EmbedSanitizer: Tool for Detecting Races in 32-bit ARM Applications

Hassan Salehe Matar, Didem Unat, Serdar Tasiran
{hmatar, dunat}@ku.edu.tr

① Introduction

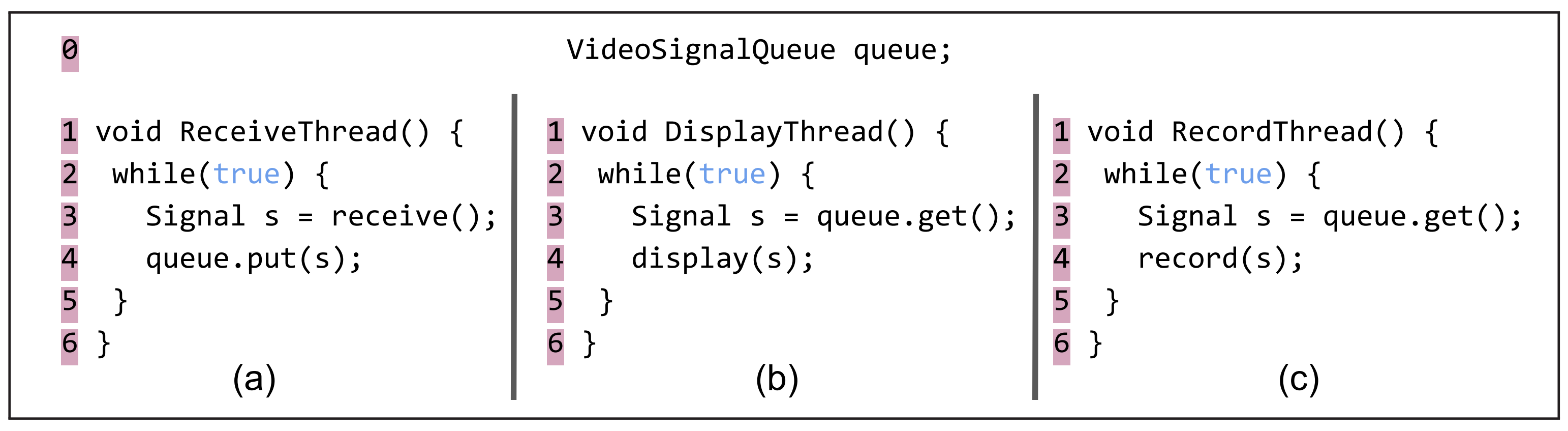
Multicore embedded systems are everywhere: smart phones, TVs, IoT. Thus, writing concurrent programs for these platforms is common. Unfortunately, it is easy to introduce data races in such applications during implementation. We propose a tool for detecting races in target embedded systems because of limitations in the literature:

- Available tools are architecture specific, e.g; ThreadSanitizer for 64-bit systems [1].
- Detecting races on another architecture may need emulation of some hardware specific features.
- Race detection through emulation is slow.

② Contributions

- EmbedSanitizer [3], a tool for detecting races in programs for 32-bit ARM.
- We motivate the idea of detecting races in target embedded systems and show usability.
- We evaluate our tool and show applicability using PARSEC benchmarks on a smart TV with ARMv7 CPU.

③ Motivation



```
VideoSignalQueue queue;

void ReceiveThread() {
 while(true) {
   Signal s = receive();
   queue.put(s);
 }
}
```
(a)

```
void DisplayThread() {
 while(true) {
   Signal s = queue.get();
   display(s);
 }
}
```
(b)

```
void RecordThread() {
 while(true) {
   Signal s = queue.get();
   record(s);
 }
}
```
(c)

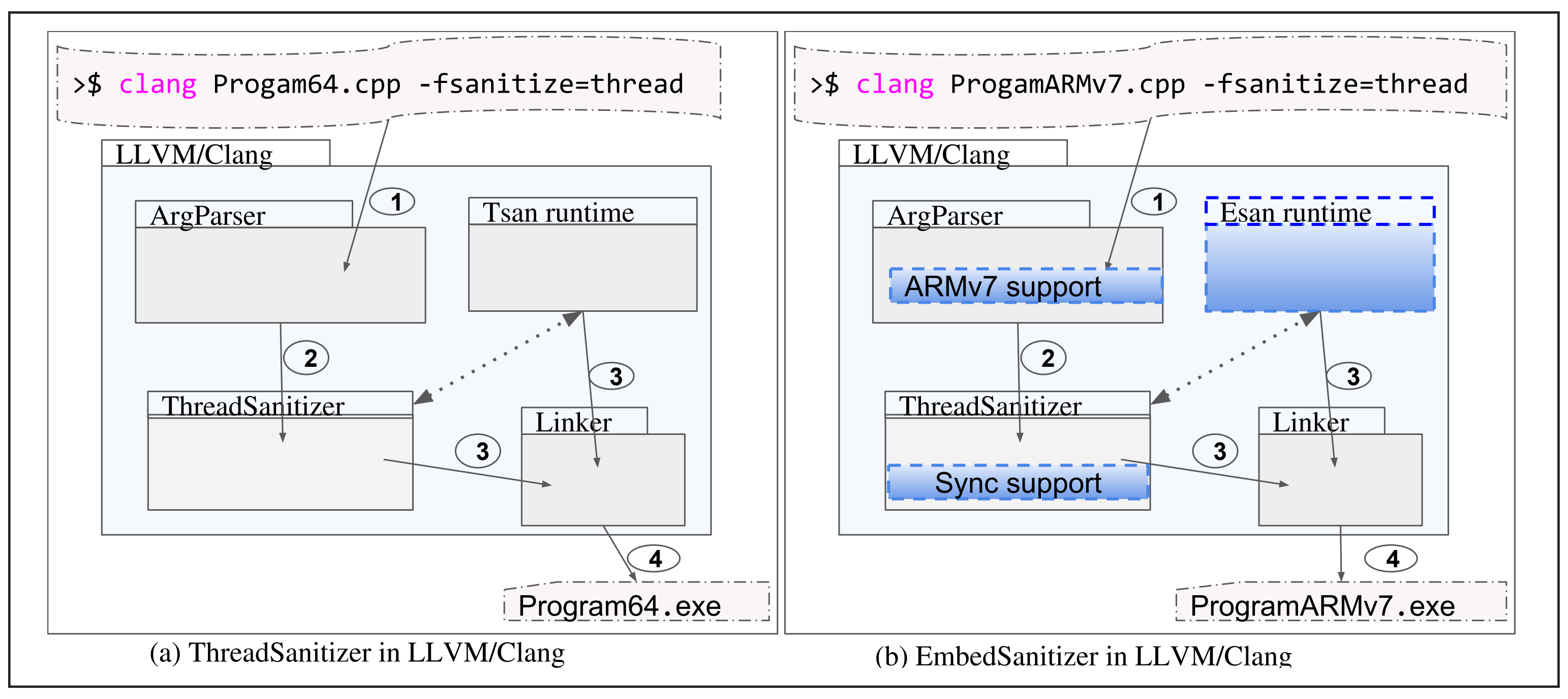
Figure 1: *A theoretical example with 3 threads concurrently accessing a shared queue queue while interacting with hardware peripherals: antenna, screen, and hard-disk.*

We promote utilization of existing race detection tools for embedded system architectures. The challenge is how to detect races while threads interact with peripherals, as in program on Figure 1. In the literature, the programmer has to emulate the hardware peripherals and then analyze the program against races on an alternative architecture or slow emulator. However, the use of target hardware for race detection improves detection precision and runtime performance.

④ Methods

Our tool is implemented as part of LLVM/Clang, in similar form to ThreadSanitizer, as shown on figure 2.

1. Modify the LLVM/Clang compiler arguments parser to support instrumentation of 32-bit ARM programs.
2. Implement instrumentation pass in to identify synchronization events and memory accesses.
3. Implement race detection runtime using FastTrack[2] race detection algorithm, for 32-bit platforms.
4. Cross-compile your 32-ARM program using Clang and execute the binary in a target hardware.



```
>$ clang Progam64.cpp -fsanitize=thread
```

```
>$ clang ProgamARMv7.cpp -fsanitize=thread
```

Figure 2: *High level overview of ThreadSanitizer and EmbedSanitizer in LLVM/Clang. In (a) ThreadSanitizer: essential LLVM modules for race detection. In (b) EmbedSanitizer: same modules we enhanced to instrument and detect races for 32-bit ARM.*

⑤ Evaluation

Precision Evaluation

Table 1: *Experimental results to compare race detection in ARMv7 using EmbedSanitizer vs in x86_64 with ThreadSanitizer.*

	Input						*ThreadSanitizer*	*EmbedSanitizer*
Benchmark	**size**	Threads	Addresses	Reads	Writes	Locks	Races	Races
blackscholes	4K options	2+1	28686	5324630	409590	0	NO	NO
fluidanimate	5K particles	2+1	149711	25832663	8457516	790	YES	YES
streamcluster	512 points	2+1	11752	21710589	352605	2	YES	YES
swaptions	400 simulations	2+1	243945	11000763	3377226	0	NO	NO

Performance Evaluation

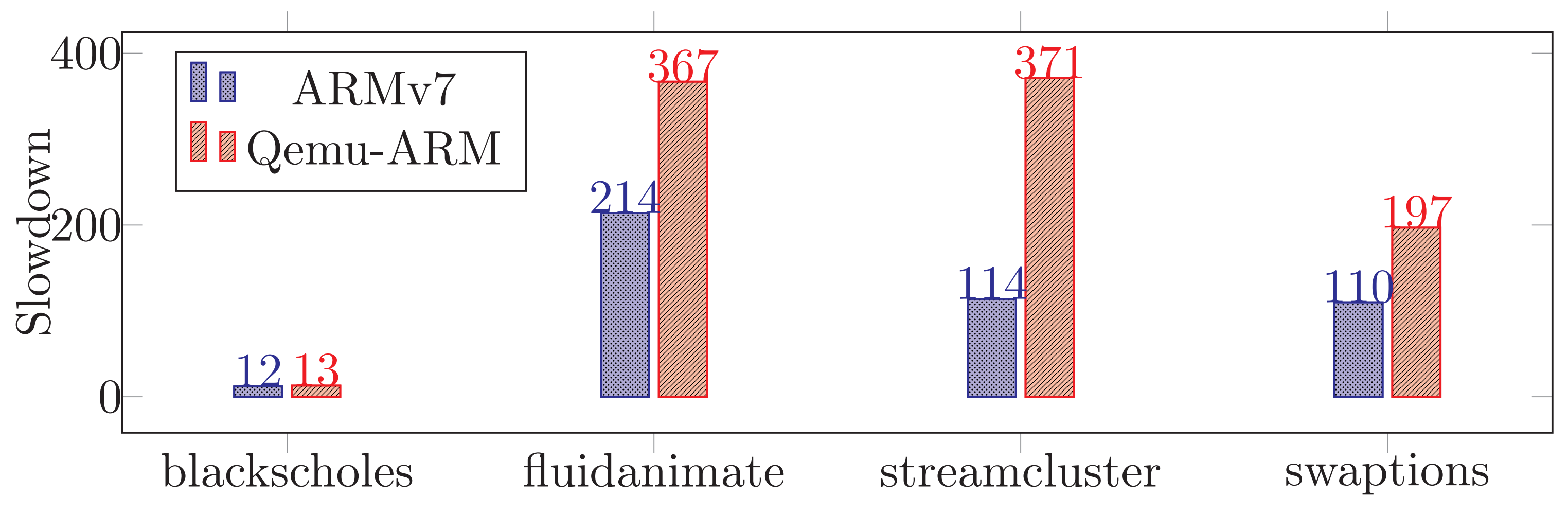


Figure 3: *Slowdown comparison of race detection on ARMv7 vs on Qemu-ARM.*

⑥ Conclusions

- We propose *EmbedSanitizer*, a tool for detecting races in 32-bit ARM applications.
- We use 4 PARSEC applications to evaluate its precision.
- We compare its slowdown on Qemu vs on target hardware.

⑦ Future Work

- Improving the efficiency of race detection runtime by using a hybrid of race detection algorithms.
- Supporting other 32-bit architectures.
- Evaluating with real-world applications which use features of the embedded systems such as sensors and actuators – a real motivation for our work.

⑧ References

[1] Konstantin Serebryany and Timur Iskhodzhanov. Threadsanitizer: Data race detection in practice. In *The Workshop on Binary Instrumentation and Apps*, USA, 2009.

[2] Cormac Flanagan and Stephen N. Freund. Fasttrack: Efficient and precise dynamic race detection. PLDI '09, New York, 2009.

[3] Embedsanitizer: www.github.com/hassansalehe/embedsanitizer.

[4] Christian Bienia. *Benchmarking Modern Multiprocessors*. PhD thesis, Princeton University, January 2011.