Dyninst arm64 port status

Itaru Kitayama
RIKEN Advanced Institute for Computer Science
Agenda

● The Dyninst toolkits
  ○ Quick overview
  ○ How it works
  ○ Comparison with uprobes
  ○ Build system
● What Works and What Not on arm64
● Dyninst arm64 Porting Work
● Testsuite Package
● Outlook
The Dyninst Toolkits - Quick overview

● APIs for static and dynamic instrumentation and modification
  ○ Based upon binary code analysis
● Platform-independent (Linux, FreeBSD, and Windows), multi-arch support (x86, PowerPC, and arm64)
  ○ AArch64 mode support is Community’s current focus
● Community-driven development, fully open-sourced under LGPL
  ○ https://github.com/dyninst/dyninst
● Used in many debugging and performance tools
  ○ Systemtap (when invoked with the --dyninst option)
  ○ TAU
  ○ Vampir
  ○ Extrae
The Dyninst Toolkits - How it works

1. Mutator process accesses mutatee’s memory and events via ptrace
2. Generate code
3. Uses “trampolines” to transfer further to the generated code
   a. Replace original instructions with a branch to base trampoline
   b. Further branched to mini trampolines that have generated code
   c. Upon return base trampoline executes original instructions
The Dyninst Toolkits - Comparison with uprobes

uprobes (ported to arm64)

- Insn at target address is copied aside and the BRK insn is called, the pre- and post-handlers are executed at EL1
- Certain instructions aren’t trappable
- Dealing with atomic sequences is best effort

Dyninst

- Direct instrumentation. No context switching, stays at EL0
- Dealing with atomic sequence is hard as well
The Dyninst Toolkits - Build system

Dyninst uses CMake for platform portability and its rich features

- Mostly built natively
- Has Cotire (compile time reducer) to speed up the builds
- Fedora 25 is a good release
- Mustang serves as a build machine

Full build time of Dyninst release v9.3.x, gcc version 6.3.1

<table>
<thead>
<tr>
<th></th>
<th>Intel(R) Xeon(R) CPU E3-1285 v3 @ 3.60GHz memory 16GB</th>
<th>APM Potenza CPU Rev B0 @ 2.40GHz memory 32GB</th>
</tr>
</thead>
<tbody>
<tr>
<td>real</td>
<td>5m1.993s</td>
<td>real 15m31.589s</td>
</tr>
<tr>
<td>user</td>
<td>23m0.040s</td>
<td>user 57m3.565s</td>
</tr>
<tr>
<td>sys</td>
<td>0m51.792s</td>
<td>sys 0m54.003s</td>
</tr>
</tbody>
</table>
What Works, What Not on arm64

WORKS
Using Fedora 25 (latest kernel is v4.9), we’ve verified:

- Process controls via ptrace
- ELF64 symbols parsing
- Decoding of the A64 instructions in AArch64 execution state

NOT SUPPORTED YET

- Dynamic instrumentation and static binary rewriting are not supported on arm64
Dyninst arm64 Porting Work

- Code base clean-up
  - Migration from Kernel v4.0 (Fedora 21) to v4.9 (Fedora 25) or higher
    - Vsyscall page to vDSO
    - Remove work-around for an old arm64 kernel syscall bug
  - Binutil related bug fix

- Code generation and relocation (Sunny Shah is working on)
  - The “relocation” branch: https://github.com/dyninst/dyninst/tree/arm64/feature/relocation
  - Completion expected sometime early summer 2017
Testsuite Package

- Repository is on GitHub: [https://github.com/dyninst/testsuite](https://github.com/dyninst/testsuite)
  - Wiki pages help you get started
- Needs an x86 platform to redefine a set of tests due to the gprolog dependency, otherwise ok on arm64
- Verified test groups on arm64
  - `test_driver -pcontrol` (ProcControlAPI tests)
  - `test_driver -symtab` (SymtabAPI tests)
Outlook

- Dynamic instrumentation support on arm64 systems will appear in the v10.0 release time frame
- Migrating away from libdwarf to libdw (part of elfutils-libs) to drop a dependency
- Contributions are welcome
References

- Dyninst Project Homepage [http://www.dyninst.org](http://www.dyninst.org)
Thank you

- Bart Miller, Bill Williams, and Sunny Shah (U. of Wisconsin)
- Josh Stone (Red Hat)
- Andy Doan (Linaro)
Thank You

#BUD17

For further information: [www.linaro.org](http://www.linaro.org)

BUD17 keynotes and videos on: [connect.linaro.org](http://connect.linaro.org)