Extending FirePerf to Userspace

Raghav Gupta, Alex Hao, Reza Sajadiany

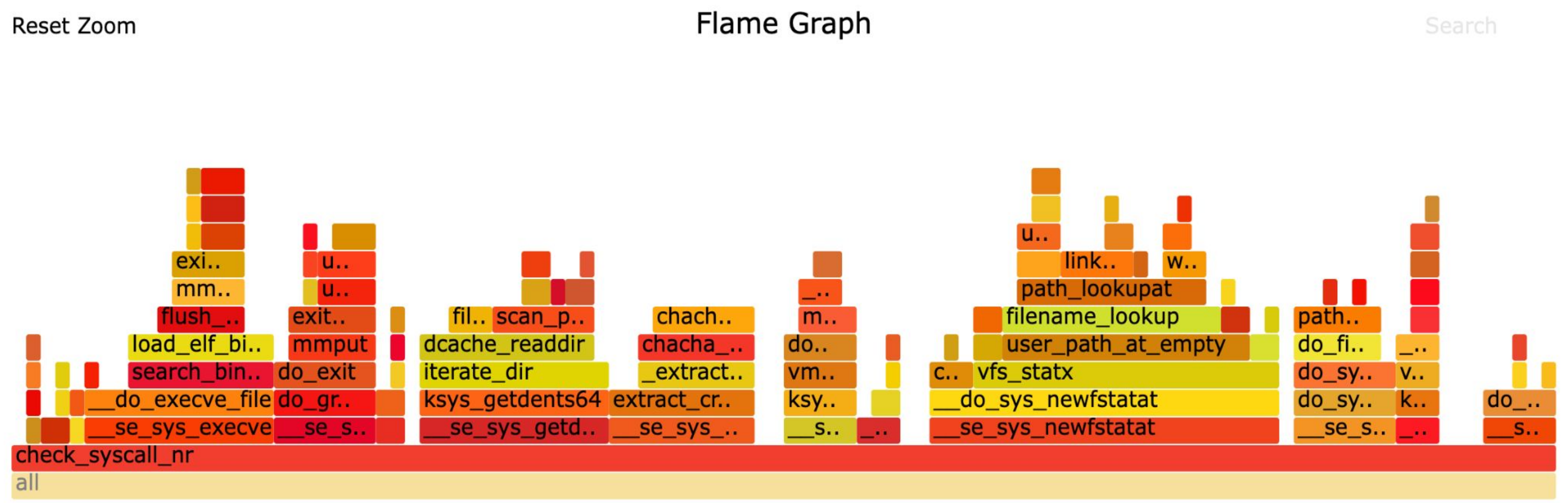
Full-Fidelity Full-Stack Performance Profiling using Realtime Call-Stack Reconstruction on Cloud FPGAs

Overview

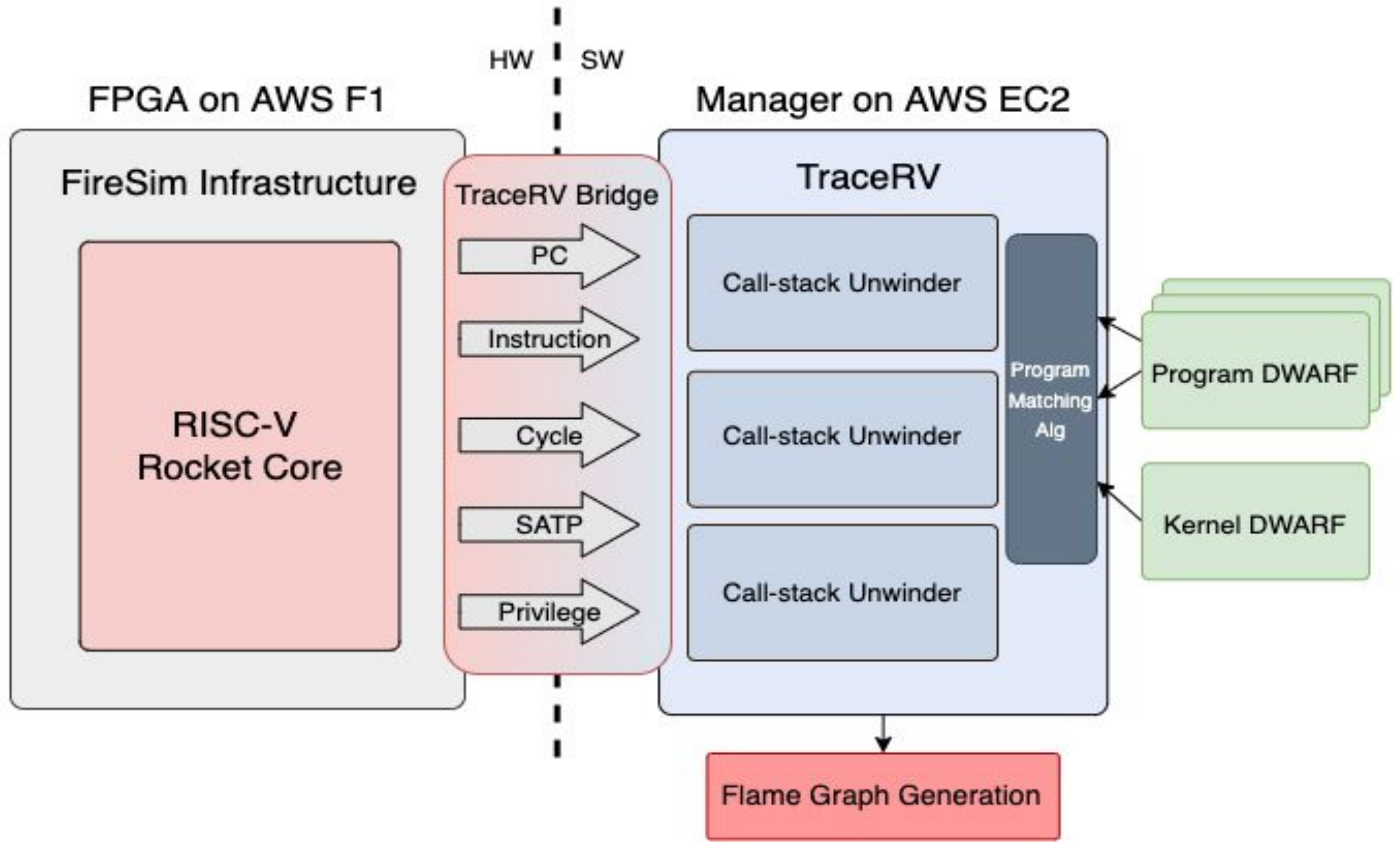
- ➢ High-fidelity introspection into HW/SW behavior via call-stack reconstruction and flame graphs
- ➢ Simply expose performance issues not discoverable by traditional software profilers
- ➢ Currently FirePerf only supports the kernel, we aim to provide full-stack support

Motivation

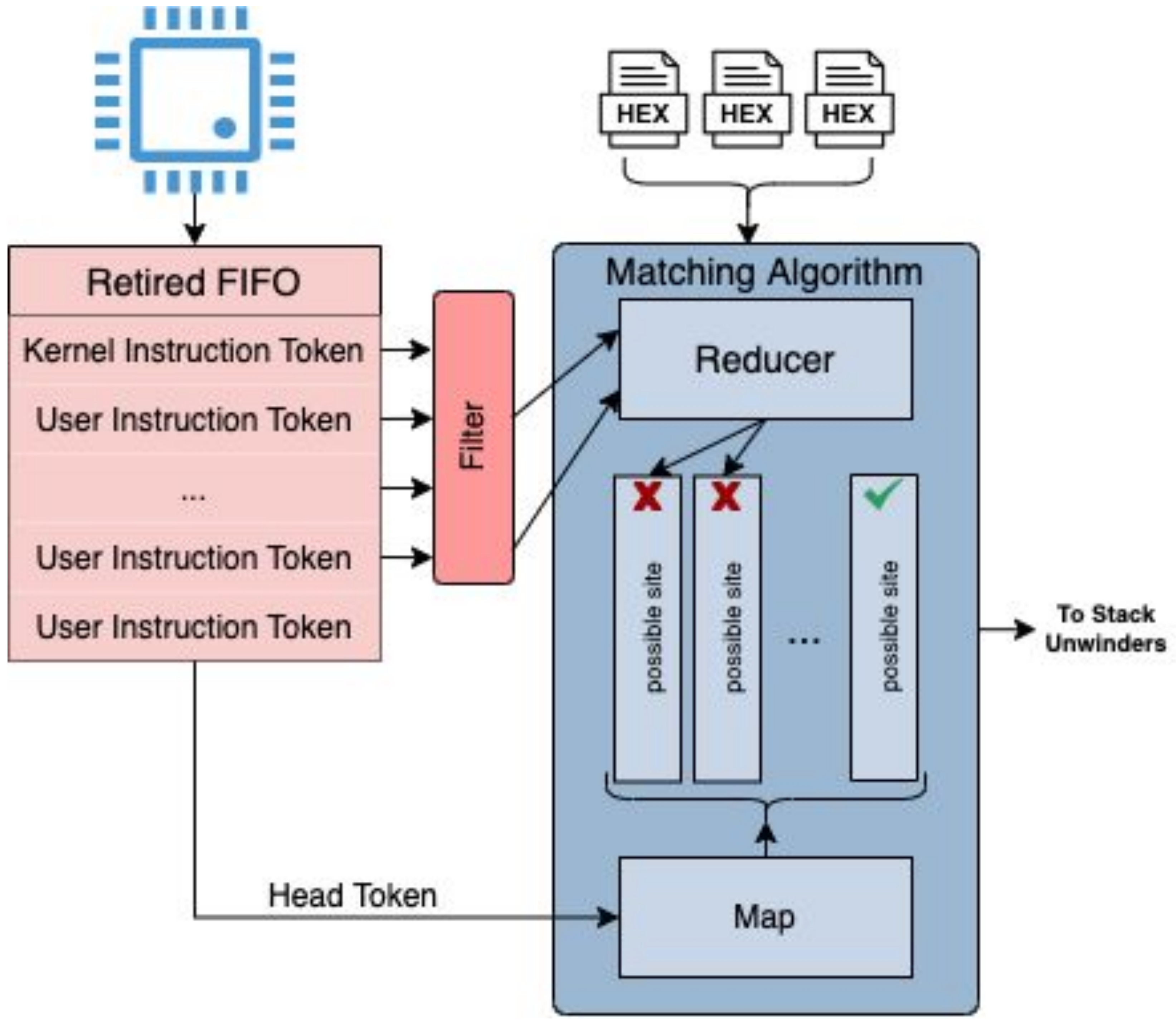
- ➢ Conventional profilers (perf, strace) are in-band
  - ○ Low sampling frequency
  - ○ Perturb target system at high frequency
- ➢ Out-of-band profilers completely decouple tool and target ⇒ minimal overhead but often slow
- ➢ Our final goal looks like this, but for user+kernel modes:



System Architecture



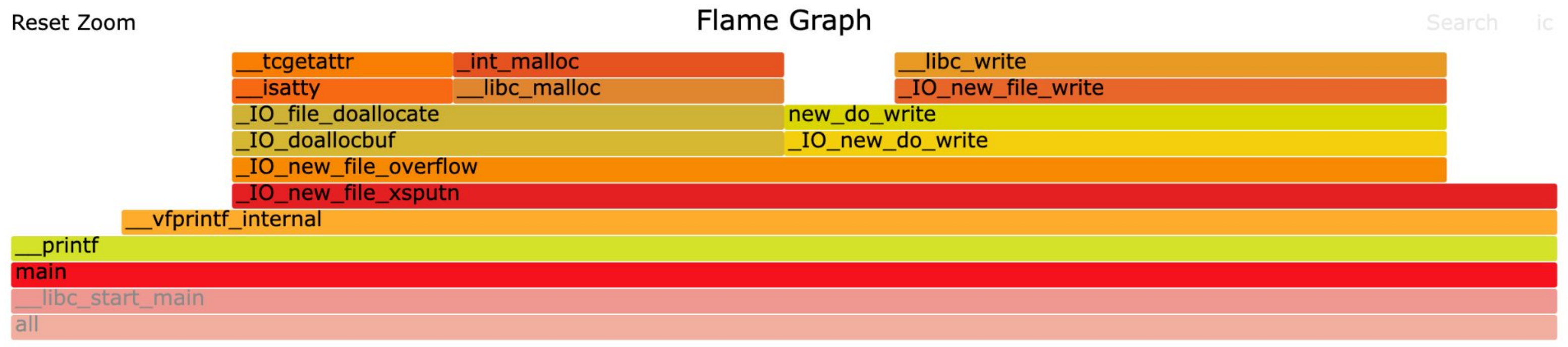
Matching Approach



- ➢ Need to match traced instructions to function names in source code
- ➢ Gather trace tokens from Rocket Core
  - ○ Instrumented trace port to provide {instr, pc, satp, priv}
- ➢ Buffer tokens in a queue ⇒ generate local history
- ➢ Match instructions per process with history + hexdumps
- ➢ Instruction offsets preserved under Virtual Memory
- ➢ For each instruction:
  - ○ Find possible sites (binary + page)
  - ○ Calculate offsets to other instructions
  - ○ Eliminate possible sites by checking matches at offsets

Proof-of-Concept

- ➢ Implementation tested with simple C program
- ➢ Successfully generated flame graph
- ➢ Core spends significant time in library write and malloc functions



- ➢ Next Steps
  - ○ Comprehensively test instruction matching and flame graph generation
  - ○ Evaluate on synthetic benchmarks and compare with perf, strace, KUTrace
  - ○ Perform case studies

Future Work

- ➢ Optimizations
  - ○ Propagate matches through history and verify mapping before use
  - ○ Cache satp ⇔ binary mappings, reduce possible sites
- ➢ Multithreading
  - ○ Threads share satp/PID
  - ○ Instrument target to differentiate
- ➢ Dynamic Linking and Position Independent Code
  - ○ DWARF/hexdump doesn't contain actual instruction addresses and jump offsets
  - ○ Instrument target + process linker for relative offsets and base addresses

Acknowledgements

Mentor Sagar Karandikar, SLICE Lab, FireSim and Chipyard Developers