Arm CPU models in QEMU

Where we are, and where we might be going

Cornelia Huck <cohuck@redhat.com>
IRC: cohuck
Today’s talk

Where are we right now? (The Present)

What do we need? (The Future)

How can we achieve it?

Discuss :)
Where are we now?
TCG

- Named CPU models
  - cortex-<foo>, neoverse-n1, ...
  - “max” CPU model
Arm CPU models on QEMU: Where are we now?

KVM

- "host"/"max CPU model
- named CPU models not very useful
  - good luck finding a model that matches your hardware...
CPU properties

- CPU features
  - e.g. pauth
- tcg-specific properties
  - e.g. pauth-impdef
- KVM-specific properties
  - e.g. kvm-stealtime
What do we need?
Defining a CPU

• reproducible (same invocation gives the same CPU)
• future proof (new features can be easily added)
• working across different accelerators
• introspectible (knobs need to be discoverable)
Compatibility handling

- depends on reproducible CPU definitions
- possible tie-in with compat machines
- limit to reasonably similar CPUs
- needs kernel support for KVM
  - e.g. for limiting features
MIDR and other fun

- we need to handle errata
- and subtle differences in behaviour for CPUs in different boards
- To what level should tcg emulate this?
How can we achieve it?
Named CPU models

• Option 1: architecture versions (e.g. Arm v8.6)
  • Problem: hard to figure out from a given CPU
  • Problem: lots of optional features, so still very heterogeneous

• Option 2: named models, as today
  • Problem: many CPUs exist, and more will exist in the future
  • Problem: CPUs may come in slightly different variations, especially in different boards

• Option 3: stick with Frankenmonster CPUs
  • i.e. build a CPU model, assign a uuid, and make it reproducible
CPU properties

- if it may differ between systems, we need a way to tweak it
- if it is visible in `/proc/cpuinfo` in Linux, we need a way to tweak it
- if it is a CPU specific implementation defined feature, we need a way to tweak it
- ...and all of this makes it grow into a large zoo of properties...
Arm CPU models on QEMU: How can we achieve it?

Accelerator support

- with tcg, CPU models control what is being emulated
  - restrictions mostly come from the board
- with KVM, CPU models need to be based on what is supported by the host
  - restrictions come from the board, the host hardware, and the actual version of the accelerator
Configuring it

- the user needs to get “reasonable” defaults when they don’t care about details
- management software like libvirt needs to be able to introspect options
Open questions

• How much flexibility?
  • Complete roll-your-own vs some constraints (e.g. no v8.9 features on a basically v8.0 CPU)
• How much compatibility?
  • Same vCPU on wildly different hosts vs small tweaks on basically the same host system
• Do we need to expose every erratum and implementation detail, or can we limit ourselves to a subset?
Let’s discuss :)
Thank you

Red Hat is the world’s leading provider of enterprise open source software solutions. Award-winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500.