Migration: Trying to make it more robust

Red Hat
Juan Quintela

KVM Forum 2014
Düsseldorf

Abstract

This talk shows what are the things we have done to improve migration to make it less fragile.

Agenda

1. What have we done?
2. What are the future plans?
3. Anything else you need?
4. Questions

redhat.

Section 1

What have we done?

redhat.

Bitmap Logging (GSOC)

Sanidhya: GSOC student

- Dump dirty bitmap to log file
- How much we want to do it
- The period

This allows to study how much one work load is dirtying memory, and have an idea of how long it is going to take to migrate.

redhat.

VMState continuous testing (GSOC)

Sanidhya: GSOC student
What is migration?

- dump memory through the network
- stop guest
- dump device state
- continue on target

redhat.

VMState continuous testing (II)

What if we do in a loop

- stop guest
- dump device state to memory
- reset guest devices
- load device state
- continue

redhat.

VMState continuous testing (III)

Why?

- We have more problems with devices than with memory
- We can repeat this as much as we want, on the same load
- If there is any problem, we really know the state of the device that is causing us problems
- Really the infrastructure was there already
- Problems, as usual, are on the details

redhat.

Migration Checker (Amit)

- Presented on KVM Forum last year
- Integrated since
- Output device state meta data in json
- We can compare the output of two qemus and see if they are the same/compatible

redhat.

VMState Validate (mst)

- Aserts
- To make sure that the values that we got make sense
- how to add the asserts

example

```
VMSTATE_VALIDATE("num_timers_in_range", hpet_validate_num_timers),
```

redhat.

VMState Testing

- there were no test for VMState
- false, Eduardo created the tests for some types
- added tests for every VMState* type
- or I removed it

redhat.

VMState Testing(II)

A simplification? You call this a simplification?

raw diff

```
git diff origin/master | diffstat
.....
 vmstate.c                                  |   98 +
 127 files changed, 2671 insertions(+), 955 deletions(-)
```

redhat.

VMState Testing(III)

We added tests for all VMState macro. If we remove that file, simplification shows

tests

```
wc -l tests/test-vmstate.c
2301 tests/test-vmstate.c
```

redhat.

Section 2

What are the future plans?

redhat.

Generated fields

tests

```
{
        .name = "fpcr",
        .version_id = 0,
        .size = sizeof(uint64_t),
        .info = &vmstate_fpcr,
        .flags = VMS_SINGLE,
        .offset = 0
},
```

redhat.

Generated fields (II)

current

```
#define VMSTATE_INT64(_f, _s, _v)...
```

proposed

```
#define VMSTATE_INT64_G(_f, _s, _v, _read, _write)
```

X

users that can be converted

```
#define VMSTATE_UINT8_EQUAL(_f, _s)                                   \
#define VMSTATE_VALIDATE(....)
```

redhat.

Move to visitors

current

```
void qemu_put_be64(QEMUFile *f, uint64_t v);
```

current

```
fops->put_be64(opaque, fops, v);
```

We already have QEMUFile, writing to a buffer and would be needed for any change in format that we see fit.

redhat.

Format

Put here your preferred rant here.

redhat.

Optional sections

- Patch just posted as RFC
- ... OK, this morning...

redhat.

Optional sections

- Patch just posted as RFC
- ... OK, this morning...

redhat.

One last thing

- We have done it wrong until now!
- What inter-version migration should we be allowing
- current: qemu-old -M pc to qemu-new -M pc
- What is pc?
- Really what we are doing is the equivalent of
- qemu-2.1 -M pc-2.1 to qemu-2.2 -M pc-2.2
- or qemu-2.0 -M pc-2.0 to qemu-2.2 -M pc-2.2
- or .... you get the idea

redhat.

One last thing

- We have done it wrong until now!
- What inter-version migration should we be allowing
- current: qemu-old -M pc to qemu-new -M pc
- What is pc?
- Really what we are doing is the equivalent of
- qemu-2.1 -M pc-2.1 to qemu-2.2 -M pc-2.2
- or qemu-2.0 -M pc-2.0 to qemu-2.2 -M pc-2.2
- or .... you get the idea

redhat.

One last thing

- We have done it wrong until now!
- What inter-version migration should we be allowing
- current: qemu-old -M pc to qemu-new -M pc
- What is pc?
- Really what we are doing is the equivalent of
- qemu-2.1 -M pc-2.1 to qemu-2.2 -M pc-2.2
- or qemu-2.0 -M pc-2.0 to qemu-2.2 -M pc-2.2
- or .... you get the idea

redhat.

One last thing

- We have done it wrong until now!
- What inter-version migration should we be allowing
- current: qemu-old -M pc to qemu-new -M pc
- What is pc?
- Really what we are doing is the equivalent of
- qemu-2.1 -M pc-2.1 to qemu-2.2 -M pc-2.2
- or qemu-2.0 -M pc-2.0 to qemu-2.2 -M pc-2.2
- or .... you get the idea

redhat.

One last thing

- We have done it wrong until now!
- What inter-version migration should we be allowing
- current: qemu-old -M pc to qemu-new -M pc
- What is pc?
- Really what we are doing is the equivalent of
- qemu-2.1 -M pc-2.1 to qemu-2.2 -M pc-2.2
- or qemu-2.0 -M pc-2.0 to qemu-2.2 -M pc-2.2
- or .... you get the idea

redhat.

One last thing

- We have done it wrong until now!
- What inter-version migration should we be allowing
- current: qemu-old -M pc to qemu-new -M pc
- What is pc?
- Really what we are doing is the equivalent of
- qemu-2.1 -M pc-2.1 to qemu-2.2 -M pc-2.2
- or qemu-2.0 -M pc-2.0 to qemu-2.2 -M pc-2.2
- or .... you get the idea

redhat.

One last thing

- We have done it wrong until now!
- What inter-version migration should we be allowing
- current: qemu-old -M pc to qemu-new -M pc
- What is pc?
- Really what we are doing is the equivalent of
- qemu-2.1 -M pc-2.1 to qemu-2.2 -M pc-2.2
- or qemu-2.0 -M pc-2.0 to qemu-2.2 -M pc-2.2
- or .... you get the idea

redhat.

One last thing

- We have done it wrong until now!
- What inter-version migration should we be allowing
- current: qemu-old -M pc to qemu-new -M pc
- What is pc?
- Really what we are doing is the equivalent of
- qemu-2.1 -M pc-2.1 to qemu-2.2 -M pc-2.2
- or qemu-2.0 -M pc-2.0 to qemu-2.2 -M pc-2.2
- or .... you get the idea

redhat.

Why not limit the problem?

- Machine type should be the same for migration to work
- qemu-2.1 -M pc-2.1 to qemu-2.2 -M pc-2.1
- qemu-2.0 -M pc-2.0 to qemu-2.2 -M pc-2.0
- This is testable, we only allow a limited number of interversion migrations
- This is basically what we do on Red Hat
- We can make a policy when some version compatiblity is dropped
- Worst case: ship old and new device

redhat.

Why not limit the problem?

- Machine type should be the same for migration to work
- qemu-2.1 -M pc-2.1 to qemu-2.2 -M pc-2.1
- qemu-2.0 -M pc-2.0 to qemu-2.2 -M pc-2.0
- This is testable, we only allow a limited number of interversion migrations
- This is basically what we do on Red Hat
- We can make a policy when some version compatiblity is dropped
- Worst case: ship old and new device

redhat.

Why not limit the problem?

- Machine type should be the same for migration to work
- qemu-2.1 -M pc-2.1 to qemu-2.2 -M pc-2.1
- qemu-2.0 -M pc-2.0 to qemu-2.2 -M pc-2.0
- This is testable, we only allow a limited number of interversion migrations
- This is basically what we do on Red Hat
- We can make a policy when some version compatiblity is dropped
- Worst case: ship old and new device

redhat.

Why not limit the problem?

- Machine type should be the same for migration to work
- qemu-2.1 -M pc-2.1 to qemu-2.2 -M pc-2.1
- qemu-2.0 -M pc-2.0 to qemu-2.2 -M pc-2.0
- This is testable, we only allow a limited number of interversion migrations
- This is basically what we do on Red Hat
- We can make a policy when some version compatiblity is dropped
- Worst case: ship old and new device

redhat.

Why not limit the problem?

- Machine type should be the same for migration to work
- qemu-2.1 -M pc-2.1 to qemu-2.2 -M pc-2.1
- qemu-2.0 -M pc-2.0 to qemu-2.2 -M pc-2.0
- This is testable, we only allow a limited number of interversion migrations
- This is basically what we do on Red Hat
- We can make a policy when some version compatiblity is dropped
- Worst case: ship old and new device

redhat.

Why not limit the problem?

- Machine type should be the same for migration to work
- qemu-2.1 -M pc-2.1 to qemu-2.2 -M pc-2.1
- qemu-2.0 -M pc-2.0 to qemu-2.2 -M pc-2.0
- This is testable, we only allow a limited number of interversion migrations
- This is basically what we do on Red Hat
- We can make a policy when some version compatiblity is dropped
- Worst case: ship old and new device

redhat.

Why not limit the problem?

- Machine type should be the same for migration to work
- qemu-2.1 -M pc-2.1 to qemu-2.2 -M pc-2.1
- qemu-2.0 -M pc-2.0 to qemu-2.2 -M pc-2.0
- This is testable, we only allow a limited number of interversion migrations
- This is basically what we do on Red Hat
- We can make a policy when some version compatiblity is dropped
- Worst case: ship old and new device

redhat.

Section 3

Anything else you need?

redhat.

Section 4

Questions

The end.

Thanks for listening.