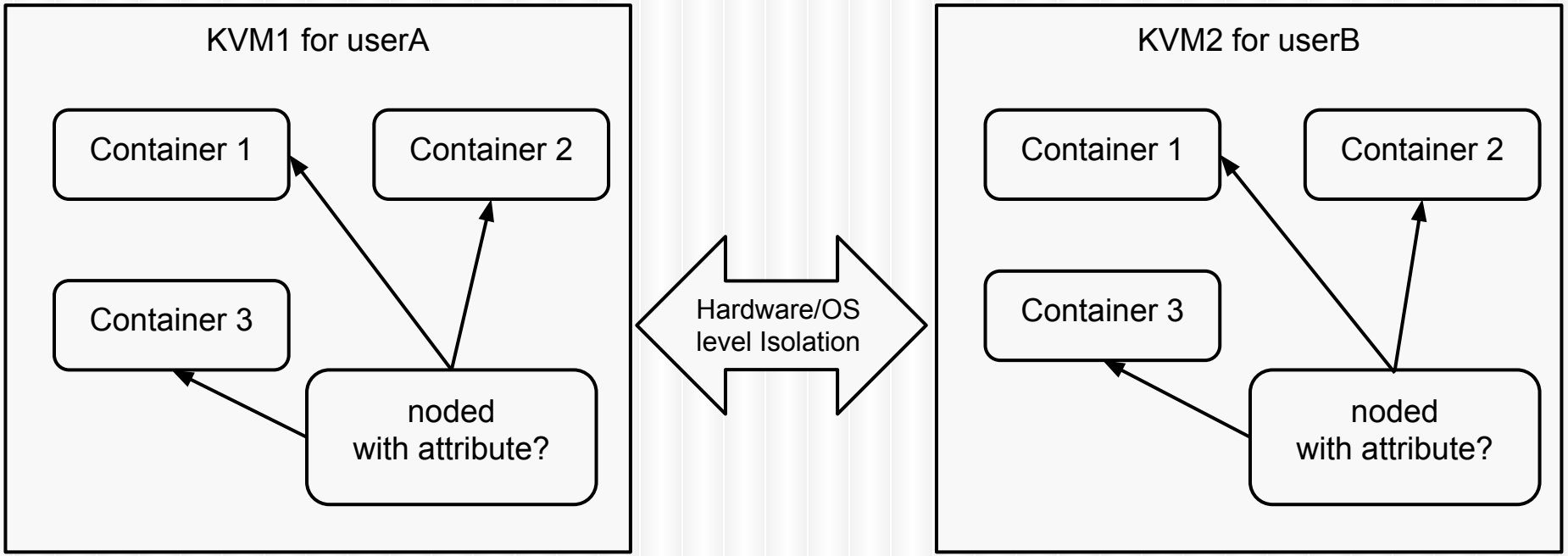
- Need more example of use
- Possibly contains platform(distribution) related problems
  - especially around the cgroup

What’s next?

Future candidates

- Two level isolation(IOW, containers inside a Virtual Machine)
- LXC instance migration

Two level isolation



Instance Migration

- LXC has no migration functionality
- Have to implement it on our own

Checkpoint/Restore

- Implemented by project CRIU (http://criu.org/)
- Able to dump a running processes into serialized images
  - Serialized in protocol buffers format

```
/path/to/checkpoint/dir
  \----- Memory image
  \----- Namespaces info
  \----- File descriptors
  \----- Cgroup info
  \----- TCP sockets
  \----- Network informations
```

lxc-checkpoint

[lxc-devel] [PATCH] Add support for checkpoint and restore via CRIU

Tycho Andersen tycho.andersen at canonical.com
Wed Aug 20 03:14:03 UTC 2014

- Previous message: [lxc-devel] default root password has to be random in default debian template
- Next message: [lxc-devel] [PATCH] Add support for checkpoint and restore via CRIU
- **Messages sorted by:** [date] [thread] [subject] [author]

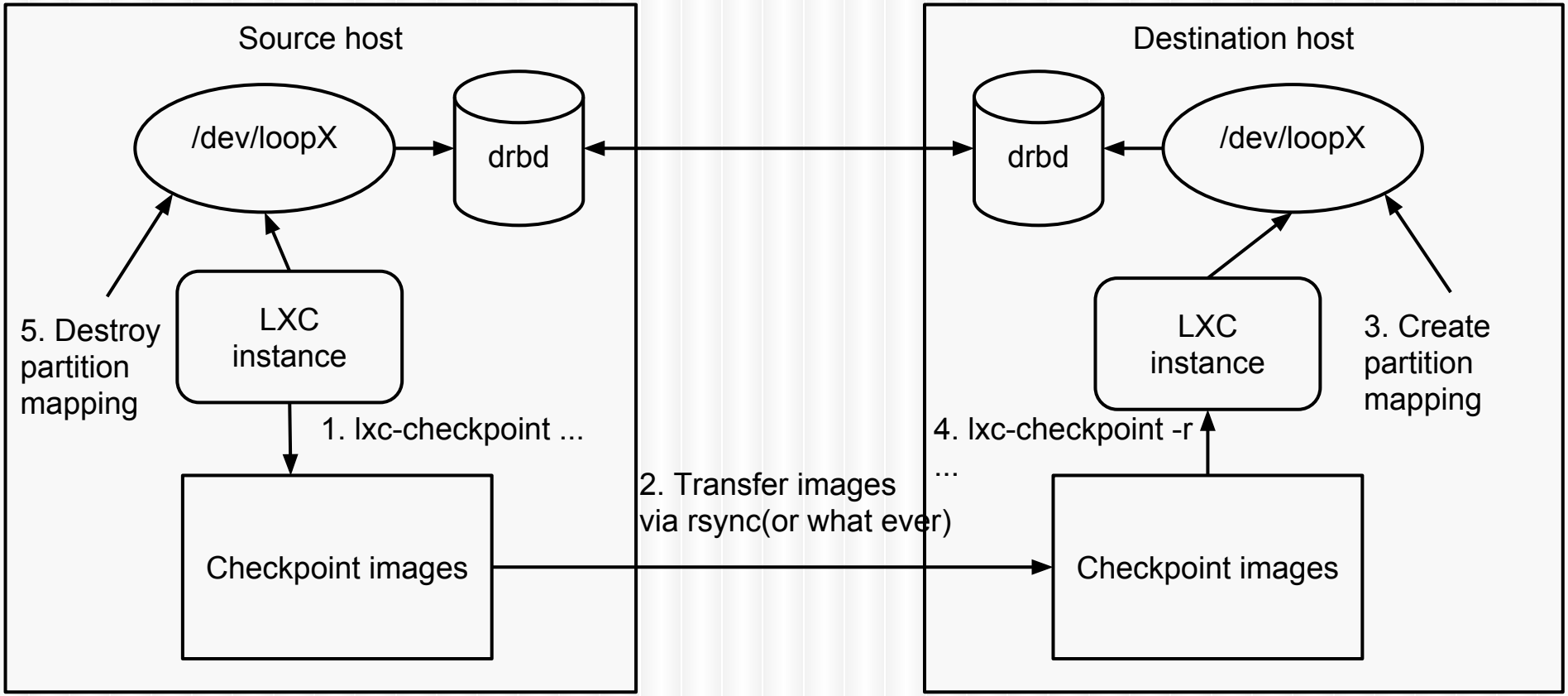
This patch adds support for checkpointing and restoring containers via CRIU.
It adds two api calls, ->checkpoint and ->restore, which are wrappers around
the CRIU CLI. CRIU has an RPC API, but reasons for preferring exec() are
discussed in [1].

To checkpoint, users specify a directory to dump the container metadata (CRIU
dump files, plus some additional information about veth pairs and which
bridges they are attached to) into this directory. On restore, this
information is read out of the directory, a CRIU command line is constructed,
and CRIU is exec()d. CRIU uses the lxc-restore-net callback (which in turn
inspects the image directory with the NIC data) to properly restore the
network.

This will only work with the current git master of CRIU; anything as of
a152c843 should work. There is a known bug where containers which have been
restored cannot be checkpointed [2].

[1]: http://lists.openvz.org/pipermail/criu/2014-July/015117.html
[2]: http://lists.openvz.org/pipermail/criu/2014-August/015876.html

LXC migration



Instance migration

- Already have working implementation
  - https://github.com/kawamuray/ganeti/commits/topic-lxc-migration
- but …
  - its temporaily implementation
  - still have lot of restrictions
  - requires patched criu
    - https://github.com/kawamuray/criu
  - requires patched lxc
    - https://github.com/kawamuray/lxc
- will be solved soon(by sending patches to originals and by fixing implementation)

Demo

We really want

- Example of use
  - on many platforms(Debian, RHEL6/7, …)
- Bug report
- Realistic use case
- Desired functionality
  - possibly I can handle it from both Ganeti and LXC aspect

Special thanks to Riba

For your great mentorship!

Questions?
Opinions?
Ideas?
Requests?