P5.Transparent CPU-FPGA Control Flow Transfer in HPC

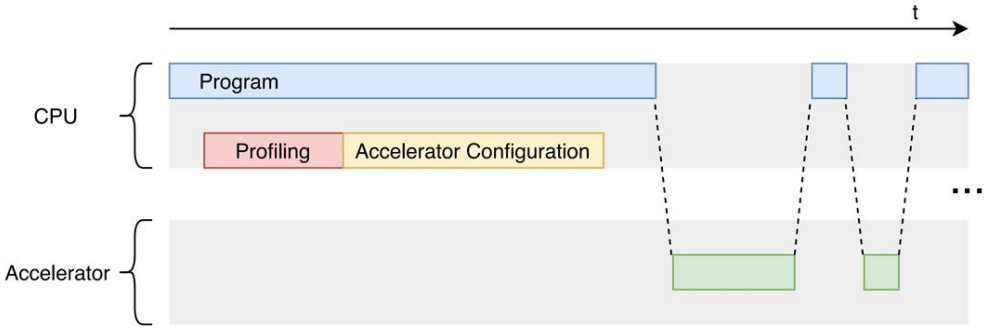
Daniel Granhão
CTM
INESC TEC

INESCTEC
TECNOLOGIA E CIÊNCIA
LABORATÓRIO ASSOCIADO
COORDENADO POR INESCPORTO

PEPCC

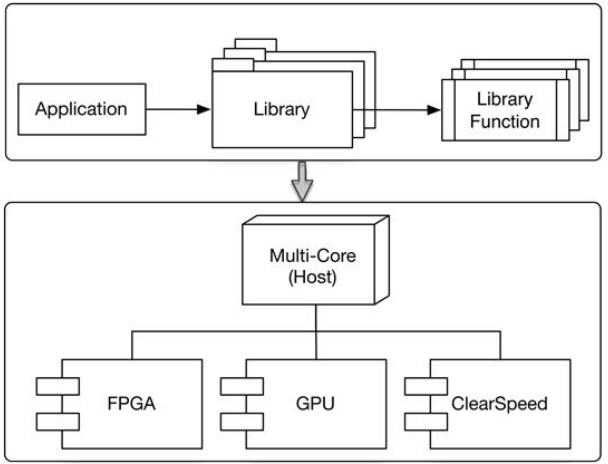
Context & Motivation

- HPC has become a huge industry
  - Key metric: Performance / Cost of Operation
- Heterogeneous platforms can help!
  - **Integrating FPGAs can provide big efficiency boosts**
- Problem: employing FPGAs requires expensive and complex application adaptation
  - **An automatic alternative is highly desirable**



Previous Work

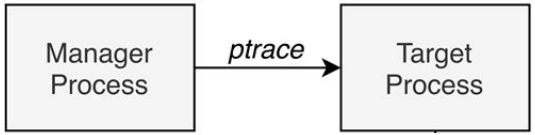
- There are several proposals for embedded platforms, but approaches are not transferable
- For HPC: Shared Library Interposing
  - Using LD_PRELOAD to load alternative shared libraries that move execution to accelerators
  - **Problem**: can only accelerate hot spots inside shared libraries



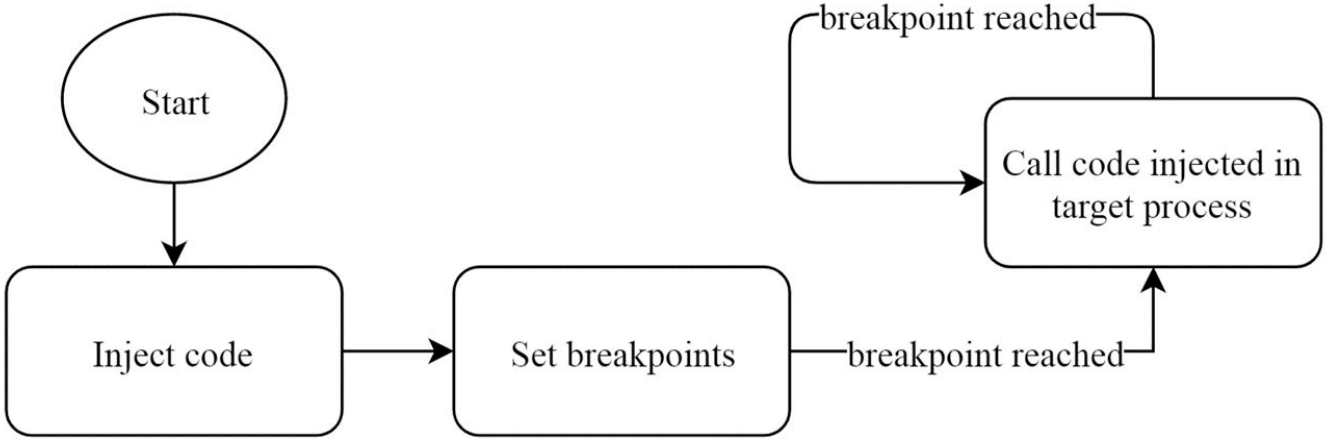
From: “Using Shared Library Interposing for Transparent Application Acceleration in Systems with Heterogeneous Hardware Accelerators” by Tobias Beisel, Manuel Niekamp and Christian Plessl

Proposed Mechanism

- Transfer the control flow by taking advantage of the *ptrace* system call
  - Linux system call that allows a process (the tracer) to control another (the tracee)
  - The tracer can change registers and memory of the tracee at will
- Idea: Introduce a new **manager** process that uses *ptrace* to track the execution of the target and makes the control flow move to an accelerator when a hot spot is reached



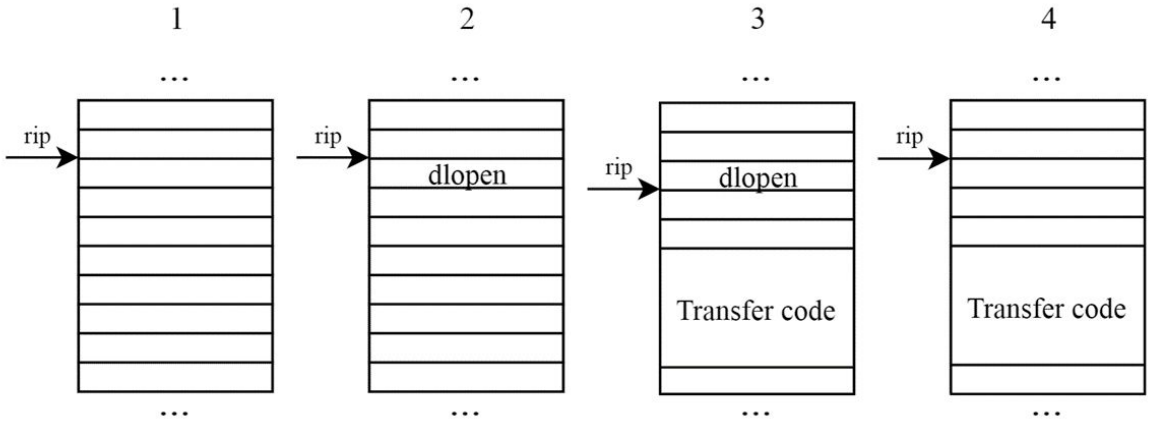
Manager Steps



Code Injection

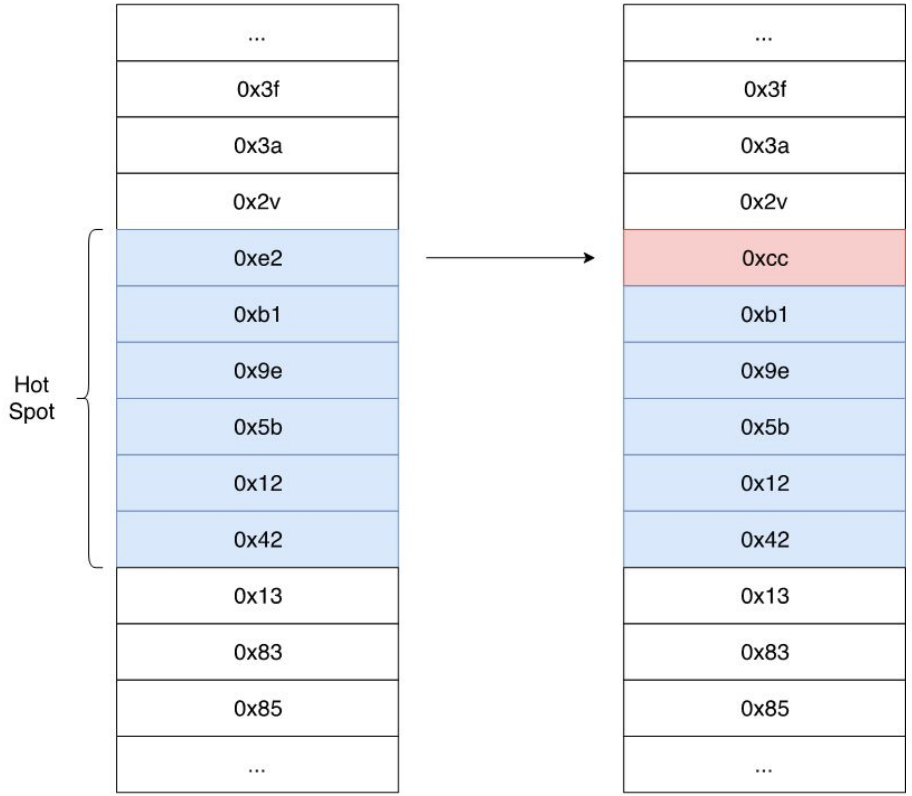
Can be made in two steps:

1. Compile transfer code into a shared library
2. Load shared library into target process using ptrace as depicted in the image



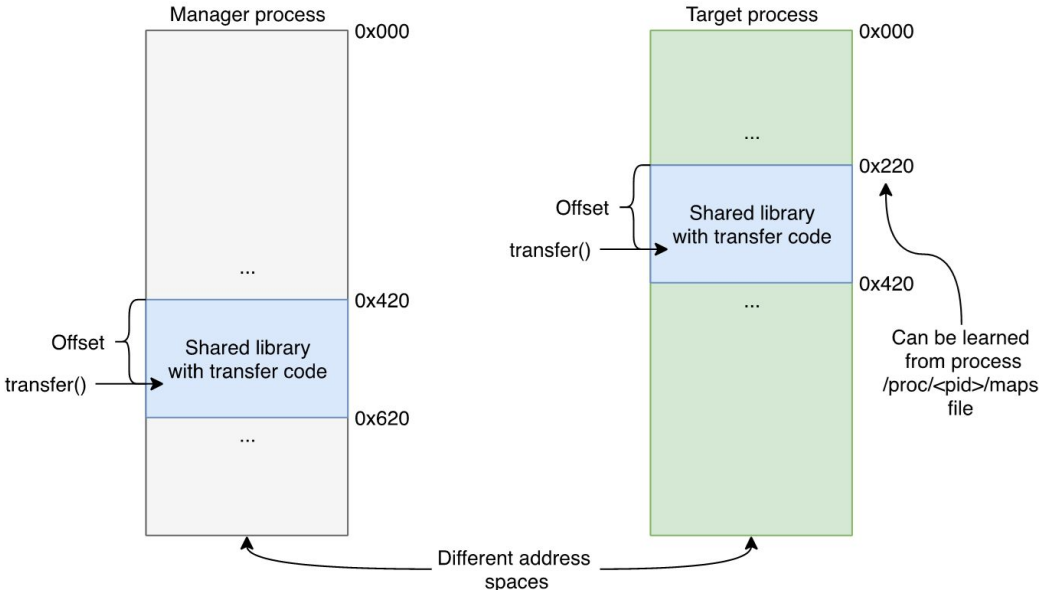
Breakpoint Configuration

- Identical to a debugger's operation



Injected Routine Execution

- Inject a call instruction using ptrace()
- But where to?
- Harder than it seems...
  - ASLR (Address Space Layout Randomization) randomizes location where shared libraries are loaded

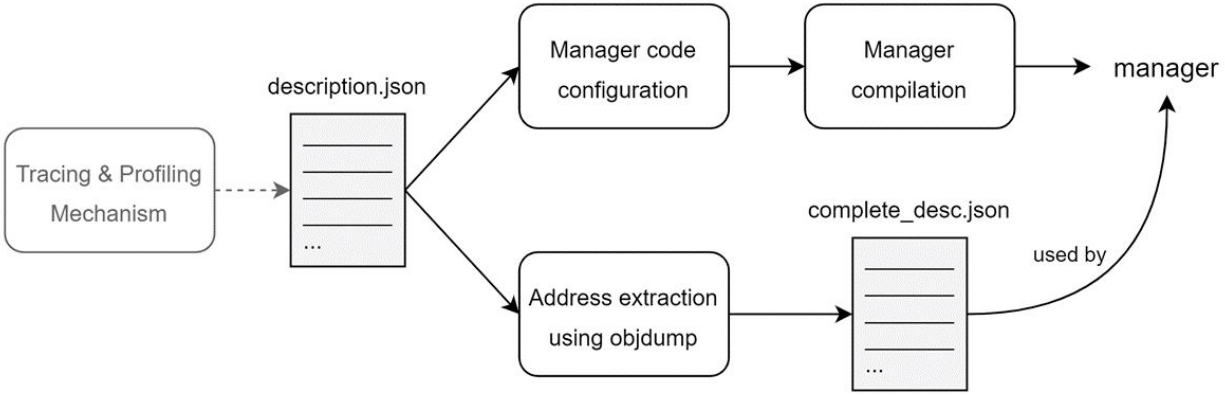


Evaluation

Comparing with Shared Library Interposing:

- Advantages:
  - Allows the transparent control flow transfer of hot spots that are not contained inside shared library functions
- Disadvantages:
  - Higher hardware dependency
  - Debugging hindered (despite being likely the target application is not in need of debugging)
  - Likely to introduce overheads

Proof of Concept Implementation

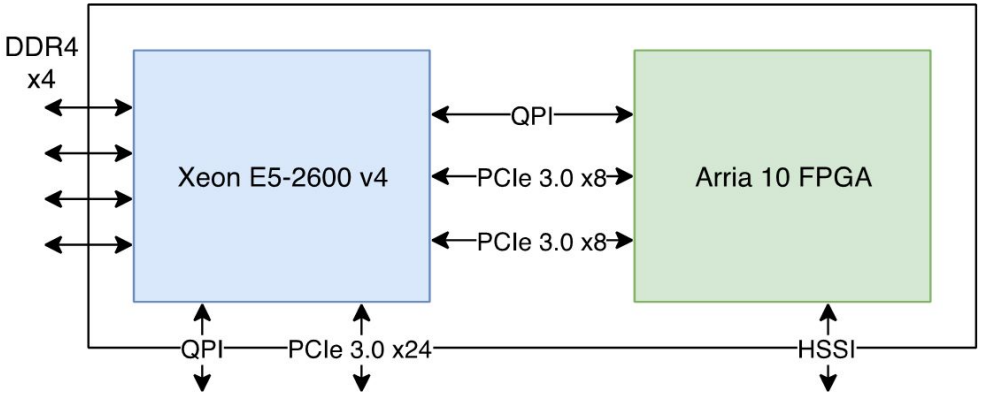


Parameter	Description
functionAddr	Address of the hot spot function
functionCalls	Array containing addresses where the hot spot function is called
targetName	Name of the target's process executable
functionName	Name of the hot spot function to be accelerated
functionArgs	Array describing each one of the hot spot function arguments
accLibString	Name of the shared library containing the transfer code
accLibPath	Path to the shared library containing the transfer code
accHeaderPath	Path to the header file of the shared library
accFunctionName	Name of the function containing the transfer code

Accelerator Requisites

- Must provide an interface in the form of a collection of CSRs
- Must itself access the main memory to access variable sized input data
- Must itself write to the main memory any variable sized output data
- Must use virtual addressing

➔ **Intel's Xeon+FPGA HPC platform**



Experimental Results

- Proof of concept implementation was applied to:
  - AES 256 CTR Mode Encryption
  - Matrix Multiplication
- Comparing with Shared Library Interposing:
  - Two types of overheads introduced:
    - An initial transfer code injection delay (not very important)
    - A per-control flow transfer adicional delay **(highly relevant)**

	Average Initial injection delay	**Average Per-transfer delay**
AES 256 CTR Mode Encryption	20 ms	**1.2 ms**
Matrix Multiplication	38 ms	**0.7 ms**

Conclusions

- *ptrace* can be used to transparently transfer the control flow of a process to an accelerator
  - **Hot spots do not have to be inside shared library functions**
  - **Although not explored in the proof of concept implementation, the approach should be applicable to other types of hotspots (such as “megablocks”)**
- The price to pay...
  - Higher dependency on hardware architecture
  - A variable overhead is introduced in each control flow transfer