Towards a CHERI-Enabled GPU

CAPcelerate Team
University of Cambridge

Background

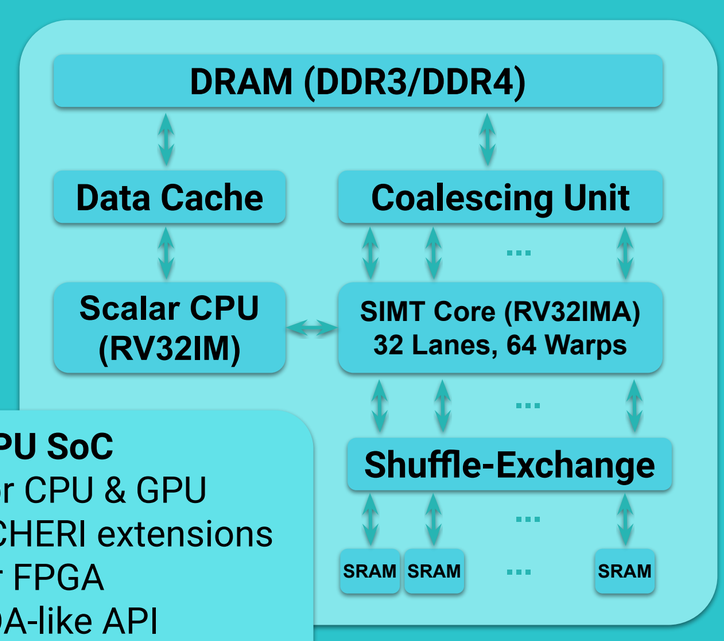
```
// CUDA device kernel
__global__ void overread() {
  int data = 0xda1a;
  int secret = 0xc0de;
  int* ptr = &data;
  printf("Address of data: %p\n", ptr);
  printf("Address of secret: %p\n", &secret);
  printf("Secret: %x\n", ptr[1]);
}
```

```
Address of data: 0x3fffd30
Address of secret: 0x3fffd34
Secret: c0de
```

GPUs are widely programmed in C-based languages such as CUDA and OpenCL, inheriting **weak memory safety**. Undefined behaviours in programs can be exploited by attackers to execute arbitrary code or leak sensitive information. To combat this, we are exploring a CHERI-enabled GPU.

Approach

We have developed a **prototype GPU** based on the Single-Instruction Multiple-Threads (**SIMT**) model popularised by NVIDIA and AMD. It implements the **RISC-V ISA**, which will allow it to be targeted from existing CHERI compiler tools. On top of this, we have implemented a **CUDA-like C++ library** and a suite of benchmark kernels. The prototype is functional and exhibits high compute density on FPGA, and will allow us to experiment with various forms of CHERI extension.



RISC-V CPU + GPU SoC

- Unified ISA for CPU & GPU
- Baseline for CHERI extensions
- Optimised for FPGA
- Provides CUDA-like API

https://github.com/blarney-lang/pebbles

Challenges

```
; Threads executing in lock
; step will all set t0 to var
la t0, var
; If the address is scalar,
; so is the loaded value
lw t1, t0(0)
; If the inputs are scalar,
; so is the output
add t2, t1, t1
```

Scalarisation is a technique that detects & tracks **uniform vectors** and processes them on a single processor/lane, reducing power and storage requirements.

Naive integration of CHERI into a massively threaded GPU will lead to a large **register file storage overhead**. However, we hypothesise that threads executing in lock-step will often share capability meta-data. If so, a technique known as **scalarisation** should be able to reduce storage overheads significantly.

Engineering and Physical Sciences Research Council