Jan 25, 2024 | RISC-V Perf Analysis SIG Meeting

Attendees: Beeman Strong tech.meetings@riscv.org Marc Casas

Notes

- **Attendees**: Bruce, Chun, Dmitriy, Robert, Snehasish, Victor, Atish, Guillem, Greg
- **Video** here
- Is this meeting using the new (2024) calendar?
    - Haven't been given a timeslot, so using the one that was most popular in our doodle poll. Could change.
    - Will we switch to 30 min like most other meetings?
        - We generally use the whole hour for our discussions, so likely will continue that unless told not to
- Intros:
    - Victor - interested in RISC-V, not HW person
- Snehasish - out for 3 next weeks, but contact on github if anything comes up
- Open Discussion
    - Have had a few member talks now, wanted to take time to discuss
- Robert - good talks, haven't digested everything yet
- Call stacks
    - Could a HW call-stack memcpy mechanism avoid the need to preserve frame pointers, saving that perf overhead?
    - Frame pointer gets pushed on stack, can also get it from DWARF but costly
- If we could change compiler to emit some other inst instead of tail calls, so CTR could record it, would that require an ABI change?
    - Atish: probably not an ABI change, just a new instruction encoding
    - Then could record the tail call, and next return would pop both the tail-call entry and the preceding entry. Would enhance RASEMU mode, better than SW stacks.
        - Could reduce the number of times we have to fall-back to SW method. But how to know when need to fall back?
    - Don't typically have call-stacks that go up/down by 40 levels during a sample period. Maybe 10s.
    - No similarly easy fix for setjmp/longjmp
        - Return mispredicts could indicate stack manipulation
    - Autofdo (etc) want branch records, about everything else just wants call-stacks. So could use RASEMU there.
        - Only the enhanced autofdo work from FB wants both together
        - But if you want to do it you have to compile with frame pointers, which means you pay for it all the time
        - If we had an enhanced RASEMU mode, could it make sense to use CTR for call-stack and trace for recent transfers?

            - Trace has more overhead, but only when profiling, and improved RASEMU might mean that common case speeds up thanks to no need for frame pointers. Good tradeoff?
        - Could consider some of this as a future extension, would want a POC (on Qemu?) to see if enhanced RASEMU gives us something better than SW stack
        - Google doesn't allow setjmp/longjmp in source, but does use C++ exceptions extensively
- Out of time
- Next meeting will be a talk from Matt Turner (Samsung) on HPC use of PMU

Action items

- ☐ Beeman Strong - Jul 28, 2022 - Reach out about proprietary performance analysis tools
- ☐ Beeman Strong - Jul 28, 2022 - Reach out to VMware about PMU enabling