Migration

How to hop from machine to machine without losing state

Red Hat
Juan Quintela

August 8, 2010

Abstract

This talk describes current migration status, and ideas for future work.

Contents

Section 1
Introduction

Types of migration

- savevm/loadvm
- migration
- live migration

Types of migration

- savevm/loadvm
- migration
- live migration

Types of migration

- savevm/loadvm
- migration
- live migration

Section 2

How to describe state State

Old way: simple device

```
static void adb_mouse_save(QEMUFile *f, void *opaque)
{
    MouseState *s = (MouseState *)opaque;
    qemu_put_sbe32s(f, &s->buttons_state);
    qemu_put_sbe32s(f, &s->last_buttons_state);
    qemu_put_sbe32s(f, &s->dx);
    qemu_put_sbe32s(f, &s->dy);
    qemu_put_sbe32s(f, &s->dz);
}
static int adb_mouse_load(QEMUFile *f, void *opaque, int version_id)
{
    MouseState *s = (MouseState *)opaque;
    if (version_id != 1)
        return -EINVAL;
    qemu_get_sbe32s(f, &s->buttons_state);
    qemu_get_sbe32s(f, &s->last_buttons_state);
    qemu_get_sbe32s(f, &s->dx);
    qemu_get_sbe32s(f, &s->dy);
    qemu_get_sbe32s(f, &s->dz);
    return 0;
}
```

New way: VMState

```
static const VMStateDescription vmstate_adb_mouse = {
    .name = "adb_mouse",
    .version_id = 1,
    .minimum_version_id = 1,
    .minimum_version_id_old = 1,
    .fields      = (VMStateField []) {
        VMSTATE_INT32(buttons_state, MouseState),
        VMSTATE_INT32(last_buttons_state, MouseState),
        VMSTATE_INT32(dx, MouseState),
        VMSTATE_INT32(dy, MouseState),
        VMSTATE_INT32(dz, MouseState),
        VMSTATE_END_OF_LIST()
    }
}
```

Arrays and code: old way

```
static void ads7846_save(QEMUFile *f, void *opaque)
{
    ADS7846State *s = (ADS7846State *) opaque;
    int i;
    for (i = 0; i < 8; i ++)
        qemu_put_be32(f, s->input[i]);
    qemu_put_be32(f, s->noise);
    qemu_put_be32(f, s->cycle);
    qemu_put_be32(f, s->output);
}
static int ads7846_load(QEMUFile *f, void *opaque, int version_id)
{
    ADS7846State *s = (ADS7846State *) opaque;
    int i;
    for (i = 0; i < 8; i ++)
        s->input[i] = qemu_get_be32(f);
    s->noise = qemu_get_be32(f);
    s->cycle = qemu_get_be32(f);
    s->output = qemu_get_be32(f);
    s->pressure = 0;
    ads7846_int_update(s);
    return 0;
}
```

Arrays and code: now VMState

```
static int ads7846_post_load(void *opaque, int version_id)
{
    ADS7846State *s = opaque;
    s->pressure = 0;
    ads7846_int_update(s);
    return 0;
}
static const VMStateDescription vmstate_ads7846 = {
    .name = "ads7846",
    .version_id = 0,
    .minimum_version_id = 0,
    .minimum_version_id_old = 0,
    .post_load = ads7846_post_load,
    .fields      = (VMStateField []) {
        VMSTATE_INT32_ARRAY(buttons_state, ADS7846State, 8),
        VMSTATE_INT32(noise, ADS7846State),
        VMSTATE_INT32(cycle, ADS7846State),
        VMSTATE_INT32(output, ADS7846State),
        VMSTATE_END_OF_LIST()
    }
}
```

Versions

```
...
if (version_id >= 10) {
    n->alluni = qemu_get_byte(f);
    n->nomulti = qemu_get_byte(f);
    n->nouni = qemu_get_byte(f);
    n->nobcast = qemu_get_byte(f);
}
...
```

Versions, now on VMState

```
...
VMSTATE_UINT8_V(alluni, VirtIONet, 10),
VMSTATE_UINT8_V(nomulti, VirtIONet, 10),
VMSTATE_UINT8_V(nouni, VirtIONet, 10),
VMSTATE_UINT8_V(nobcast, VirtIONet, 10),
...
```

.. and tests

```
static bool version_is_5(void *opaque, int version_id)
{
    return version_id == 5;
}
        ...
        VMSTATE_UINT32_TEST(halted, CPUState, version_is_5),
        ...
```

More state for a device

- Increase version
- problem with stable branches
- state is a hierarchy

More state for a device

- Increase version
- problem with stable branches
- state is a hierarchy

More state for a device

- Increase version
- problem with stable branches
- state is a hierarchy

Subsections

- Some state is optional
- newer versions always understand old versions
- allow *some* migration to older versions

Subsections

- Some state is optional
- newer versions always understand old versions
- allow *some* migration to older versions

Subsections

- Some state is optional
- newer versions always understand old versions
- allow *some* migration to older versions

Subsections (II)

```
static bool ide_drive_pio_state_needed(void *opaque)
{
    IDEState *s = opaque;
    return (s->status & DRQ_STAT) != 0;
}
const VMStateDescription vmstate_ide_drive_pio_state = {
    .name = "ide_drive/pio_state",
...
}
const VMStateDescription vmstate_ide_drive = {
    .name = "ide_drive",
...
    .subsections = (VMStateSubsection []) {
        {
            .vmsd = &vmstate_ide_drive_pio_state,
            .needed = ide_drive_pio_state_needed,
        }, {
            /* empty */
        }
    }
```

More VMState

- arrays of variable length
- arrays of pointers
- structs
- arrays of structs
- ...

More VMState

- arrays of variable length
- arrays of pointers
- structs
- arrays of structs
- . . .

More VMState

- arrays of variable length
- arrays of pointers
- structs
- arrays of structs
- ...

More VMState

- arrays of variable length
- arrays of pointers
- structs
- arrays of structs
- ...

More VMState

- arrays of variable length
- arrays of pointers
- structs
- arrays of structs
- . . .

RAM

- it's BIG, a.k.a. it is going to take time
- live migration makes things more complicated
- it's BIG
- layout changes with hotplug

RAM

- it's BIG, a.k.a. it is going to take time
- live migration makes things more complicated
- it's BIG
- layout changes with hotplug

RAM

- it's BIG, a.k.a. it is going to take time
- live migration makes things more complicated
- it's BIG
- layout changes with hotplug

RAM

- it's BIG, a.k.a. it is going to take time
- live migration makes things more complicated
- it's BIG
- layout changes with hotplug

Block devices

- they are backed by files
- files are external to QEMU
- qcow2
- NFS

Block devices

- they are backed by files
- files are external to QEMU
- qcow2
- NFS

Block devices

- they are backed by files
- files are external to QEMU
- qcow2
- NFS

Block devices

- they are backed by files
- files are external to QEMU
- qcow2
- NFS

Section 3

Future Work

End VMState conversion

- virtio: patches exist, have to rebase and sent.
- slirp: difficult.
- rest of cpus: work and testing.
- other 73 devices (not pc ones).

End VMState conversion

- virtio: patches exist, have to rebase and sent.
- slirp: difficult.
- rest of cpus: work and testing.
- other 73 devices (not pc ones).

End VMState conversion

- virtio: patches exist, have to rebase and sent.
- slirp: difficult.
- rest of cpus: work and testing.
- other 73 devices (not pc ones).

End VMState conversion

- virtio: patches exist, have to rebase and sent.
- slirp: difficult.
- rest of cpus: work and testing.
- other 73 devices (not pc ones).

VMState simplification

- removal of field version (use test)
- removal of `load_state_old`
- removal of pre 0.12 state?
- arrays can be handled better inside types

VMState simplification

- removal of field version (use test)
- removal of `load_state_old`
- removal of pre 0.12 state?
- arrays can be handled better inside types

VMState simplification

- removal of field version (use test)
- removal of `load_state_old`
- removal of pre 0.12 state?
- arrays can be handled better inside types

VMState simplification

- removal of field version (use test)
- removal of `load_state_old`
- removal of pre 0.12 state?
- arrays can be handled better inside types

Migration Format

- add size field?
- add checksum field?
- self descriptive?

Migration Format

- add size field?
- add checksum field?
- self descriptive?

Migration Format

- add size field?
- add checksum field?
- self descriptive?

main QEMU state

- current: running/stopped
- outgoing?: migration finished,
- incoming?: we are expecting migration

main QEMU state

- current: running/stopped
- outgoing?: migration finished,
- incoming?: we are expecting migration

main QEMU state

- current: running/stopped
- outgoing?: migration finished,
- incoming?: we are expecting migration

Incoming migration

- create a command
- create machine from description

Incoming migration

- create a command
- create machine from description

Users outside QEMU

- crash

The end.

Thanks for listening.