Debugging the Virtualization layer (libvirt and QEMU) in OpenStack

Kashyap Chamarthy <kchamart@redhat.com>

OpenStack Summit
Tokyo 2015

Part I
Problem background and overview

Problem background

- Lots of moving parts: OpenStack services, Virt drivers, System components, etc
- Tracking interactions between multiple components is challenging
- Finding relevant log patterns in complex systems can become cumbersome

$\rightsquigarrow$ Effective root cause analysis with right tooling

What *kind* of bugs?

- Unexpected guest crashes
- Heisenbugs! (e.g. Nova bug: #1334398)
- Bugs introduced by load (e.g. OpenStack CI infra: ~800 test jobs/hr[*])
- Subtle issues in complex features

  (e.g. live migration), perf. degradation

[*] http://status.openstack.org/zuul/

OpenStack Nova

- Compute workloads
- Pluggable Virtualization drivers

  ```
  [libvirt]
  virt_type=kvm|qemu|xen|[...]
  ...
  ```

- `nova-compute`: faciliates interactions between hypervisors (libvirt/KVM) & VMs, via the virt driver interface

KVM Virtualization building blocks

KVM – Linux hardware virt (`vmx`|`svm`)

QEMU – Emulator: Devices (disk, networks, display, sound, PCI, etc); CPU

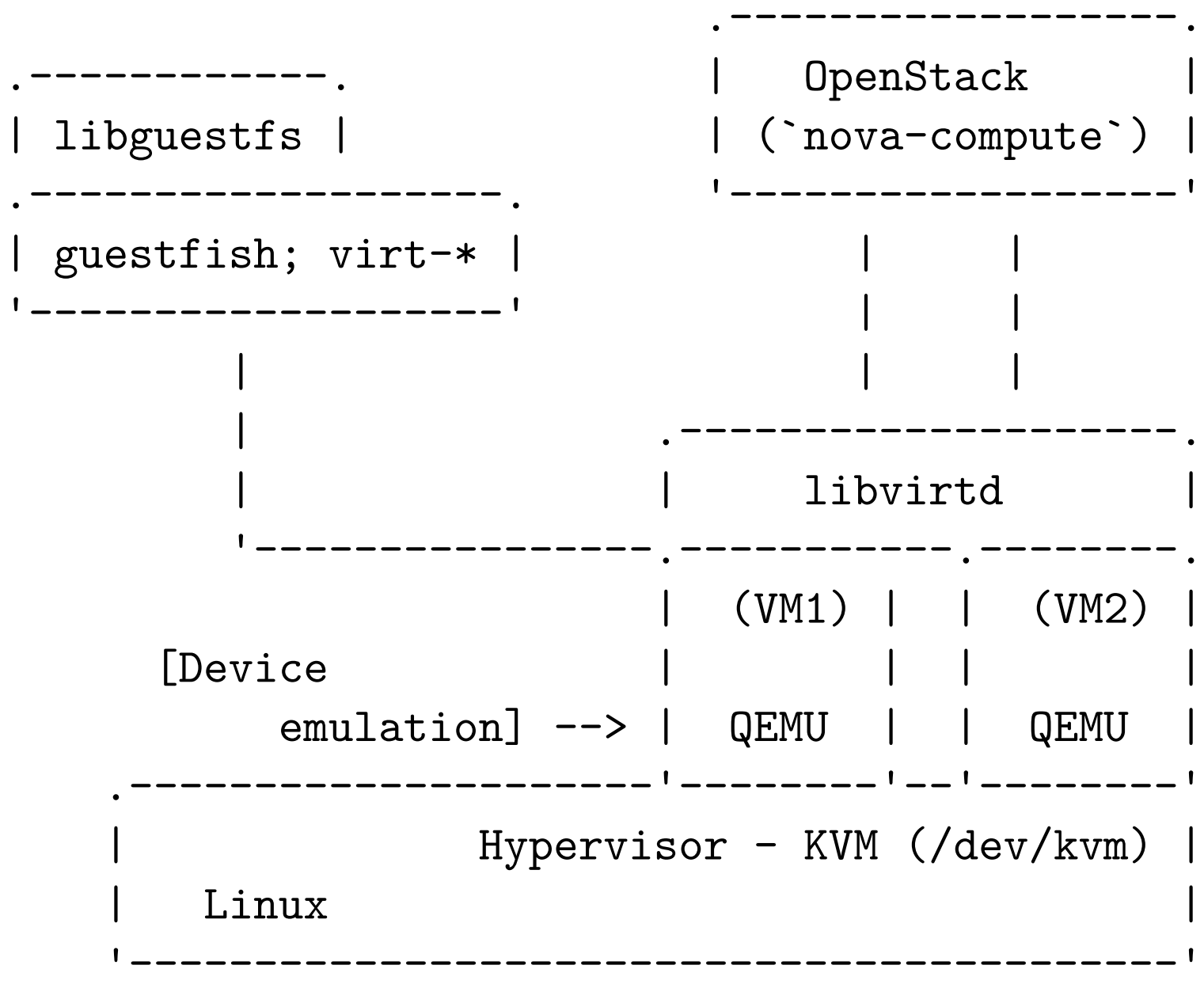
```
$ qemu-system-x86_64 -device \?
$ qemu-system-x86_64 -cpu \?
```

– Interactions with libvirt: **QMP** JSON RPC interface, **command-line**

libvirt – Hypervisor agnostic virtualization library

$\rightsquigarrow$ Default virtualization drivers in OpenStack

OpenStack KVM Virtualization building blocks

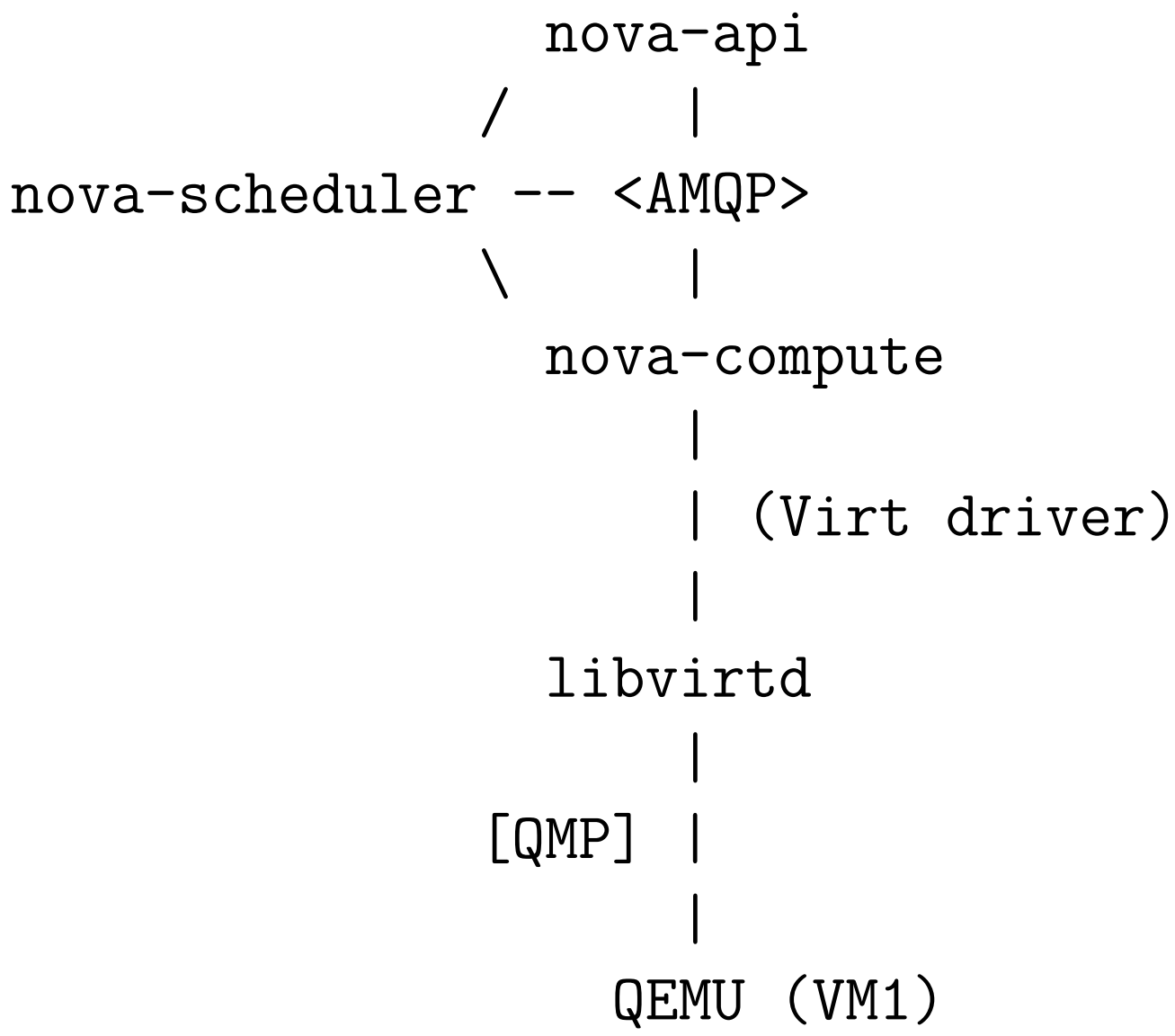


redhat.

Part II

OpenStack Compute debugging utilities

Nova, libvirt & QEMU



Debugging utilities (in no order)

Component	Tools
Nova	debug/verbose = True
Compute	"Guru Meditation" Error Reports
libvirt	log_filters, virsh, journalctl
QEMU	QMP query commands, mpstat, coredumpctl, `gdb -p $QEMU-PID`
KVM	kvm_stat(1) perf(1), trace-cmd(1), tcpdump(1)

Oslo "Guru Meditation" Error Reports (1)

To get live error report (will be redirected to stderr) of a Nova Compute process:

```
$ kill -s USR1 `pgrep nova-compute`
```

- `SIGUSR1`, `SIGUSR2` == User-defined signals
  - Refer: `man 7 signal`

⇝ From *'Mitaka'* release, default: `USR2`

Oslo "Guru Meditation" Error Reports (2)

Sections of the error report:

- Distribution package versions
- Running processes
- Green Threads, Native Threads
- Nova configuration details

$\rightsquigarrow$ No prior action required by the admin!

Example report:

http://docs.openstack.org/developer/oslo.reports/usage.html

redhat.

Part III

Libvirt and QEMU debugging controls

Virtual Machine specific logs

Located here:

`/var/log/libvirt/qemu/$vm.log`

Contains:

- libvirt-generated QEMU command-line arguments
- QEMU error messages
- libvirt `stderr` is redirected here

Granular logging infrastructure with libvirt

Log messages, filters and outputs.

– A set of patterns & priorities to accept or reject a log message.

E.g. Capture `DEBUG` for QEMU & libvirt but only `WARN` + `ERROR` for the rest.

Libvirt daemon logging: filters, log prorities

In */etc/libvirt/libvirtd.conf*, set:

```
log_filters="1:qemu 1:libvirt 3:security
             3:event 3:util 3:file"
log_outputs="1:file:/var/log/libvirt/libvirtd.log"
```

Restart libvirt daemon:

```
$ systemctl restart libvirtd
```

$\rightsquigarrow$ Better signal-to-noise ratio with log filters

Libvirt library logging: env. variables, outputs

To log all libvirt API calls, `export`:

```
LIBVIRT_DEBUG=1
LIBVIRT_LOG_FILTERS="1:qemu"
LIBVIRT_LOG_OUTPUTS="1:journald 1:file:virsh.log"
```

Specify multiple log outputs:

- systemd `journald`
- `file`

$\rightsquigarrow$ Applicable for libvirt daemon logging, too

Querying systemd journal for libvirt messages(1)

Structured logging with libvirt specifc journal fields:

- `LIBVIRT_SOURCE`, `CODE_FILE`
- `CODE_FUNC`, `CODE_LINE`
- `LIBVIRT_CODE`, `LIBVIRT_DOMAIN`

Querying systemd journal for libvirt messages(2)

Examples:

```
$ journalctl /usr/sbin/libvirtd [-f]

$ journalctl -o verbose \
             /usr/sbin/libvirtd

$ journalctl -u libvirtd -l -p err \
             --since=today
```

Live querying the VM: libvirt primitives

`virsh` command	Summary
qemu-monitor-command	Inspect/Modify VM state
qemu-monitor-event	Observe QMP Events
domblkstat	Fetch device block stats
[. . .]	[. . .]

⇝ For plenty more utilities: **man virsh**

Live querying VM state: `qemu-monitor-command`

- Query (or optionally modify) VM state.
- Enumerate all available QMP commands:

```
$ virsh qemu-monitor-command \
        vm1 --pretty \
        '{"execute":"query-commands"}'
```

Query available QMP commands: `query-commands`

```
$ virsh qemu-monitor-command vm1 -pretty \
        '{"execute":"query-commands"}'
        [...]

            "name": "query-events"
        ,

            "name": "query-cpu-definitions"
        ,

            "name": "drive-mirror"
        ,

            "name": "block-commit"
        ,
        [...]
```

For storage migration

Query block device info: query-block

```
$ virsh qemu-monitor-command vm1 -pretty \
        '{"execute":"query-block"}'
        [...]
            "io-status": "ok",
            "device": "drive-virtio-disk0",
            [...]
                "iops_rd": 0,
                ...
                "image": {
                    "backing-image": {
                        "virtual-size": 3221225472,
                        "filename": "[...]/_base/6b3d28"
                        "format": "raw"
                    ...
                    "virtual-size": 21474836480,
                    "filename": [...]/instances/disk",
                    ...
                    "format": "qcow2",
        [...]
```

Live querying VM: `qemu-monitor-event`

During a live block operation (e.g. in-progress live disk copy/migration), invoke this on a Nova instance:

```
$ virsh qemu-monitor-event \
        instance-00000001 \
        --pretty --loop
```

Prints details of the events as they occur

⇝ Can observe arbitrary QMP events

redhat.

Part IV

Example: Tracing the flow of a guest crash during Nova live block migration

Nova live block migration: Why this example?

- Multiple Nova Compute processes, libvirt daemons
- Thereby, multiple QEMU instances
- Examine commands libvirt requests QEMU to execute (src $\leftrightarrow$ dest)

$\rightsquigarrow$ Observe interactions at different layers

Live block migration: Nova invocation

Invoke the Nova live block migration command:

```
$ nova live-migrate \
        --block-migrate vm1 $DEST-HOST
```

Sets libvirt migration flags as config attributes:

. . .

`live_migration_flag=VIR_MIGRATE_LIVE`,[...]

`block_migration_flag=VIR_MIGRATE_NON_SHARED_INC`,[...]

. . .

[NB: These are default (but configurable), no admin action needed.]

Live block migration: equivalent libvirt invocation

Perform live block migration via libvirt's shell interface:

```
$ virsh migrate -verbose \
        --copy-storage-inc \
        --p2p --live vm1 \
        qemu+ssh://root@dest/system
```

⇝ Nova is making calls to this infrastructure under the hood

Live block migration: libvirt invocation result

`stderr` says. . .

```
$ virsh migrate -verbose \
    --verbose \
    --copy-storage-inc \
    --p2p --live vm1 \
    qemu+ssh://root@dest/system
error: internal error: guest unexpectedly quit
```

Oops!

Guest doesn't run anymore!

Debug VM crash: Inspect for libvirt daemon errors

In libvirt daemon logs, on relevant Compute hosts:

```
$ less /var/log/libvirt/libirtd.log
[...]
error : qemuMonitorIO:662 : internal error:
End of file from monitor
[...]
debug : qemuMonitorIO:738 : Triggering EOF callback
debug : qemuProcessHandleMonitorEOF:307 :
        Received EOF on 0x7f2be0003bb0 'vm1'
debug : qemuProcessHandleMonitorEOF:325 : Monitor
connection to 'vm1' closed without SHUTDOWN event;
assuming the domain crashed
[...]
```

?

Assumption

Debug VM crash: Inspect for QEMU errors

In guest-specific logs, maintained by libvirt:

```
$ tail /var/log/libvirt/qemu/vm1.log
[. . .]
/usr/bin/qemu-kvm -name vm1 -S
-machine pc-i440fx-2.3,accel=kvm,usb=off
-cpu Nehalem -m 1024 -realtime mlock=off
-smp 1,sockets=1,cores=1,threads=1
-drive file=/export/cirros-0.3.3.qcow2,if=none,
 id=drive-virtio-disk0,format=qcow2
[. . .]
Co-routine re-entered recursively
2015-09-28 10:45:26.232+0000: shutting down
```

Error from QEMU

Debug VM crash: Look for Core dumps (1)

Use tools like `coredumpctl` (or equivalent):

```
$ coredumpctl
TIME    PID    UID    GID  SIG  PRESENT EXE
[...]   7194   107    107  11 */usr/bin/qemu-system-x86_64
```

Libvirt's assumption confirmed

Debug VM crash: Look for Core dumps (2)

Extract the coredump for the crashed QEMU process, report/fix bug:

```
$ coredumpctl dump 7194
           PID: 7194 (qemu-system-x86)
           UID: 107 (qemu)
           GID: 107 (qemu)
        Signal: 11 (SEGV)
      . . .
  Command Line: /usr/bin/qemu-system-x86_64 -machine[...]
     Coredump: /var/lib/systemd/coredump/core.qemu[...].xz
       Message: Process 7194 (qemu-system-x86)
                of user 107 dumped core.

             Stack trace of thread 7194:
             #0  0x00007fa52fa4680b __libc_siglongjmp (libc.so.6)
             #1  0x00007fa53d1670c9 longjmp (libpthread.so.0)
             #2  0x00005632def06370 qemu_coroutine_switch (qemu-system-x86_64)
             #3  0x00005632def05a05 qemu_coroutine_enter (qemu-system-x86_64)
             [. . .]
```

Debug VM crash: Root cause, resolution

- Turns out to be a bug (RH#1266936)
  in the guts of QEMU's disk mirroring code
- Fixed upstream:

```
$ git show e424aff

commit e424aff5f307227b1c2512bbb8ece891bb895cef
Author: Kevin Wolf <kwolf@redhat.com>
Date:   Thu Aug 13 10:41:50 2015 +0200

    mirror: Fix coroutine reentrance

    This fixes a regression introduced by commit dcfb3beb ("mirror: Do
    zero write on target if sectors not allocated"), which was reported
    to cause aborts with the message "Co-routine re-entered
    recursively".
    [. . .]
```

References

- Slides
  https://kashyapc.fedorapeople.org/OpenStack-Tokyo-Summit-2015
- "Observability in KVM" by Stefan Hajnoczi
  https://archive.fosdem.org/2015/schedule/event/observability/
- Blog:
  http://kashyapc.com

Thanks for listening.

Questions?

redhat.

Part V

[Backup] What else? Observe interactions between multiple libvirt daemons/QEMU instances

Observe commands sent from src → dest libvirtd

In a *successful case*, QEMU QMP command `drive-mirror` constructed by source libvirt (during live storage migration):

```
$ grep "Send command" /var/log/libvirt/libvirtd| less
[. . .]
debug : qemuMonitorJSONCommandWithFd:290 : Send command

'"execute":"drive-mirror",
    "arguments":"device":"drive-virtio-disk0",
    "target":"nbd:devstack2:49153:exportname=drive-virtio-disk0",
    "speed":8796093022207,"sync":"top","mode":"existing",
    "id":"libvirt-11"'
[. . .]
```

Likewise for **dest → src**