Machine-type Introspection and Configuration

Where Are We Going?

Eduardo Habkost
Red Hat, Inc.

KVM Forum 2016

Agenda

Concepts

Current Needs

Possible Solutions

Previous Work

Some overlap with KVM Forum 2015 talks:

- *Rethinking machine types* – David Gibson
- *QEMU interface introspection: from hacks to solutions* – Markus Armbruster

Slightly different problems are addressed here.

Concepts

Machine Types

- `-machine` command-line option

Machine Types

- `-machine` command-line option
- Goals:
  - Emulation of different machines by the same QEMU binary
  - Backwards compatibility (command-line, migration, guest ABI)

Machine Types

- `-machine` command-line option
- Goals:
  - Emulation of different machines by the same QEMU binary
  - Backwards compatibility (command-line, migration, guest ABI)
- Encapsulates:
  - Construction of basic/default devices
  - Migration stream format options
  - Other default options

2005: QEMUMachine

```
struct QEMUMachine {
    const char *name;
    const char *desc;
    QEMUMachineInitFunc *init;
    struct QEMUMachine *next;
};
```

2005: 5 machines

2016: MachineClass

```
struct MachineClass {
    ObjectClass parent_class;
    const char *family;
    const char *name;
    const char *alias;
    const char *desc;
    BlockInterfaceType block_default_type;
    int units_per_default_bus;
    int max_cpus;
    unsigned int no_serial:1, no_parallel:1, use_virtcon:1, use_sclp:1, no_floppy:1,
        no_cdrom:1, no_sdcard:1, has_dynamic_sysbus:1, pci_allow_0_address:1, legacy_fw_cfg_order:1;
    int is_default;
    const char *default_machine_opts;
    const char *default_boot_order;
    const char *default_display;
    GArray *compat_props;
    const char *hw_version;
    ram_addr_t default_ram_size;
    bool option_rom_has_mr;
    bool rom_file_has_mr;
    void (*init)(MachineState *state);
    void (*reset)(void);
    void (*hot_add_cpu)(const int64_t id, Error **errp);
    int (*kvm_type)(const char *arg);
    HotplugHandler *(*get_hotplug_handler)(MachineState *machine, DeviceState *dev);
    unsigned (*cpu_index_to_socket_id)(unsigned cpu_index);
    CPUArchIdList *(*possible_cpu_arch_ids)(MachineState *machine);
    HotpluggableCPUList *(*query_hotpluggable_cpus)(MachineState *machine);
};
```

2016: **120+** machines

Where differences are encoded

Most machine type differences are encoded at:

- `MachineClass::init()` function
  (e.g. `ppc_spapr_init()`, `pc_q35_init()`, `pc_init1()`, ...)
- `MachineClass` fields
  (e.g. `max_cpus`, `default_machine_opts`, `compat_props`, ...)

Current Needs

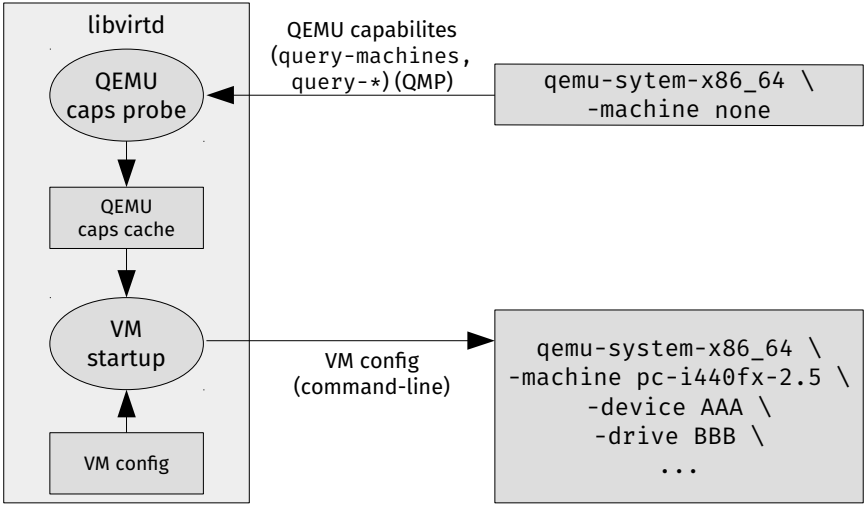
Current Needs: libvirt

```
[libvirt]$ git grep '\<STR.*machine' src/qemu
src/qemu/qemu_capabilities.c:            STREQ(def->os.machine, "ppce500"))
src/qemu/qemu_capabilities.c:            STREQ(def->os.machine, "prep"))
src/qemu/qemu_capabilities.c:            STREQ(def->os.machine, "bamboo"))
src/qemu/qemu_capabilities.c:        if (STREQ(def->os.machine, "mpc8544ds"))
src/qemu/qemu_capabilities.c:        if (STREQ(machines[i]->name, "none"))
src/qemu/qemu_capabilities.c:        STREQ(def->os.machine, "isapc");
src/qemu/qemu_capabilities.c:        (STRNEQ(machine, "pseries") && !STRPREFIX(machine, "pseries-")))
src/qemu/qemu_capabilities.c:        (STRNEQ(machine, "pseries") && !STRPREFIX(machine, "pseries-")))
src/qemu/qemu_capabilities.c:    if (STRNEQ(domCaps->machine, "virt") &&
src/qemu/qemu_capabilities.c:        !STRPREFIX(domCaps->machine, "virt-"))
src/qemu/qemu_command.c:    if (STRPREFIX(def->os.machine, "s390-virtio") &&
src/qemu/qemu_domain.c:        if (STREQ(def->os.machine, "isapc")) {
src/qemu/qemu_domain.c:        if (STREQ(def->os.machine, "versatilepb"))
src/qemu/qemu_domain.c:    return (STRPREFIX(def->os.machine, "pc-q35") ||
src/qemu/qemu_domain.c:            STREQ(def->os.machine, "q35"));
src/qemu/qemu_domain.c:    return (STREQ(def->os.machine, "pc") ||
src/qemu/qemu_domain.c:            STRPREFIX(def->os.machine, "pc-0.") ||
src/qemu/qemu_domain.c:            STRPREFIX(def->os.machine, "pc-1.") ||
src/qemu/qemu_domain.c:            STRPREFIX(def->os.machine, "pc-i440") ||
src/qemu/qemu_domain.c:            STRPREFIX(def->os.machine, "rhel"));
src/qemu/qemu_domain.c:    char *p = STRSKIP(def->os.machine, "pc-q35-");
src/qemu/qemu_domain.c:    return STRPREFIX(def->os.machine, "s390-ccw");
src/qemu/qemu_domain.c:    if (STRNEQ(def->os.machine, "virt") &&
src/qemu/qemu_domain.c:        !STRPREFIX(def->os.machine, "virt-"))
src/qemu/qemu_domain.c:    if (STRNEQ(def->os.machine, "pseries") &&
src/qemu/qemu_domain.c:        !STRPREFIX(def->os.machine, "pseries-"))
src/qemu/qemu_domain.c:        STREQ(def->os.machine, "malta") ||
src/qemu/qemu_domain.c:        STREQ(def->os.machine, "sun4u") ||
src/qemu/qemu_domain.c:        STREQ(def->os.machine, "g3beige");
src/qemu/qemu_domain_address.c:    if (!(STRPREFIX(def->os.machine, "vexpress-") ||
src/qemu/qemu_domain_address.c:    if (STREQ(def->os.machine, "versatilepb"))
```

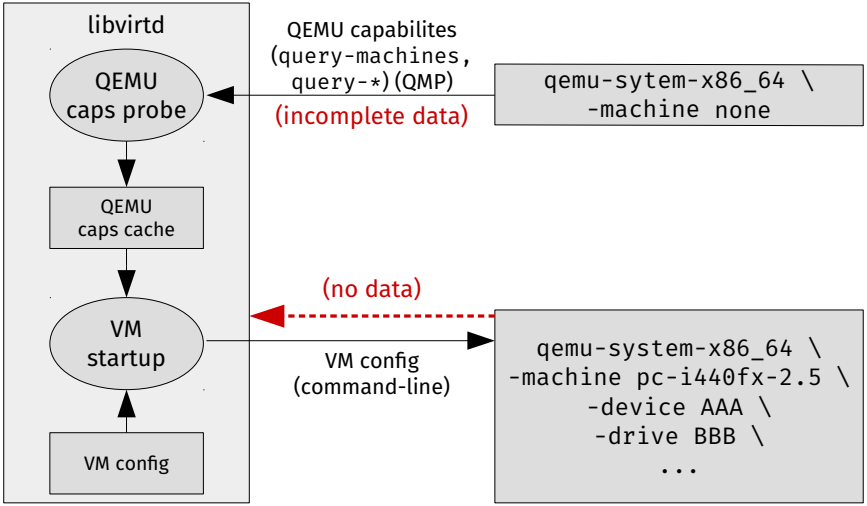
Current Needs: libvirt

```
bool qemuDomainMachineIsQ35();
bool qemuDomainMachineIsI440FX();
bool qemuDomainMachineNeedsFDC();
bool qemuDomainMachineIsS390CCW();
bool qemuDomainMachineIsVirt();
bool qemuDomainMachineIsPSeries();
bool qemuDomainMachineHasBuiltinIDE();
const char *qemuDomainDefaultNetModel();
bool virQEMUCapsHasPCIMultiBus();
bool virQEMUCapsSupportsVmport();
bool qemuDomainSupportsPCI();
```

Current Situation



Current Situation



query-machines QMP command

```
{ 'command': 'query-machines', 'returns': ['MachineInfo'] }

{ 'struct': 'MachineInfo',
  'data': { 'name': 'str', '*alias': 'str',
            '*is-default': 'bool', 'cpu-max': 'int',
            'hotpluggable-cpus': 'bool'} }
```

Extending `query-machines`

Extending `query-machines` is easy:

1. Move data to `MachineClass`
2. Add new fields to `MachineInfo`

Types of Queries

Some information depend on:

0. QEMU binary only (QEMU capabilities) — already available

Types of Queries

Some information depend on:

0. QEMU binary only (QEMU capabilities) — already available
1. Machine type name only — `query-machines`
    - Easy to extend

Types of Queries

Some information depend on:

0. QEMU binary only (QEMU capabilities) — already available
1. Machine type name only — `query-machines`
   - Easy to extend
   - But, how much can it grow?

Types of Queries

Some information depend on:

0. QEMU binary only (QEMU capabilities) — already available
1. Machine type name only — `query-machines`
   - Easy to extend
   - But, how much can it grow?
2. Machine type + machine options (including accelerator)
   - Can't be queried today

Types of Queries

Some information depend on:

0. QEMU binary only (QEMU capabilities) — already available
1. Machine type name only — `query-machines`
   - Easy to extend
   - But, how much can it grow?
2. Machine type + machine options (including accelerator)
   - Can't be queried today
3. Machine type + other QEMU options
   - Can't be queried today

Example 1: CPU slots

How to convert:

```
$ qemu-system-x86_64 -smp 4 -cpu Haswell \
  -numa node,cpus=0-1 -numa nodes,cpus=2-3
```

to:

```
$ qemu-system-x86_64 \
  -numa node -numa node \
  -device Haswell-x86_64-cpu,socket=0,node=0 \
  -device Haswell-x86_64-cpu,socket=1,node=0 \
  -device Haswell-x86_64-cpu,socket=2,node=1 \
  -device Haswell-x86_64-cpu,socket=3,node=1
```

Example 1: CPU slots

How to convert:

```
$ qemu-system-x86_64 -smp 4 -cpu Haswell \
  -numa node,cpus=0-1 -numa nodes,cpus=2-3
```

to:

```
$ qemu-system-x86_64 \
  -numa node -numa node \
  -device Haswell-x86_64-cpu,socket=0,node=0 \
  -device Haswell-x86_64-cpu,socket=1,node=0 \
  -device Haswell-x86_64-cpu,socket=2,node=1 \
  -device Haswell-x86_64-cpu,socket=3,node=1
```

- Set of available CPU slots is arch-dependent, machine-dependent
- See `query-hotpluggable-cpus`

Example 2: Address Assignment

- Device address assignment depend on:
  - The default devices for the machine type
  - Additional devices configured for the VM
- Current solution: `libvirt/src/qemu/qemu_domain_address.c`
  - A large collection of special cases (1.2k LOC)
  - Duplicates QEMU logic

Example 2: Address Assignment

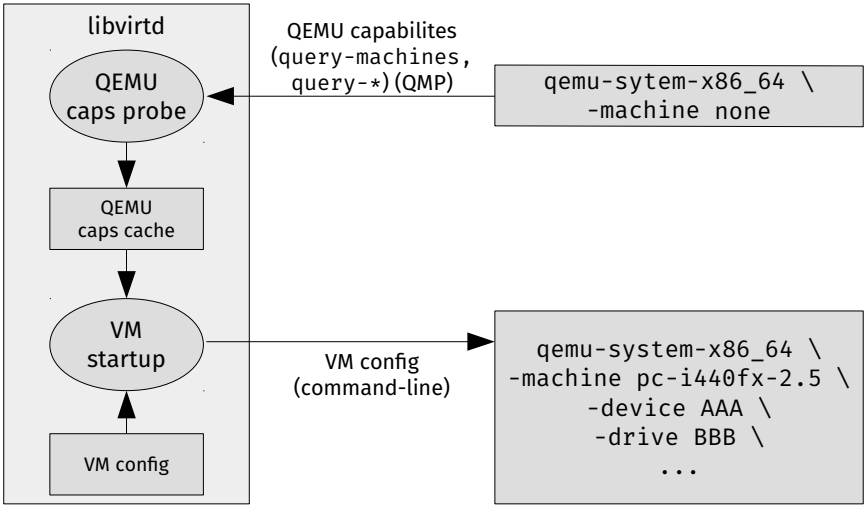
```
static int
qemuDomainValidateDevicePCISlotsChipsets(virDomainDefPtr def,
                                         virQEMUCapsPtr qemuCaps,
                                         virDomainPCIAddressSetPtr addrs)
{
    if (qemuDomainMachineIsI440FX(def) &&
        qemuDomainValidateDevicePCISlotsPIIX3(def, qemuCaps, addrs) < 0) {
        return -1;
    }

    if (qemuDomainMachineIsQ35(def) &&
        qemuDomainValidateDevicePCISlotsQ35(def, qemuCaps, addrs) < 0) {
        return -1;
    }

    return 0;
}
```

Example 2: Address Assignment

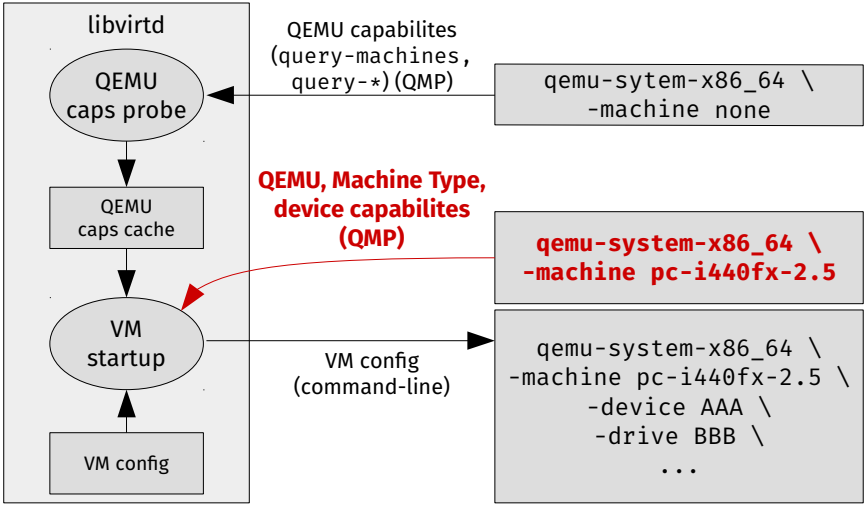
```
for (i = 0; i < def->nserials; i++) {
    if (def->serials[i]->deviceType == VIR_DOMAIN_CHR_DEVICE_TYPE_SERIAL &&
        qemuDomainMachineIsPSeries(def))
        def->serials[i]->info.type = VIR_DOMAIN_DEVICE_ADDRESS_TYPE_SPAPRVIO;
    if (qemuDomainAssignSpaprVIOAddress(def, &def->serials[i]->info,
                                        VIO_ADDR_SERIAL) < 0)
        goto cleanup;
}
```

Possible Solutions

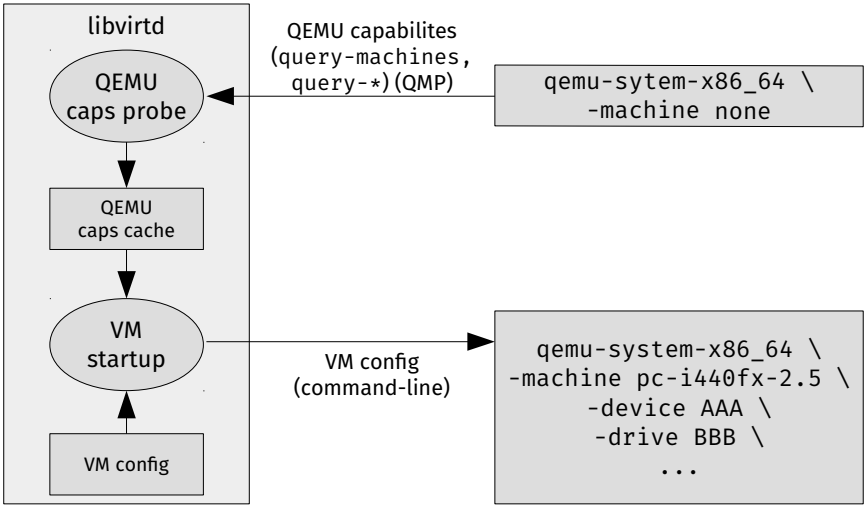
Solution 1


libvirtd
QEMU
caps probe
QEMU
caps cache
VM
startup
VM config
QEMU capabilites
(query-machines,
query-*) (QMP)
qemu-sytem-x86_64 \
-machine none
VM config
(command-line)
qemu-system-x86_64 \
-machine pc-i440fx-2.5 \
-device AAA \
-drive BBB \
...

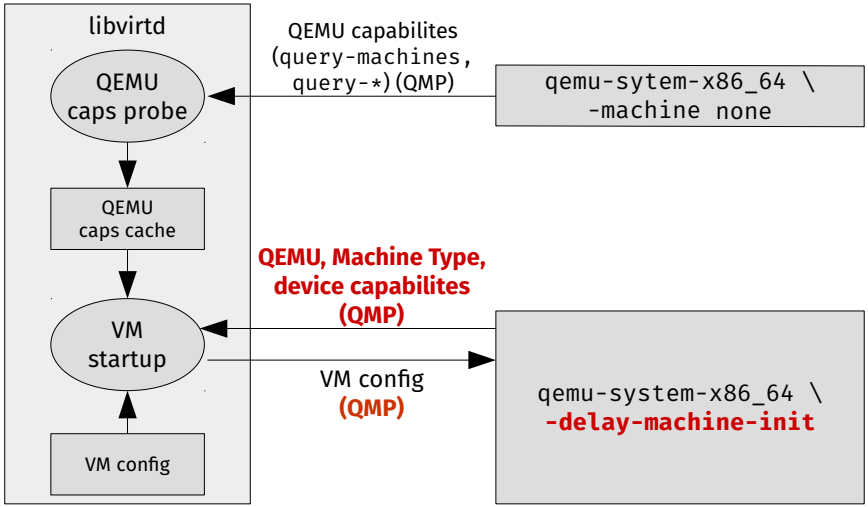
Solution 1


libvirtd
QEMU caps probe
QEMU caps cache
VM startup
VM config
QEMU capabilites (query-machines, query-*) (QMP)
qemu-sytem-x86_64 \
-machine none
QEMU, Machine Type, device capabilites (QMP)
qemu-system-x86_64 \
-machine pc-i440fx-2.5
VM config (command-line)
qemu-system-x86_64 \
-machine pc-i440fx-2.5 \
-device AAA \
-drive BBB \
...

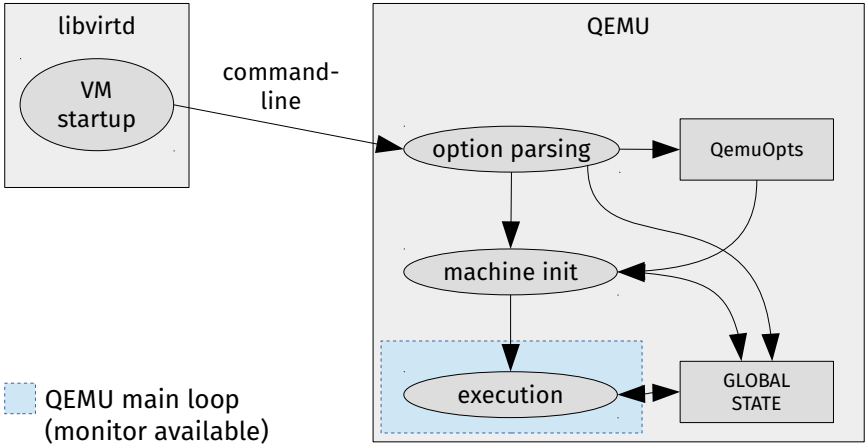
Solution 2


libvirtd
QEMU
caps probe
QEMU
caps cache
VM
startup
VM config
QEMU capabilites
(query-machines,
query-*) (QMP)
qemu-sytem-x86_64 \
-machine none
VM config
(command-line)
qemu-system-x86_64 \
-machine pc-i440fx-2.5 \
-device AAA \
-drive BBB \
...

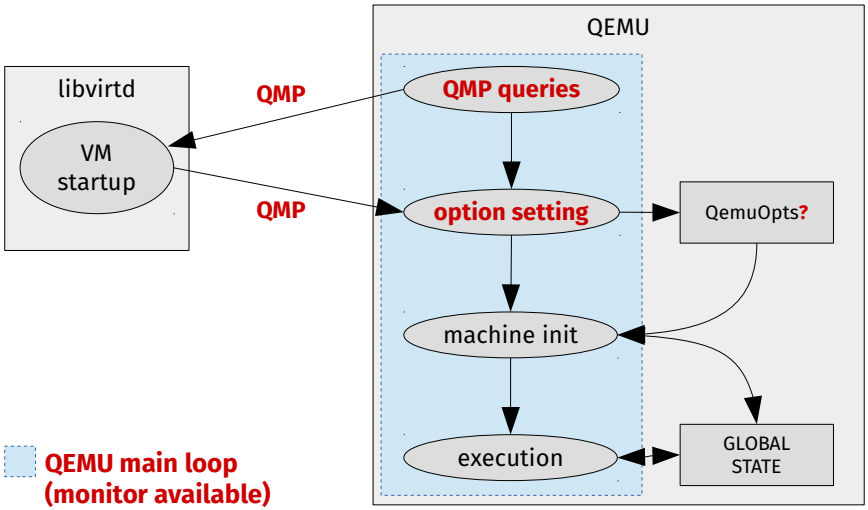
Solution 2


libvirtd
QEMU caps probe
QEMU caps cache
VM startup
VM config
QEMU capabilites (query-machines, query-*)(QMP)
qemu-sytem-x86_64 \ -machine none
QEMU, Machine Type, device capabilites (QMP)
VM config (QMP)
qemu-system-x86_64 \ -delay-machine-init

Solution 2: VM configuration



Solution 2: VM configuration



Tasks/Challenges

1. Enumerate what information is needed by libvirt

Tasks/Challenges

1. Enumerate what information is needed by libvirt
2. Extend `query-machines` where appropriate (easy)

Tasks/Challenges

1. Enumerate what information is needed by libvirt
2. Extend `query-machines` where appropriate (easy)
3. Design/implement new QMP queries (hard)

Tasks/Challenges

1. Enumerate what information is needed by libvirt
2. Extend `query-machines` where appropriate (easy)
3. Design/implement new QMP queries (hard)
4. `-delay-machine-init`:
   - Make QMP monitor available before machine init (doable?)
   - QMP commands to replace QEMU command-line options (doable?)

Open Questions

- Configuration commands: `QemuOpts`, QOM-based, QAPI-based?
- How to encode more complex rules. e.g.:
  - Address allocation
  - Machine+bus+device types compatibility (PCI, PCIe)
- Is there a limit on the complexity of `query-machines`?

Future Work

- Simplify libvirt QEMU capability probing (on-demand probing?)
- Machine-friendly error reporting from machine init
- Split/refactor machine initialization and execution steps further
- Machine types as data: exporting/loading complete machine types definitions (far future?)

The End

Thanks for listening.

Questions?

Answer the KVM Forum poll: **http://goo.gl/SCCpky**

Some History

How machine types evolved:

- 2005: `struct QEMUMachine` — 5 machines
- 2009: `QEMUMachine::compat_props` (*pc-0.10* is born)
- 2012: `query-machines` QMP command
- 2012: “none” machine type introduced
- 2013: `pc_init_pci_X_Y()`
- 2013: `cpu-max` field on `query-machines`
- 2014: *pseries-2.1* is born
- 2014: `TYPE_MACHINE` QOM class.
- 2015: `QEMUMachine` replaced by QOM `TYPE_MACHINE` subclasses (`MachineClass`) — 120+ machines
- 2015: Portions of `QEMUMachine::init` code moved to `MachineClass` fields
- 2016: `query-machines` return CPU hotplug information

Config Options

2009: `-readconfig` and `-writeconfig` options introduced.

Config Options

2009: `-readconfig` and `-writeconfig` options introduced.

From the commit message:

> *In theory you should be able to do:*
>
> ```
> qemu <machine config cmd line switches here> -writeconfig vm.c
> qemu -readconfig vm.cfg
> ```
>
> ***In practice it will not work**. Not all command line switches are converted to* `QemuOpts`*, so you'll have to keep the not-yet converted ones on the second line. Also there might be bugs lurking which prevent even the converted ones from working correctly.*

Initialization: Execution of QEMU binary

Done by management software (libvirt).

Input:

- VM configuration
- Capabilities of each QEMU binary

Output:

- QEMU binary path
- Command-line options and/or config file

Initialization: Option Parsing

Input:

- Command-line options
- Config file(s) (`-readconfig`)

Output:

- Static and global variables
- Local variables in `main()`
- `QemuOpts` sections

Initialization: Machine Type Lookup

Input:

- `QemuOpts`' `"machine"` section

Output:

- Local variable: `MachineClass *machine_class`

Initialization: Machine Initialization

`MachineClass::init()` function runs.

Input:

- Global variables
- Local variables
- `QemuOpts` sections

Output:

- Global and static variables
- QOM global properties
- `MachineState` fields
- Device tree
- `QemuOpts`?
- More?