SFO17-106
The Purpose of QEMU Emulation
Christoffer Dall
What is QEMU

● System Emulation
  ○ Emulates a specific hardware platform and run a foreign kernel and applications.
  ○ Example: Emulate a RaspBerry Pi 2 using an x86 server
  ○ Example: Android Emulator

● User-mode Emulation
  ○ Runs foreign binaries on existing kernel
  ○ Example: Run AArch64 GCC on x86 development laptop

● Driver for KVM
  ○ KVM is a kernel feature which lets user space run native code in a sandbox (VM) directly on the hardware
  ○ QEMU uses KVM to configure and run VMs
  ○ Provides I/O, device emulation, and other features such as migration for KVM
QEMU Emulation

- System Emulation
- User-mode Emulation
Why QEMU Emulation?

● Running CI
  ○ Easier and cheaper alternative than running board farm

● Debug environment
  ○ Single-stepping kernels without JTAG
  ○ Introspection
  ○ Tracing
  ○ Record/replay, reverse execution, system snapshotting, ...

● Cross-compilation
  ○ Targets may be slow or memory-constrained making native compiles impossible
  ○ Building and running applications directly on development machine significantly improves productivity

● System Modeling
  ○ Easy customizability and quick prototyping
Currently out of scope

- **Cycle-accurate simulation**
  - Other better alternatives, like Gem5
- **Proven Architecture Compliant Modeling**
  - There is no practical way for us to do this
- **Make QEMU more modular**
  - Allows QEMU to be used as a flexible device modeling tool
  - Potentially better alternatives like SystemC
  - GPL challenges
  - Member interest unknown?
- **Model all possible ARM CPUs and boards**
  - Way too much work (requires team of 20+ people)
  - Choose representative and popular chips
- **Provide better-than-native performance**
  - There is powerful ARM hardware available these days
Potential Future QEMU Work for Linaro

- New architecture features (ARMv8.1, ARMv8.2, …)
  - Prevents being stuck in the stone-age when QEMU support for new architecture features are requested by members which rely on existing support for new architecture features
  - Essential for CI loops of software that supports newer architecture features
  - Important for GCC testing of new instructions added in revisions of the architecture
  - Useful for kernel developers
QEMU Missing Architecture Support

- Currently unsupported ARMv8.1 features:
  - 'far atomics'
  - SIMD rounding double multiply add/sub
  - Page Table Hierarchical Permission Disables
  - Hardware Management of access flag/dirty bit
  - Privileged Access Never
  - Limited ordering regions [utility unclear]
  - 16 bit VMIDs
  - Virtualization Host Extensions (VHE)

- Currently unsupported ARMv8.2 features:
  - 52-bit physical addresses
  - DC CVAP [clean to point-of-persistency, for non-volatile memory]
  - PAN-aware AT instructions
  - UserAccessOverride PSTATE bit
  - 2-bit eXecute Never fields
  - CnP (Common not Private) [for sharing TLB entries in multithreading]
  - RAS extension [at least the minimum-required-level]
QEMU Missing Architecture Support

- Currently unsupported ARMv8.3 features:
  - Pointer Authentication
  - Nested Virtualization
  - Javascript fp-convert instruction
  - RCpc memory-consistency load/store insns
Potential Future QEMU Work for Linaro

- Better Guest Introspection
  - Improve Tracing Infrastructure
  - Potentially allow users to load dynamic modules to control and parse tracing events
  - Improve record/replay (mostly undocumented, unused, and untested)
  - Deterministic Execution
  - Thread Sanitizing tool to detect race conditions
Potential Future QEMU Work for Linaro

• Better user-mode emulation support
  ○ Linaro has significantly improved user-mode emulation support, but there are still many things left to do
  ○ Support complex large multithreaded applications (e.g. node.js)
  ○ Refactoring to make it easier to maintain and more integrated with the rest of the code
  ○ Support VDSO
  ○ Potentially support record/replay for user-mode emulation
Potential Future QEMU Work for Linaro

● Core QEMU Project Improvements
  ○ (Not necessarily ARM specific)
  ○ Documentation Infrastructure Improvements
  ○ Documentation improvements
  ○ Support writing devices in something else than C
    ■ Using a higher-level language reduces security bugs and makes prototyping and system integration much easier.
    ■ Using something like Rust would allow this without rewriting all of QEMU and has some potential
  ○ Modularization
    ■ QEMU is a big project
    ■ Allowing partial compile etc.
  ○ Cleanup
    ■ Getting rid of broken and unmaintained code
    ■ Update outdated documentation
Questions to members

- What is your company’s primary use/interest of QEMU Emulation?
  - ☐ System modeling
  - ☐ Software prototyping
  - ☐ Kernel development
  - ☐ CI
  - ☐ Cross-compilation
  - ☐ ____________
Questions to members

- How Important is this to your company?
  - [ ] Business Critical
  - [ ] Essential tool for efficiency
  - [ ] Nice-to-have
Questions to members

- Which teams in your company uses QEMU system-mode or user-mode emulation?
  - R&D
  - Software Engineering
  - Hardware Engineering
  - Quality Assurance (QA)
  - ____________
Other feedback?

Any other feedback or wishes in terms of priority or desired efforts for QEMU emulation?
Thank You

#SFO17

BUD17 keynotes and videos on: connect.linaro.org
For further information: www.linaro.org