Outline

- Introduction
- Dataflow programs and design space exploration
  - RVC-CAL model of computation
  - Design space exploration
  - Performance estimation
- Performance estimation methodology
- Experimental evaluation
- Conclusion and Future Work
Introduction

- Moore’s law is slowing down
- Explosion of multicore systems
- Scale-out by an increase in processor count is not enough anymore
- The trend is to go heterogeneous
- Heterogeneous systems are difficult to program
  - Analyze
  - Optimize
  - Portable

- Contributions and objectives
  - Computer-assisted software design for heterogeneous platforms
  - Accurate performance estimation for profiling on heterogeneous platforms
RVC-CAL model of computation

Actors:
- Computational kernel
- Communication via FIFO buffers
- Atomic execution of actions
- Encapsulate states variables

Actions:
- Consume/Produce tokens
- Can modify internal variables

```c
actor Producer () int i => int 0:
  uint counter := 0;
  p: action => 0!{counter} // ! means send
  guard counter < #
  do counter := counter + 1; end
end

actor CopyTokens (String name) int i => int 0:
  c: action I!{val} => 0!{val} end
end

actor PingPong () int i => int 0:
  p1: action I!{val} => 0!{val} end
  do println("PingPong(p1)! + val"); end
  p2: action I!{val} => 0!{-val} end
  do println("PingPong(p2)! - val"); end
  schedule fun a_p1:
    a_p1(p1) => a_p2,
    a_p2(p2) => a_p1;
end

actor Merger () int I1, int I2 => : uint counter := 0;
  m: action I1: v1, I2: v2 => do
    println("Merger(\"counter\")! = v1 +"++ v2");
    counter := counter + 1;
  end
end
```
RVC-CAL model of computation

- Direct code generation for software or hardware
- Explicit memory contention management
  - No data race
- Profiling capabilities
- **Turnus**, framework for design space exploration
  - Creates ETG through code interpretation
  - Use profiling weights
  - Performance estimation engine
- **Turnus**, framework for design space exploration
  - Creates ETG through code interpretation
  - Use profiling weights
  - Performance estimation engine
**Weights** are measured on the platform by the generated software implementation
- scheduling weights
- communication weights
- computation weights
Methodology - Clock-accurate profiling

- Sequential GPU partition
- Automatically code generation with instrumentation
  - RDTSC for intel
  - SASS code for Nvidia
- change code generation for separate read/write and compute
- Generate execution, scheduling, and communication weights

```c
void action() {

    Read FIFOs
    FIFO R1
    ...
    FIFO Rn

    Action body

    Write FIFOs
    FIFO W1
    ...
    FIFO Wm

    // -- PROFILING: TICK
    asm volatile("mov.u64,%0,%1\%clock64;" : =l(\_cycles_0) :: "memory");
    // action body code to be profiled ....

    // -- PROFILING: TOCK
    asm volatile("mov.u64,%0,%1\%clock64;" : =l(\_cycles_1) :: "memory");
    actor_a->profilingData->addFiringDevice(ACTOR_ID::actor_a, ACTION_ID::compute, (_cycles_1 - _cycles_0));
}```
Methodology - Heterogeneous estimation

- Heterogeneous weight
  - depends on the frequency
  - need to normalize across platform

- Clock volatility
  - disable boost, turbo boost, step speed or dynamic clocking
Methodology - Heterogeneous estimation

- Four different configurations are sufficient
  - CPU only
  - GPU only
  - CPU with cross-platform FIFO
  - GPU with cross-platform FIFO
Experimental evaluation - Setup

- Nvidia GeForce GTX 1660 SUPER GPU
  - 6 GB of memory
  - stabilized to 1.8GHz

- Intel Skylake i5-6600 CPU
  - 16 GB of DDR4 RAM
  - stabilized to 2.9GHz

- CUDA 11.3.1
Experimental evaluation - JPEG Decoder

- Normalized results
- Varying buffer sizes
  - 512
  - 1024
- Configuration
  - Src/Display CPU
  - others GPU
Experimental evaluation - JPEG Decoder

- Normalized results

- Varying buffer sizes
  - 512
  - 1024

- Varying configurations
  - 5 random mapping
Experimental evaluation - Smith-Waterman aligner
Experimental evaluation - Smith-Waterman aligner

- Normalized results

- Varying buffer sizes
  - 256
  - 1024

- Configuration
  - Src CPU
  - others GPU
Experimental evaluation - Smith-Waterman aligner

- Normalized results
- Varying buffer sizes
  - 256
  - 1024
- Varying configurations
  - 5 random mapping
Conclusion and Future Work

Conclusion:
- Methodology for accurately estimating the performance of dataflow application software modeled with the RVC-CAL programming language and executed on heterogeneous CPU/GPU co-processing platforms
- The results show the accuracy and effectiveness of the performance estimation over two application programs and a variety of configurations

Future Work:
- Adapt TURNUS heuristics to integrate the heterogeneous inputs into the automatic design space exploration framework
- Better model for the estimation of the communications between CPU and GPU