Sep 26, 2024 | RV Performance Event Sampling TG

Attendees: Beeman Strong Snehasish Kumar tech.meetings@riscv.org

Notes

- **Attendees**: Beeman, Chun, Dmitriy, Punit, Atish, Snehasish, Bruce, Daniel, Ved
- **Slides/video** here
- **Reviewing Precise Event Sampling extension proposal**
- Event-based sampling is a useful alternative to Instruction Sampling, to more quickly identify instructions that incur rare events
- But making LCOFIs precise on non-spec events is hard for high-performance implementations, don't know overflow until at/near/after retire
    - Adds complexity, may have to run slower
- Propose 2 new CSRs
    - Sample PC
    - Sample Metadata
        - This can hold data used to derive the sample PC, if the above CSR isn't precise (e.g., only holds PC of first inst to retire in the cycle)
- Allows LCOFIs to skid, but SW can still collect the precise sample PC
    - But any other state collected would be impacted by skid
- perf would default to attributing sample to Sample PC CSR value, but implementations could include a custom function to use the Sample Metadata CSR to fix that up
    - Where would this custom function live?
        - Perf driver
- Are these CSRs per counter?
    - Proposing not. Have 2 CSRs per counter feels like overkill for handling the corner case of multiple overflows at once. Have more about this on later slide.
- Need call-stack to be reasonably precise too
    - Can still use Instr Sampling for cases where need other state to be precise. Just takes longer to get samples for rare events.
    - But can't use Instr Sampling for sampling on cycles, which is most common usage
        - Including retire delay cycles in Instr Sampling could identify instructions that take the most cycles
        - But want to track cycles per function, need to be able to attribute cycles to the right function
    - Could skid be bounded? Can't pass JALR?
        - Direct branches can change the call-stack too, in optimized code
        - But if you know current PC then you know the leaf
        - Still hard, could sample instr right before JALR
    - We could include an extension that indicates the implementation supports precise LCOFIs, ala ARM's synchronous performance event exceptions
- Google uses PMI interrupts on x86 (without PEBS) to get call-stack, which has skid

    - Skid got much better ~10 years ago
- Have looked at using LBR call-stack to verify leaf stack entries, but wasn't worth it
    - With larger skid this might be worthwhile
- Could use CTR to observe stack changes during skid
    - Not in RASEMU, but rather to see all branches between sample PC and LCOFI
- Intel has levels of precision in perf, we probably would too
    - Base: LCOFI sampling (with skid), use epc for attribution
    - More precise: LCOFI sampling (with skid), with precise PC, and possibly patched call-stack
    - Most precise: precise LCOFIs, or instr sampling
- In enterprise short functions are common, skid/imprecision causes problems
- In-line functions are tricky, would like to see them in call-stack, CTR and even call-stack can't help with these
    - Can distinguish them using debug info
    - If have precise PC then can get right debug info even with skid
- Worried that implementations will implement the simpler extension
    - Need to make hierarchy and value of each option clear
- Better to put complexity in SW than extra gates/power in HW
    - Correcting PEBS N+1 in SW is not reliable, need to keep SW complexity reasonable
- **Out of time**, will continue discussion next meeting
    - Will meet at least once before summit

Action items

- [ ]