KVM and CPU feature enablement

Eduardo Habkost <ehabkost@redhat.com>
Developer Conference 2014
Agenda

• Basic concepts
• Existing mechanisms and current challenges
• Current work and future plans
Basics
Introduction: Basics

Host CPU

Kernel

KVM

QEMU

libvirt

VM

Guest OS

(management)
Introduction: Basics

```
(management)

libvirt

QEMU

Kernel

KVM

Guest OS

VM

command-line, monitor (QMP)

Host CPU
```
Introduction: Stable guest ABI

• Guest OS should see the “same” machine, even if the host system has changed

• Hard requirement for live migration

• Soft requirement for non-live migration

• Host system may change a lot, but VM should look the same
x86 CPUID instruction

- Returns information about the running CPU
  - Most information shown on `/proc/cpuinfo`
- Feature flags indicating a feature is present
- Other more complex data
  - e.g.: cache and topology information
- **CPUID data is part of guest ABI**
Existing Mechanisms
CPUID handling

<table>
<thead>
<tr>
<th>QEMU</th>
<th>Guest OS</th>
</tr>
</thead>
</table>

Kernel

HOST CPU
CPUID handling

QEMU

GET_SUPPORTED_CPUID

Guest OS

Host
CPUID
instruction

Kernel

HOST CPU
CPUID handling

Diagram:

- QEMU
- Guest OS
- GET_SUPPORTED_CPUID
- SET_CPUID
- Host CPUID instruction
- CPUID table
- Host CPU

Kernel
CPUID handling

Kernel

QEMU

GET_SUPPORTED_CPUID

Host CPUID instruction

CPUID table

Guest OS

SET_CPUID

emulate CPUID

Get Supported CPUID

Guest CPUID instruction

HOST CPU
The Stack

**Decision** to enable a feature (should be) taken in the upper layers

Lower layers affect the **ability** to enable a feature
Enabling new features

• We can't silently enable or disable a feature:
  • It breaks guest ABI
  • May unexpectedly prevent migration to other (less powerful) hosts
CPU models

- **CPU model table**, different CPUID data on each entry
  - `qemu-system-x86_64 -cpu SandyBridge`
  - `qemu-system-x86_64 -cpu Haswell`

- Controlling individual features. e.g.: `-cpu Nehalem,+aes`

- CPU model entries may change, **machine-types** keep compatibility
  - `qemu-system-x86_64 -machine pc-1.6 -cpu SandyBridge`
  - `qemu-system-x86_64 -machine pc-1.7 -cpu SandyBridge`

- **enforce** flag. e.g.: `-cpu SandyBridge, enforce`

- **Required** to ensure predictable results
CPU models

• Special CPU model: \texttt{--cpu host}

  • Will enable everything that’s supported by the host

  • No stable guest ABI
Management requirements

• Ensuring that the resulting CPUID data is what was asked for

• Knowing which CPU models can be enabled in a host

• Knowing which features are available in a host

• Knowing to which hosts a VM can be migrated
Issues
Problem: querying CPU model information

• libvirt has its own list of CPU models

• libvirt doesn't know QEMU CPU models can change over time

• QEMU's fault, there's no good API for that (yet!)
Problem: no “enforce” mode

- libvirt doesn’t use the enforce flag
- Error reporting not machine-friendly
- Most serious issue so far
- Fix involves implementing CPU model and host capability APIs
Problem: querying host capabilities

- libvirt queries host CPU features directly using CPUID instruction
- Ignores KVM capabilities
- Ignores QEMU capabilities
- Ignores features that require extra CPU capabilities
- QEMU’s fault, there’s no good API for that (yet!)
Solutions
Solutions

• Existing interfaces: CPU-specific options and commands
  • -cpu, cpu-add, query-cpu-definitions

• New interfaces: based on common infrastructure (QDev, QOM)
QDev

- QDev = QEMU Device Model
- QOM = QEMU Object Model
- QDev devices are QOM objects
- `-device` command-line option
- QMP commands:
  - Adding devices/objects (`device_add`, `object-add`)
  - Removing devices/objects (`device_del`, `object-del`)
  - Getting/setting devices properties (`qom-get`, `qom-set`)
  - Listing objects and object classes (`qom-list`, `qom-list-types`)
QDev-based solution

- CPUs are QDev devices (done)
  - CPU devices and its properties visible through QMP
- feature-words property (done)
  - Query CPU model info
  - Query host capabilities ("host" CPU model)
  - Incomplete: no machine-type-specific data
- filtered-features property (done)
  - Used to emulate "enforce" mode with better error reporting
- Not used by libvirt yet
What’s missing (1/2)

• Querying CPU model information without re-running QEMU
  • Solution: separate QOM types for each CPU model
• Exposing machine-type-specific data
  • No defined solution yet
• Use QOM properties to control all feature flags
• Changing libvirt to use the new stuff
What’s missing (2/2)

• Long term plans:

  • Deprecate `-cpu, cpu-add` and use only QDev commands
    (`-device, device_add`)

  • Better interfaces to specify CPU topology (NUMA nodes, sockets, cores, threads)
Future

- Reporting capabilities reliably → smarter management systems
  - Usability (automatically choosing good defaults)
  - Smarter VM scheduling
- May require extending libvirt API
Thanks

Feedback:

http://devconf.cz/f/34

Additional info / pointers:

http://wiki.qemu.org/Features/
CPUModels

ehabkost@redhat.com

Questions?