Using Upstream QEMU for Computer Architecture and Software Studies

Christopher Covington

August 19th, 2015

Agenda

- Background, definitions
- Prior work
- Selected use cases
  - Instruction count based fast forwarding
  - Control flow tracing
  - Memory access tracing
  - Checkpoint interoperability
- Discussion

Background

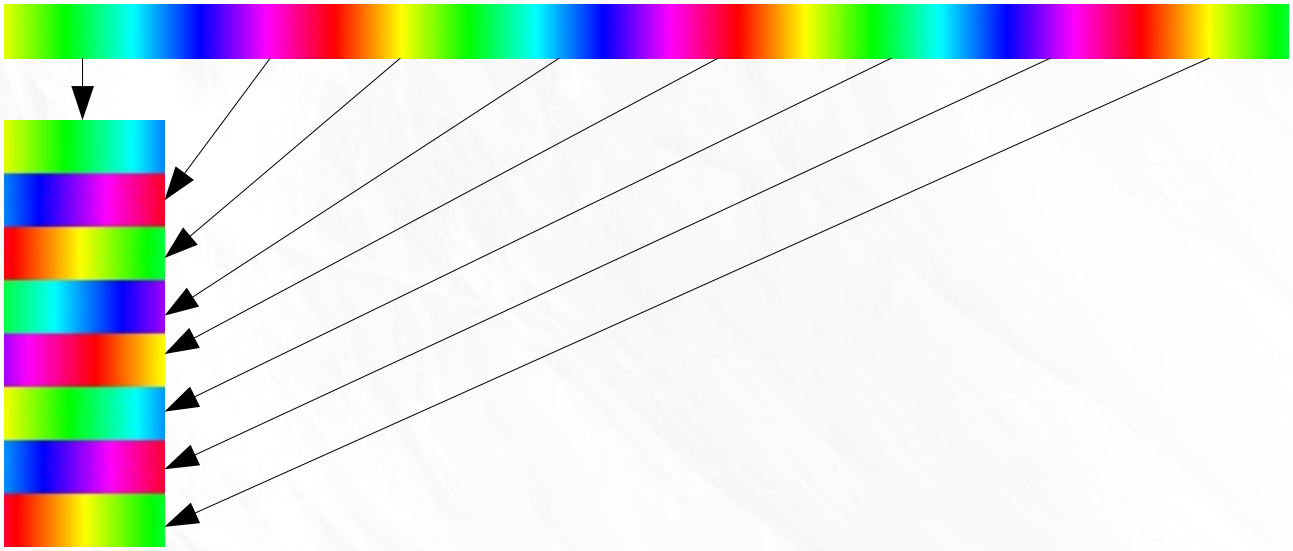
- Most experienced with AArch64
- System types
  - “Functional” models
  - “Timing” models
  - Hardware description language simulators
- CRIU: Checkpoint/Restore In Userspace
- Fast forwarding: dumping a checkpoint on a fast system and restoring it on a slow system

Prior Work

- Thank you QEMU developers!
- Computer Architecture
  - QSim http://manifold.gatech.edu/projects/qsim-a-multicore-emulator/
  - MARSSx86 http://marss86.org/~marss86/index.php/Home
  - QEMU Trace http://web.eece.maine.edu/~vweaver/projects/qemu-trace
  - VCSIMx86 http://www3.cs.stonybrook.edu/~hkang/software/vcsimx86.html
- Software Analysis
  - QEMU BBV http://web.eece.maine.edu/~vweaver/projects/qemusim/
  - PANDA https://github.com/moyix/panda
  - S2E http://s2e.epfl.ch/

Fast Forwarding

Fast Forwarding



Fast Forwarding: Assumptions

- Determinism: Starting from the same initial state and running for the same duration faithfully recreates subsequent state
- Checkpointing: Checkpoints faithfully recreate initial state

Fast Forwarding: Using QEMU, Linux, and CRIU

Architecturally executed instructions used as basic unit of measurement.

On QEMU (functional model):

```
ptrace-wait $pid $(($isize * $inum))
criu dump -j -t $pid
```

On timing model:

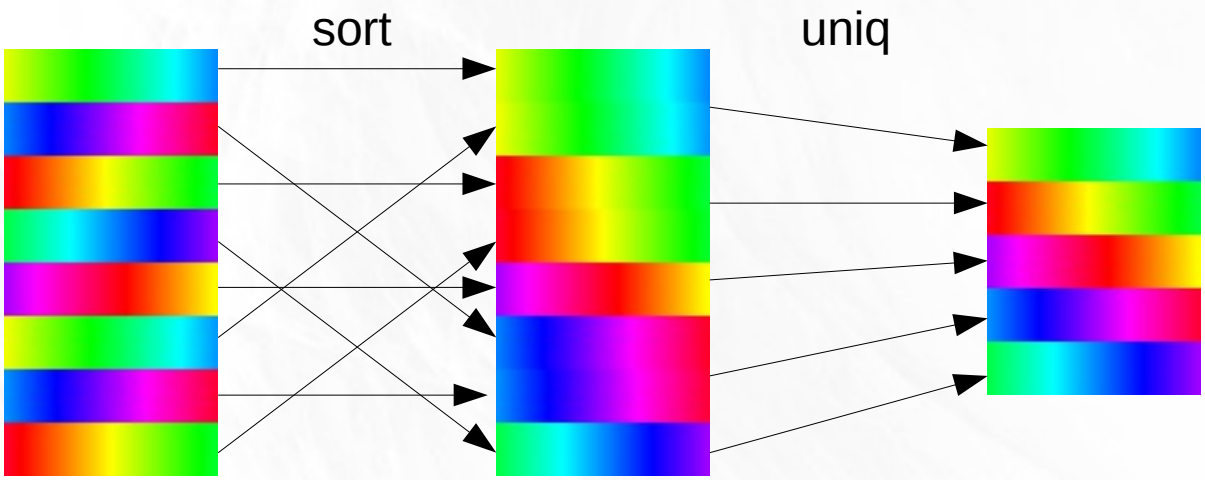
```
criu restore
perf stat -t $pid
ptrace-wait $pid $isize
```

Fast Forwarding: Functionality Required of QEMU

- `ptrace-wait` uses `perf_event_open` and the instructions event, which on AArch64 uses ARM PMUv3 hardware.
- ARM PMUv3 support for counting instructions and sending interrupts on overflow is missing.

- Superior alternatives?
- Any parts already implemented in QEMU, such as on other architectures?
- Useful for other purposes?

Fast Forwarding: Sampling to Avoid Redundant Work



SMARTS statistical sampling

http://users.ece.cmu.edu/~jhoe/doku/doku.php?id=smarts_simulation_sampling

SimPoint k-means clustering

https://www.cs.ucsb.edu/~sherwood/pubs/IEEEMicro-phases.pdf

Control Flow Tracing

- Useful for high-level characterization of fixed-work applications
- Can record
  - Number, variety, and duration (in instructions)
  - Of system calls, library calls, function calls, and loops
- Basic Block Vectors (BBVs) used by SimPoint are essentially histograms of control flow

Control Flow Tracing: Functionality Required of QEMU

- `-d exec` option gets most of the way there
- In addition, need to know
  - Length of each block
  - If a block is only partially executed, how much of it is executed/abandoned
  - If a block is linked circularly, how many iterations are executed

Control Flow Tracing:
Functionality Required of QEMU

- icount has most of this information
- Exposing information to target/guest would be nice too, probably in the form of an emulated Embedded Trace Macrocell (ETM) device

- Superior alternatives?
- Any parts already implemented in QEMU, such as on other architectures?
- Useful for other purposes?

Memory Access Tracing

- More application detail
  - To drive simulators of memory hierarchy components, such as caches
  - To create self-restoring checkpoints (“Intrinsic Checkpoints with Binary Modification”) http://deepblue.lib.umich.edu/handle/2027.42/60726

Memory Access Tracing: Functionality Required of QEMU

- Need to record
  - Read or write operation
  - Guest/target virtual address
  - Guest/target physical address
  - Value being read or written

Memory Access Tracing: Functionality Required of QEMU

- Not sure what related facilities QEMU already has
- Exposing information to target/guest would be nice too, probably in the form of an emulated Embedded Trace Macrocell (ETM) device

- Superior alternatives?
- Any parts already implemented in QEMU, such as on other architectures?
- Useful for other purposes?

Checkpoint Interoperability

- Speculative, but what if QEMU linux-user mode, CRIU, and self-restoring checkpoints could interoperate?

- Superior alternatives?
- Any parts already implemented in QEMU, such as on other architectures?
- Useful for other purposes?

Thank You

Christopher Covington is an employee of the Qualcomm Innovation Center, Inc. The Qualcomm Innovation Center, Inc. is a member of the Code Aurora Forum, a Linux Foundation Collaborative Project.