Using FPGAs as Microservices

David Ojika, Ann Gordon-Ross, Herman Lam, Bhavesh Patel, Gaurav Kaul, Jayson Strayer

(University of Florida, DELL EMC, Intel Corporation)

The 9th Workshop on Big Data Benchmarks, Performance Optimization and Emerging Hardware | ASPLOS 2018

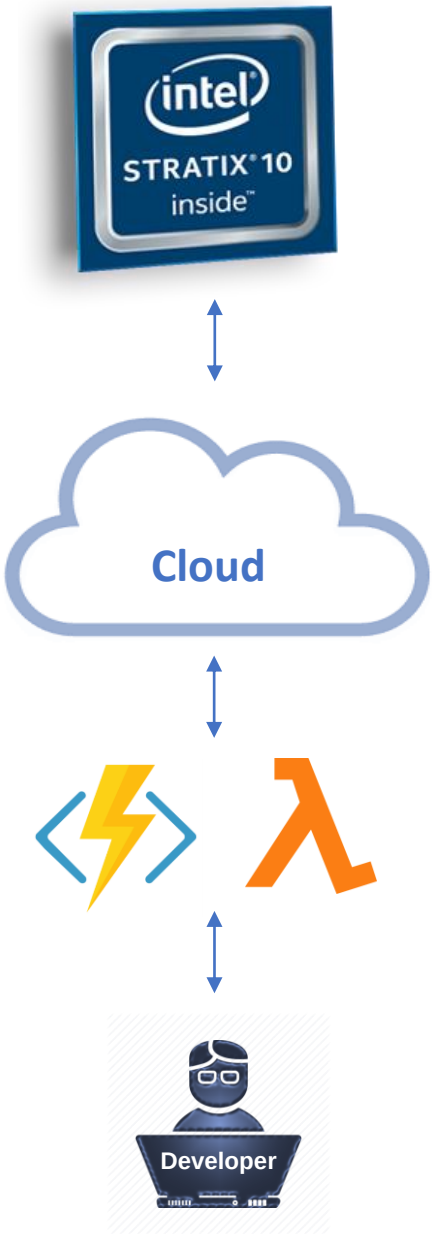
March 25, 2018

Contents

- Motivation
- Challenges of FPGA integration
- Proposed solutions
- Case study with Spark
- Key design elements
- Conclusions & future work

Motivation

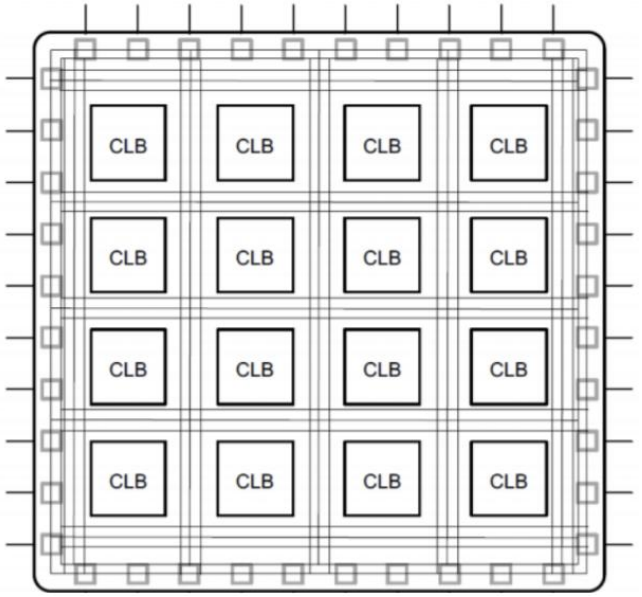
- FPGAs are increasingly being used in low-latency / high throughput computing (e.g., big data, machine learning)
- Cloud service providers (CSPs) continue to introduce FPGAs in their datacenters (e.g., Amazon, Microsoft) to meet computing demands
- Emerging cloud computing trends (microservices, serverless) show promise in hyperscale datacenters, increased developer productivity



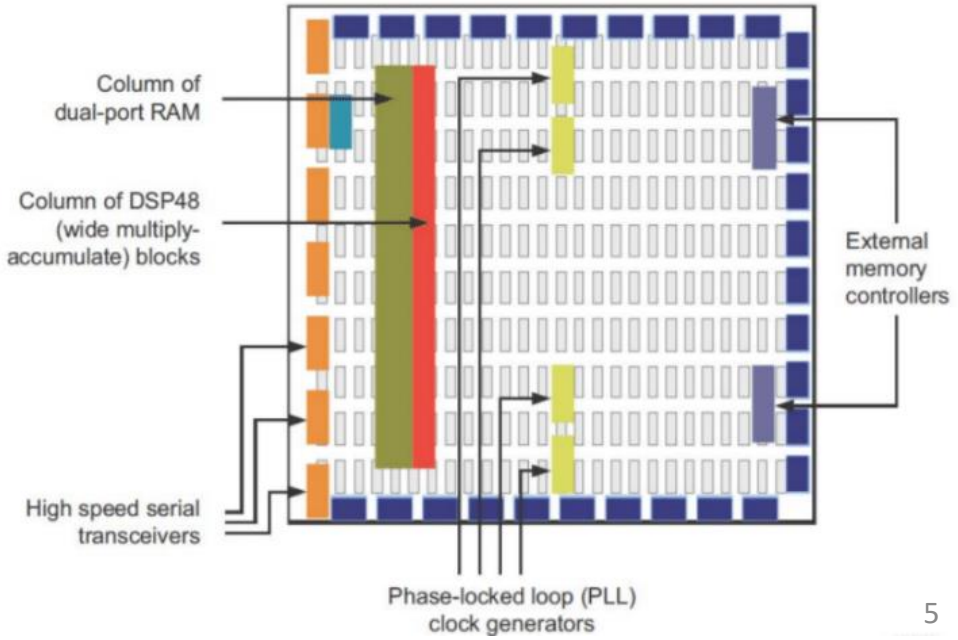
FPGA + Microservices ?

Overview of FPGAs

- FPGA: Field-programable gate array
- A reconfigurable chip composed of CLBs
- Basic structure consists of:
  - Look-up table (LUT), Flip-Flop (FF), Routing wires, I/O pads
- Contemporary FPGAs architectures incorporate more advanced resources:
  - Multiply-accumulate (MAC)
  - Off-chip memory controllers
  - High-speed serial transceivers
  - Embedded, distributed memories
  - Phase-locked loops (PLLs)
  - Up to 2 million logic cells



CLB: configurable logic block

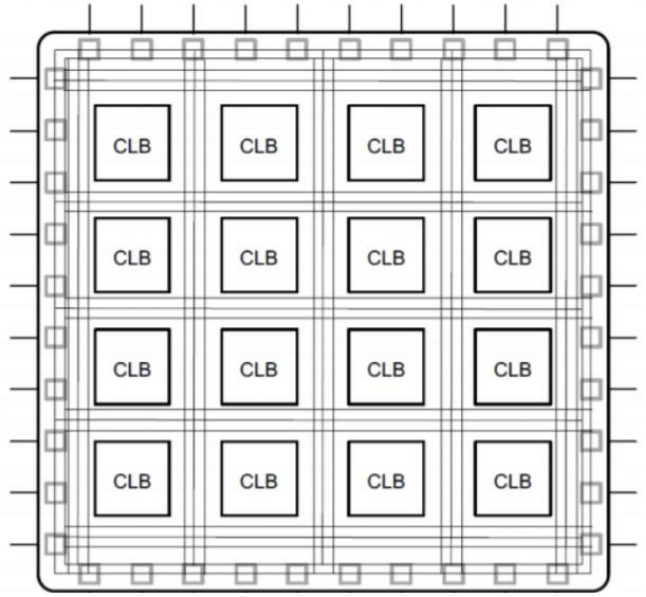


Overview of FPGAs

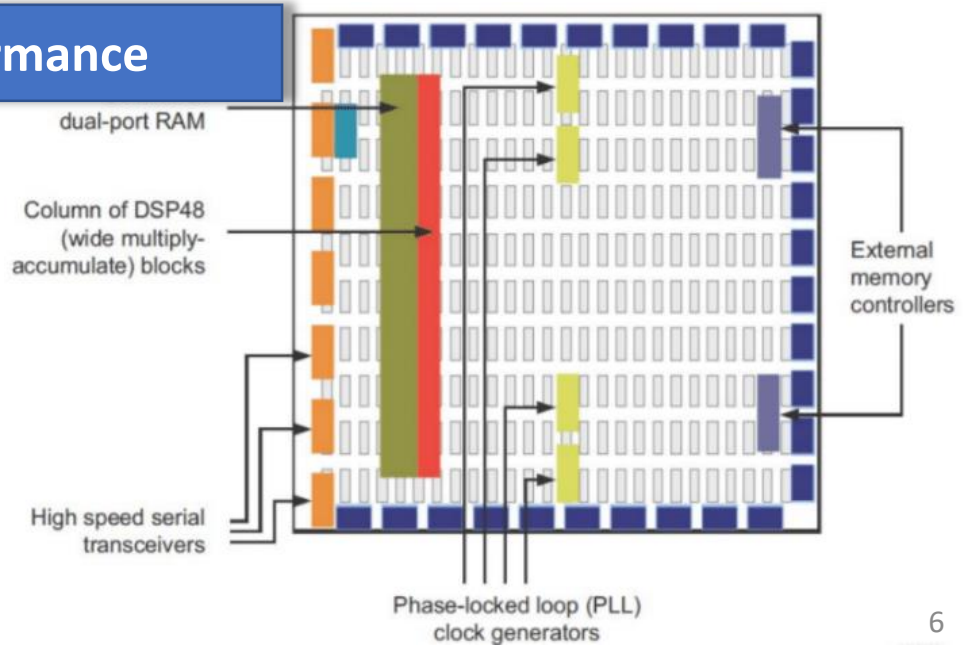
- FPGA: Field-programable gate array
- A reconfigurable chip composed of CLBs
- Basic structure consists of:
  - Look-up table (LUT), Flip-Flop (FF), Routing wires, I/O pads

Key benefits: Flexibility, Power, Performance

- Contemporary FPGAs architectures incorporate more advanced resources:
  - Multiply-accumulate (MAC)
  - Off-chip memory controllers
  - High-speed serial transceivers
  - Embedded, distributed memories
  - Phase-locked loops (PLLs)
  - Up to 2 million logic cells

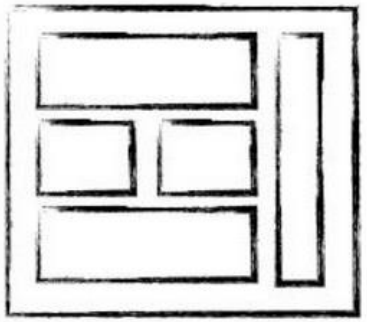


CLB: configurable logic block

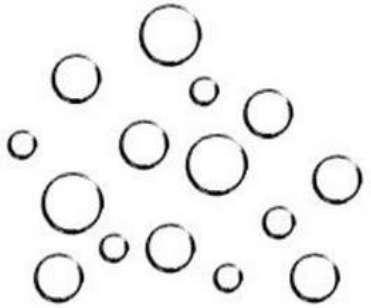


What are Microservices

- Microservices*:* a collection of loosely-coupled software services
- Unlike *Monolithic*, provide lightweight (shared) services
- Scale consistently with changing workloads



Monolith / Layered



Microservice / Disaggregated

- Our approach: use Microservices *(as collection of loosely-coupled FPGA accelerator functions)* to support FPGAs as “software services”

What are Microservices

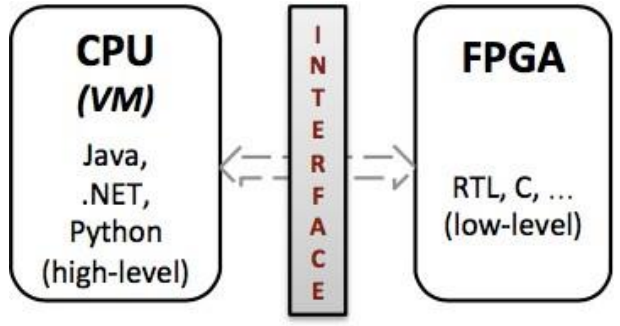
- Microservices*:* a collection of loosely-coupled software services
- Unlike *Monolithic*, provide lightweight (shared) services
- Scale consistently with changing workloads

- Our approach: use Microservices *(as collection of loosely-coupled FPGA accelerator functions)* to support FPGAs as "software services"

Challenges of FPGA integration

1. Intricate software interface

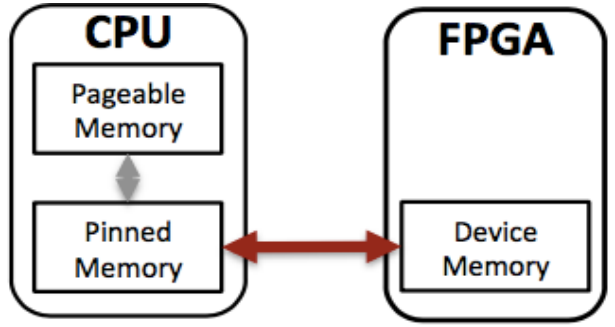
- JVM-to-FPGA interfacing is cumbersome
- How to keep FPGA accelerator fully-utilized



Expose all FPGA functions ?

2. Non-trivial FPGA sharing and data movement

- FPGA and CPU thread co-existence is complicated
- Data transfer overheads can hurt performance

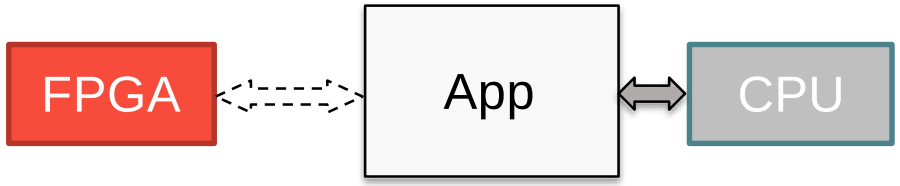


Low-latency, high-bandwidth link

FPGA mgt. & data movement ?

3. FPGA reconfiguration

- Reconfiguration can take milliseconds to a few seconds
- Certain applications may be intolerable to downtime

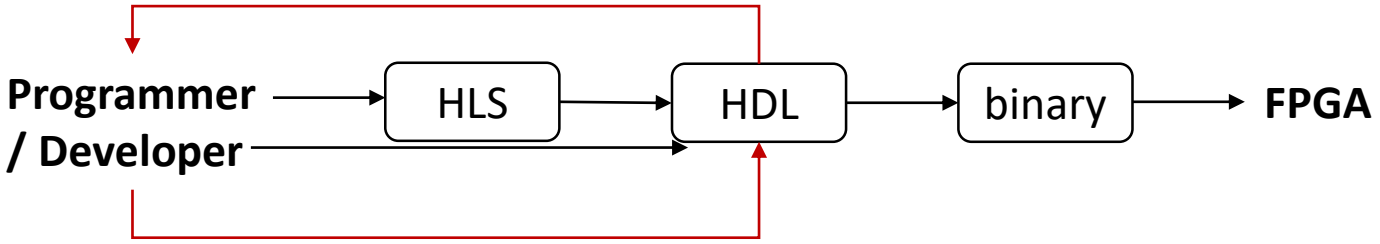


Hiding reconfiguration time

CPU fallback ?

4. Challenging programming model

- Requirement on hardware-specific knowledge
- Long synthesis (compile) time

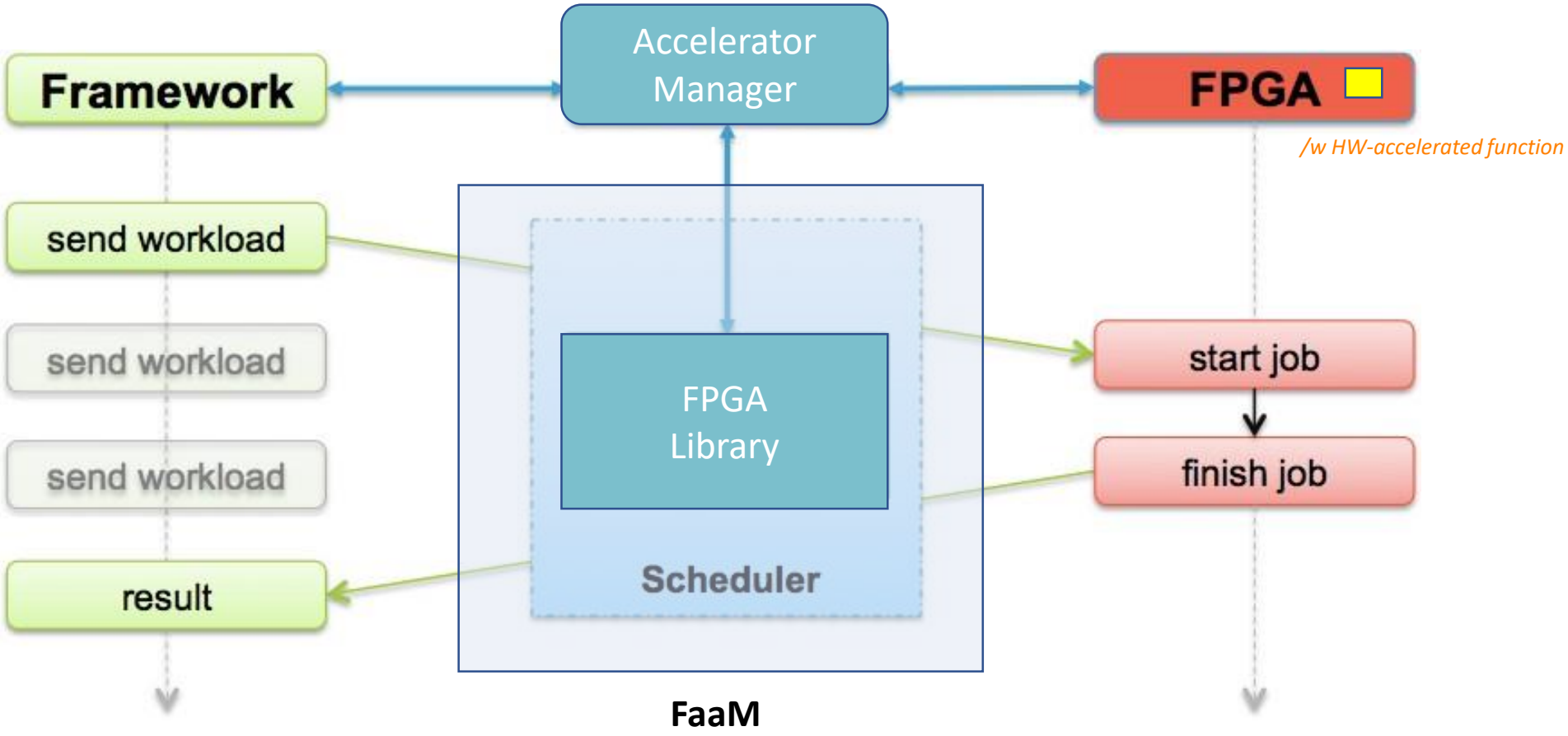


Programmer vs Developer ?

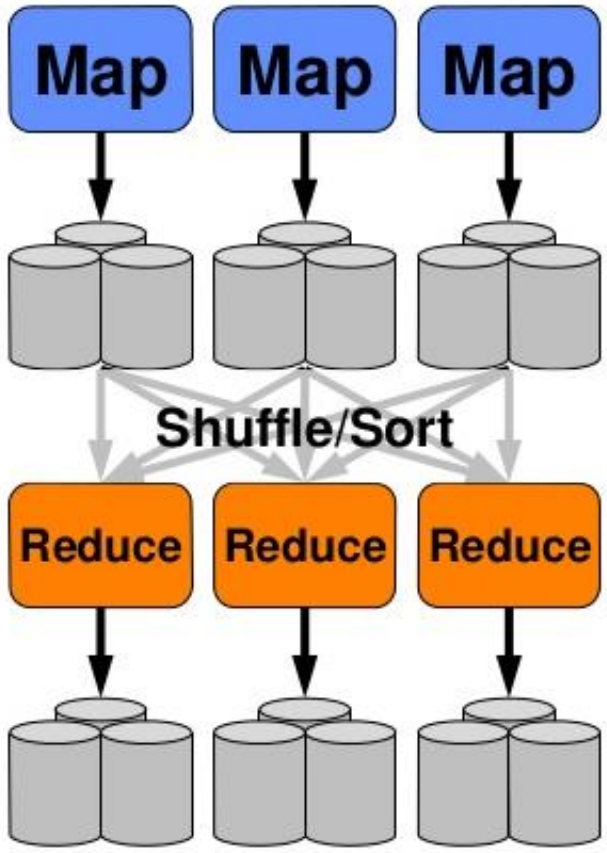
Proposed Solution

- FaaM: a runtime framework for deploying FPGA accelerators as microservices
  - Leverage optimized hardware libraries
  - Share accelerator across threads, applications, users
  - Platform-agnostic (OpenCL, etc.)
  - Seamless integration with Java, support wide range of apps/frameworks
  - *Takes care of accelerator management (init./setup, buffer mgt., etc.)*

FPGA Microservices Overview



Experimenting with Spark



Data Shuffle/Sort in Spark

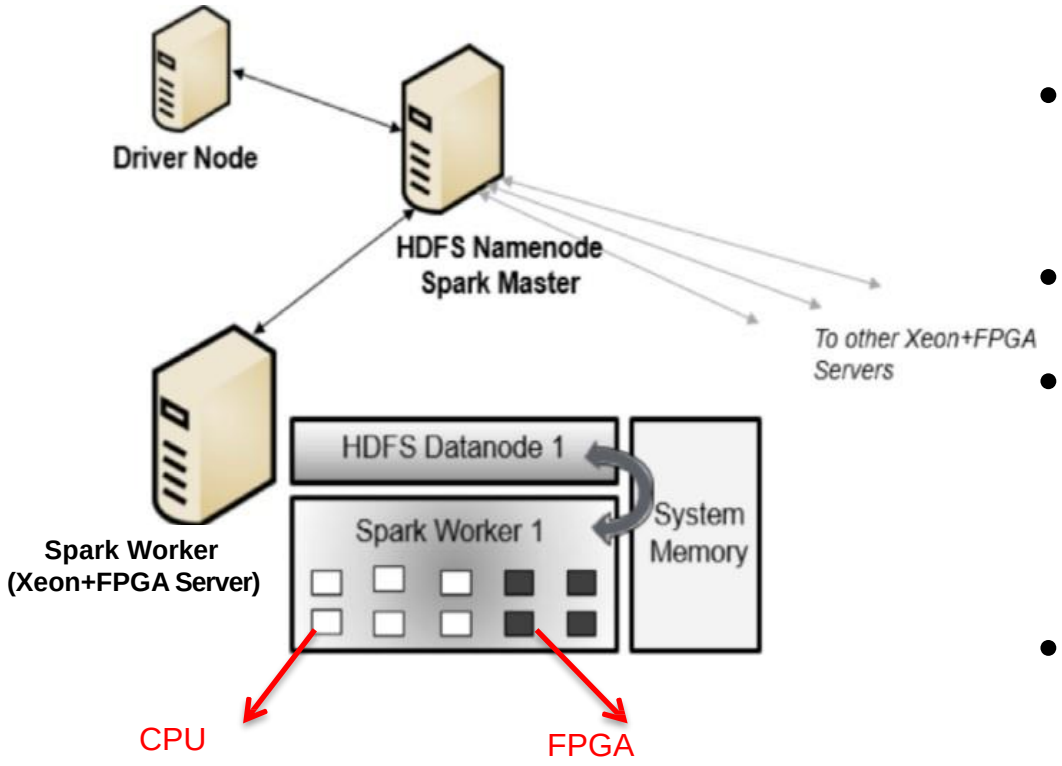
1. **Map** – convert raw input into key/value pairs. Output to memory (or disk)

2. **Shuffle/Sort** – All reduces **retrieve all records from all mappers over network**

3. **Reduce** – For each distinct key, do 'something' with all the corresponding values. Output to disk

…data movement can hurt performance

Experiment Setup

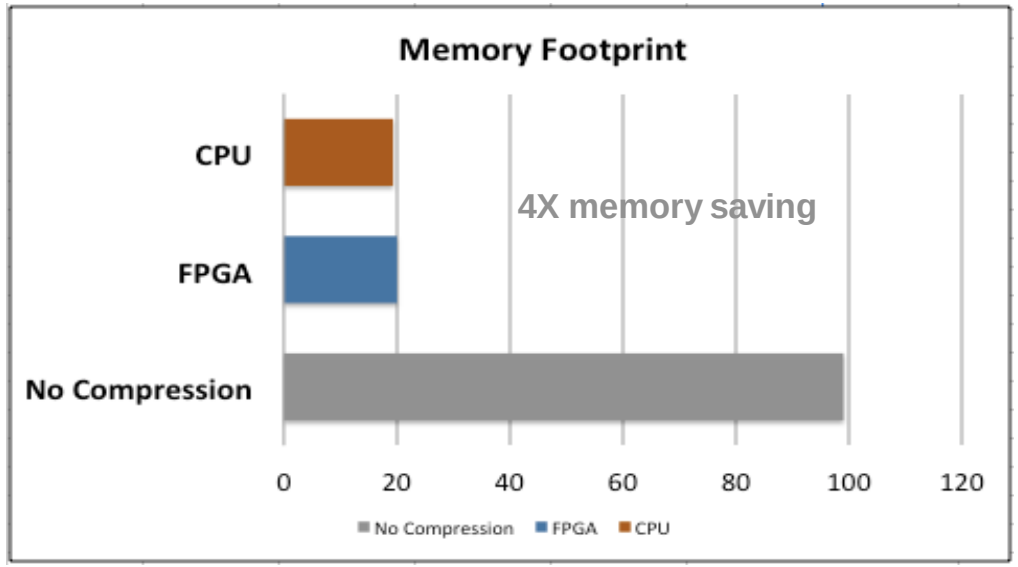


- **Platform:** Xeon+FPGA with Arria10 FPGA
- **Hardware-accelerated function:** DEFLATE compression algorithm (level 9)
- **Workload**: Sort
- **Evaluation technique**: Focus on compression of Spark *Output RDDs*

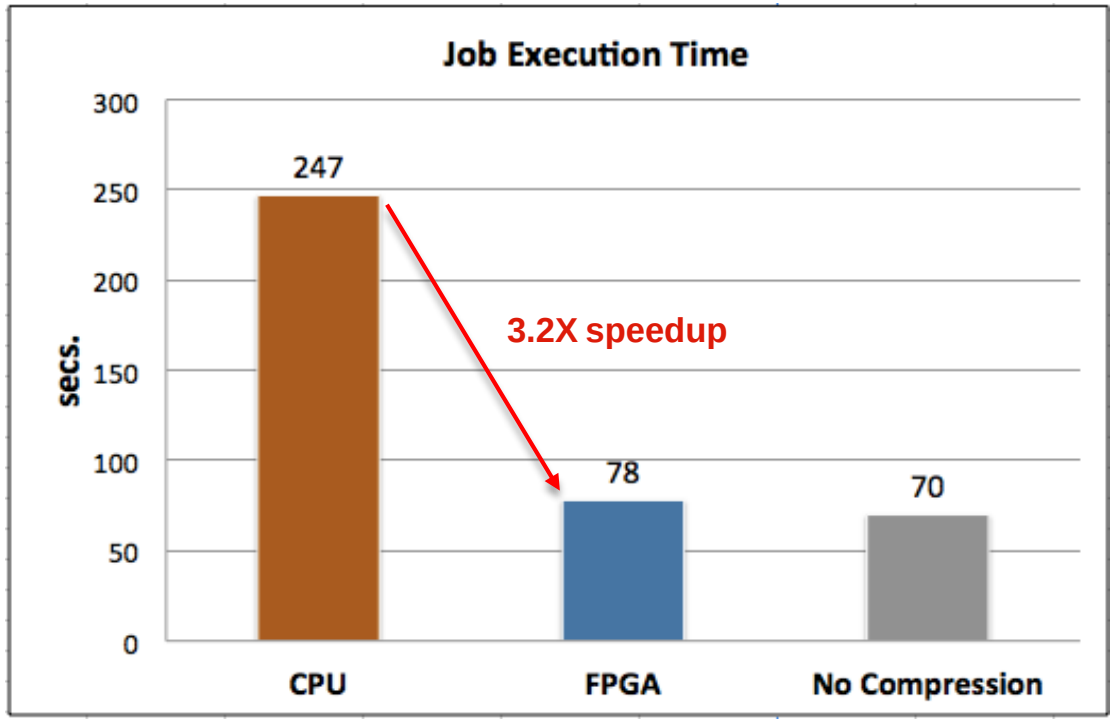
- **With FaaM (FPGA Microservice):**
  - No change to application code
  - Only update Spark config. files

Performance Results

FPGA as “drop-in replacement” via FaaM

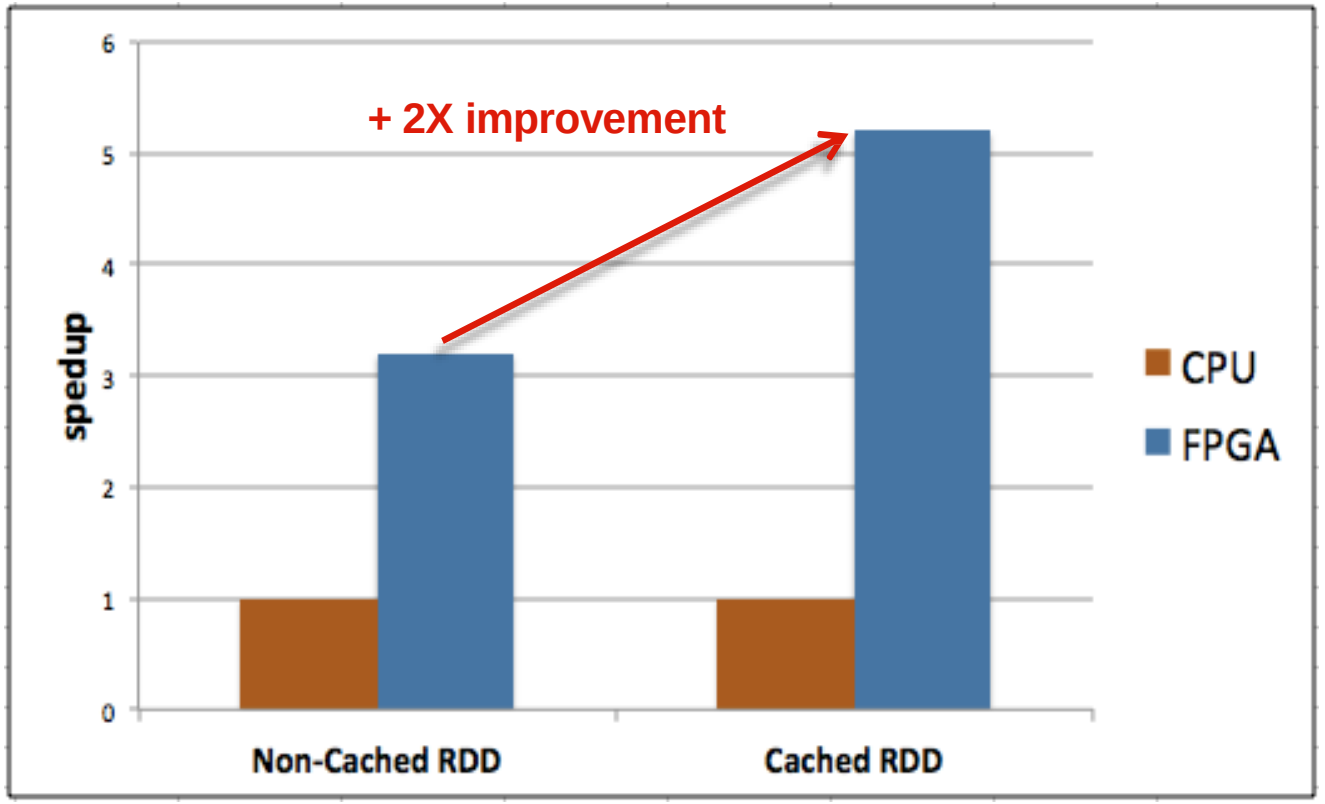


Functional Evaluation



Performance Evaluation

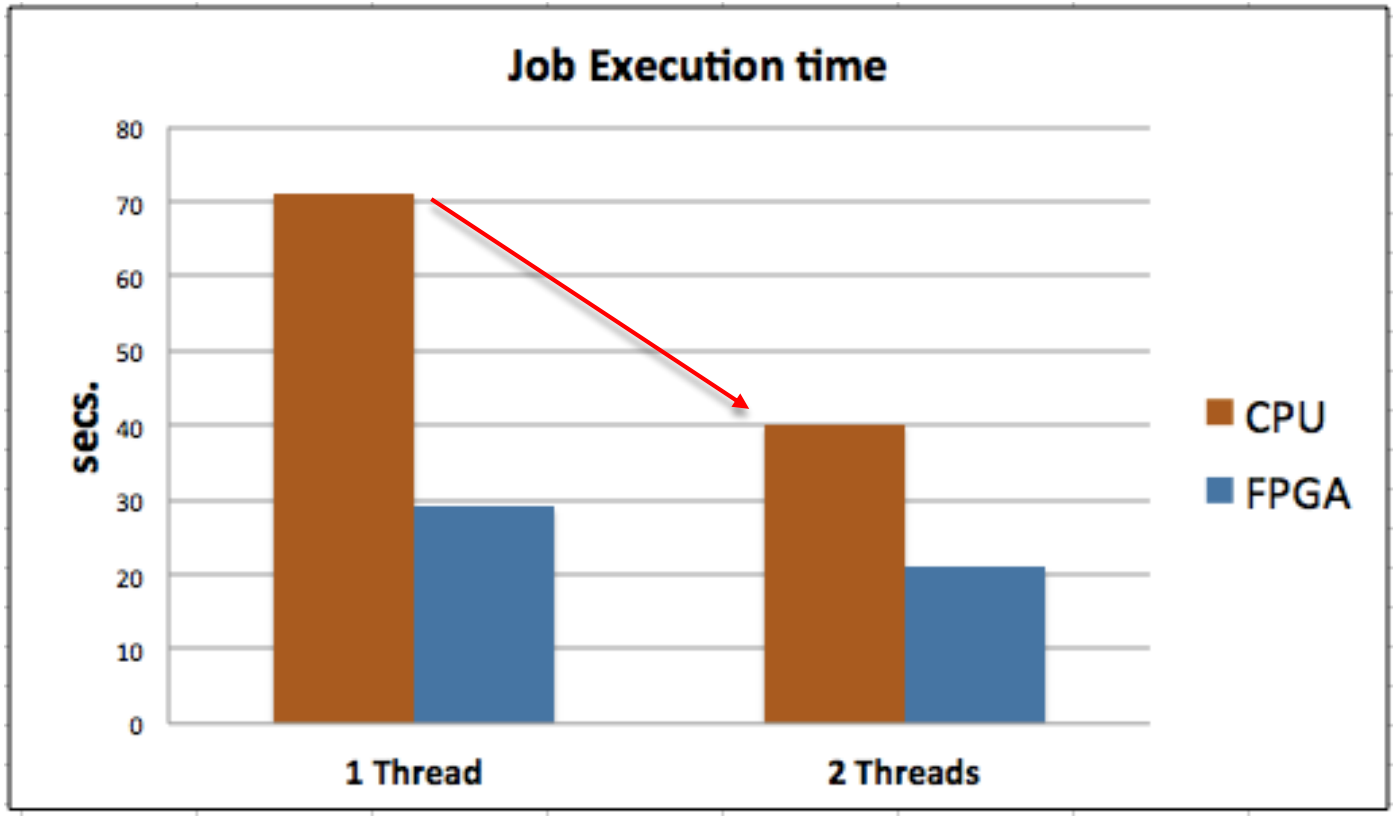
System-level optimization #1: *enable RDD caching*



**CPU used only as baseline.*

RDD: Resilient Distributed Disk

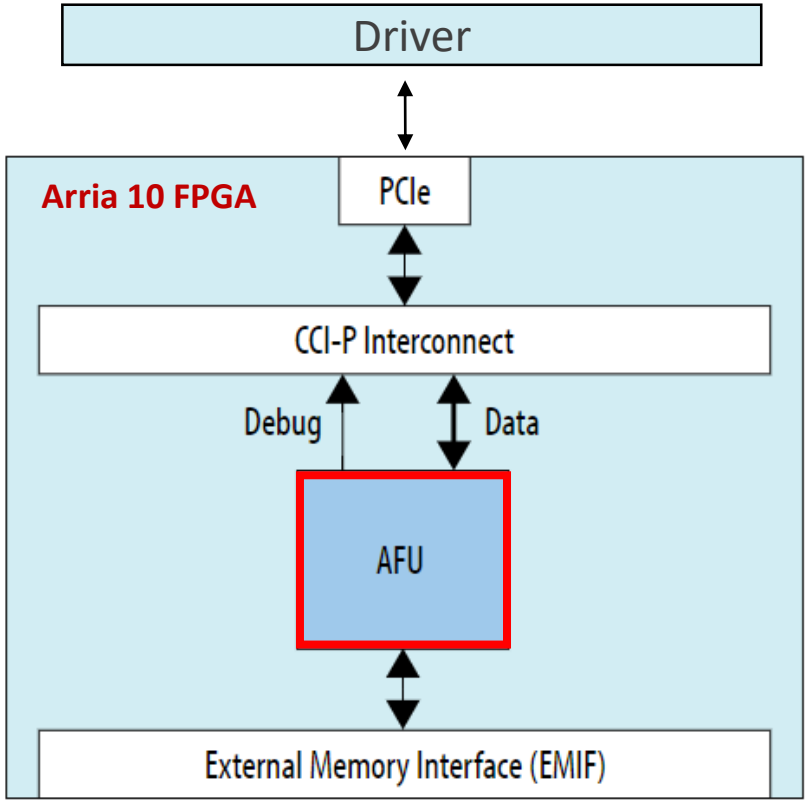
System-level optimization #2: *increase CPU count*



Increased FPGA accelerator utilization, resulting in improved application performance

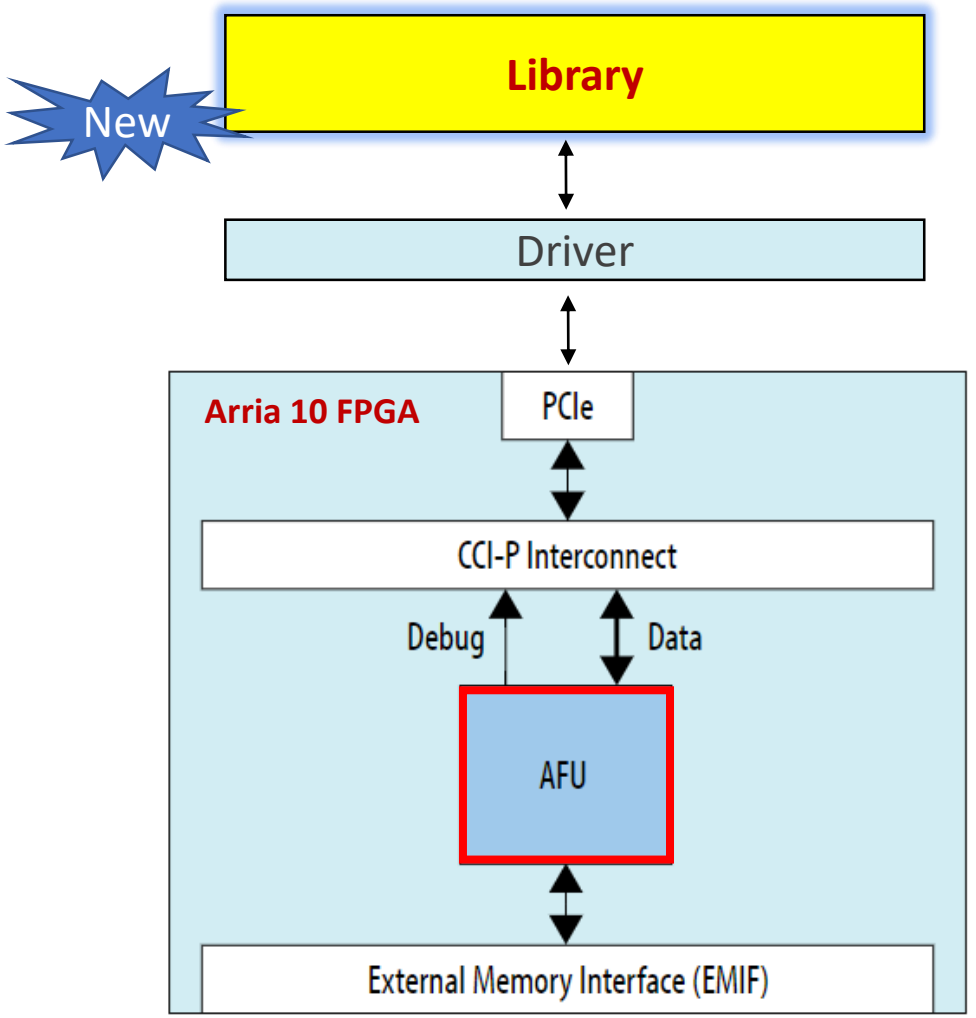
Key design elements

1. FPGA accelerator abstraction



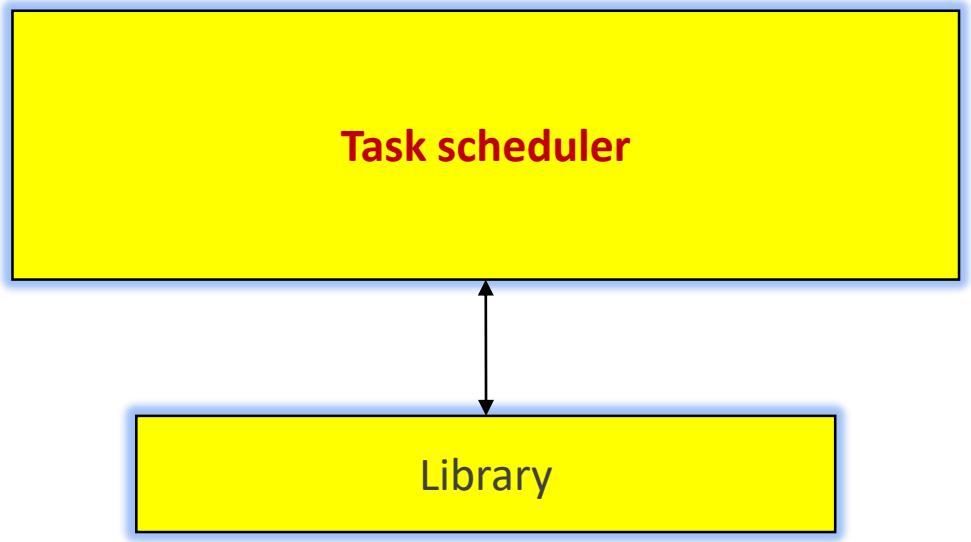
AFU: Accelerator Functional Unit

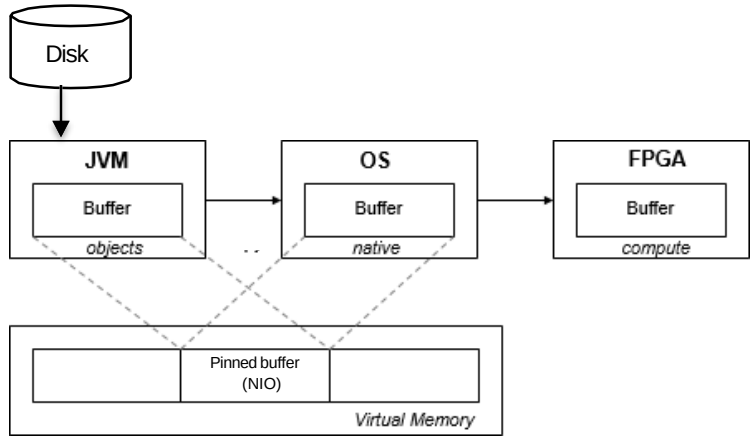
1. FPGA accelerator abstraction



AFU: Accelerator Functional Unit

2. Task scheduler

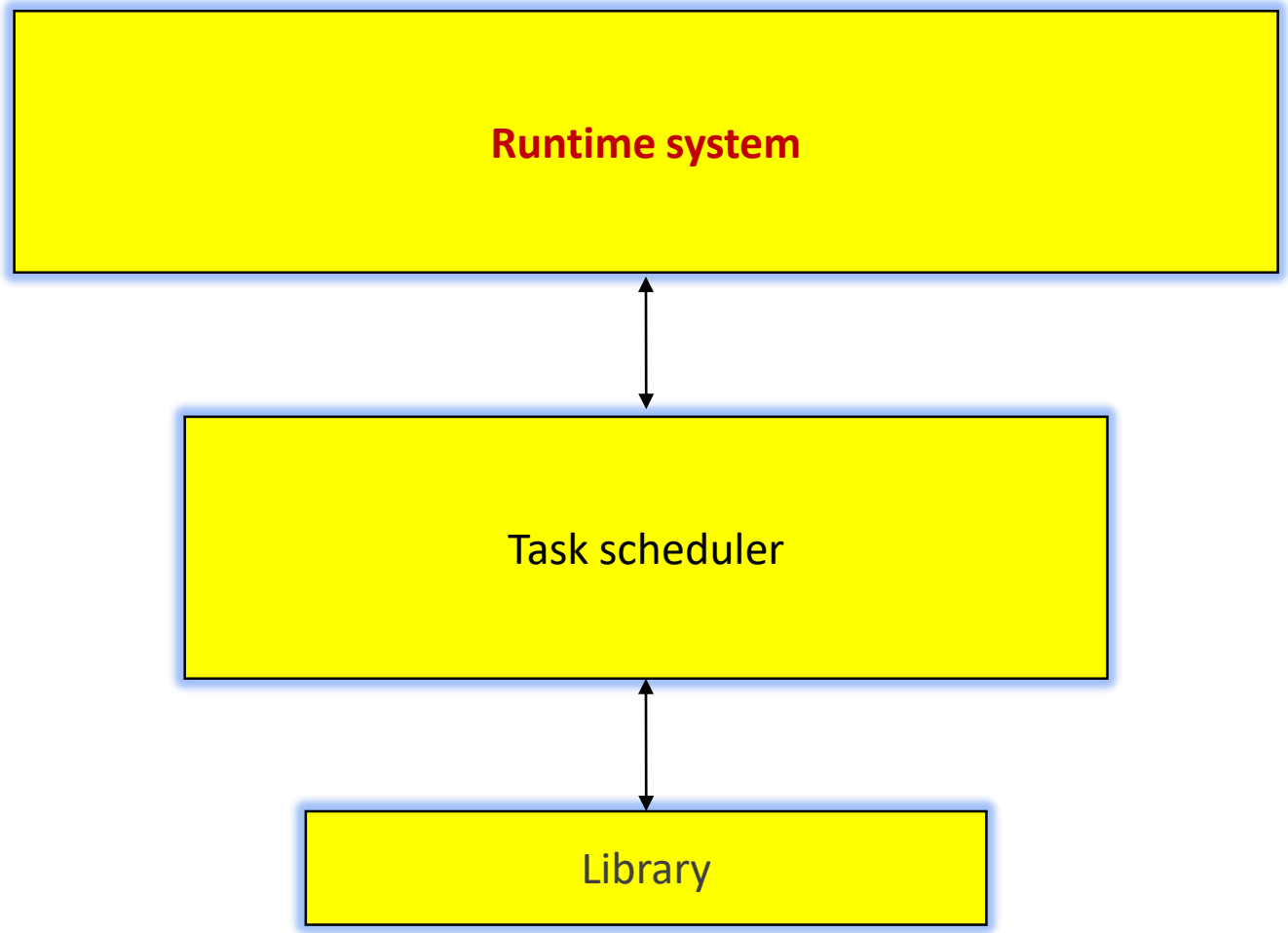


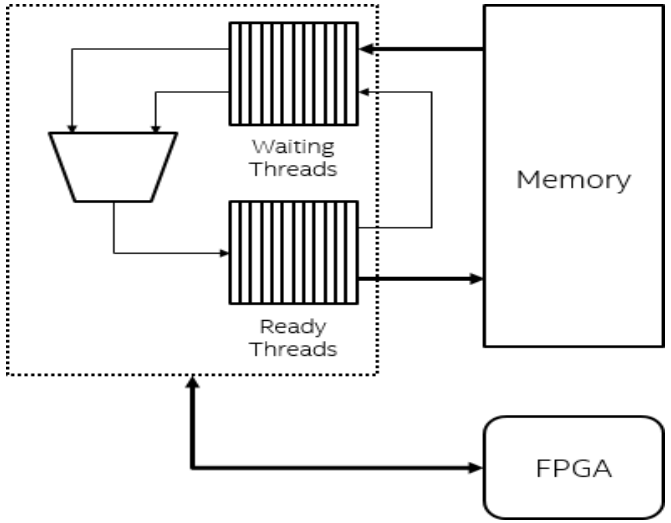


FPGA-to-Host Memory Communication

- JVM and native memory data transfers can incur significant overhead
  - Leverage non-blocking I/O; access data in streaming fashion
  - No garbage collection
  - Buffer re-use among groups of SW threads
  - Hde data transfer latency by overlapping comm. with comp.

3. JVM runtime system



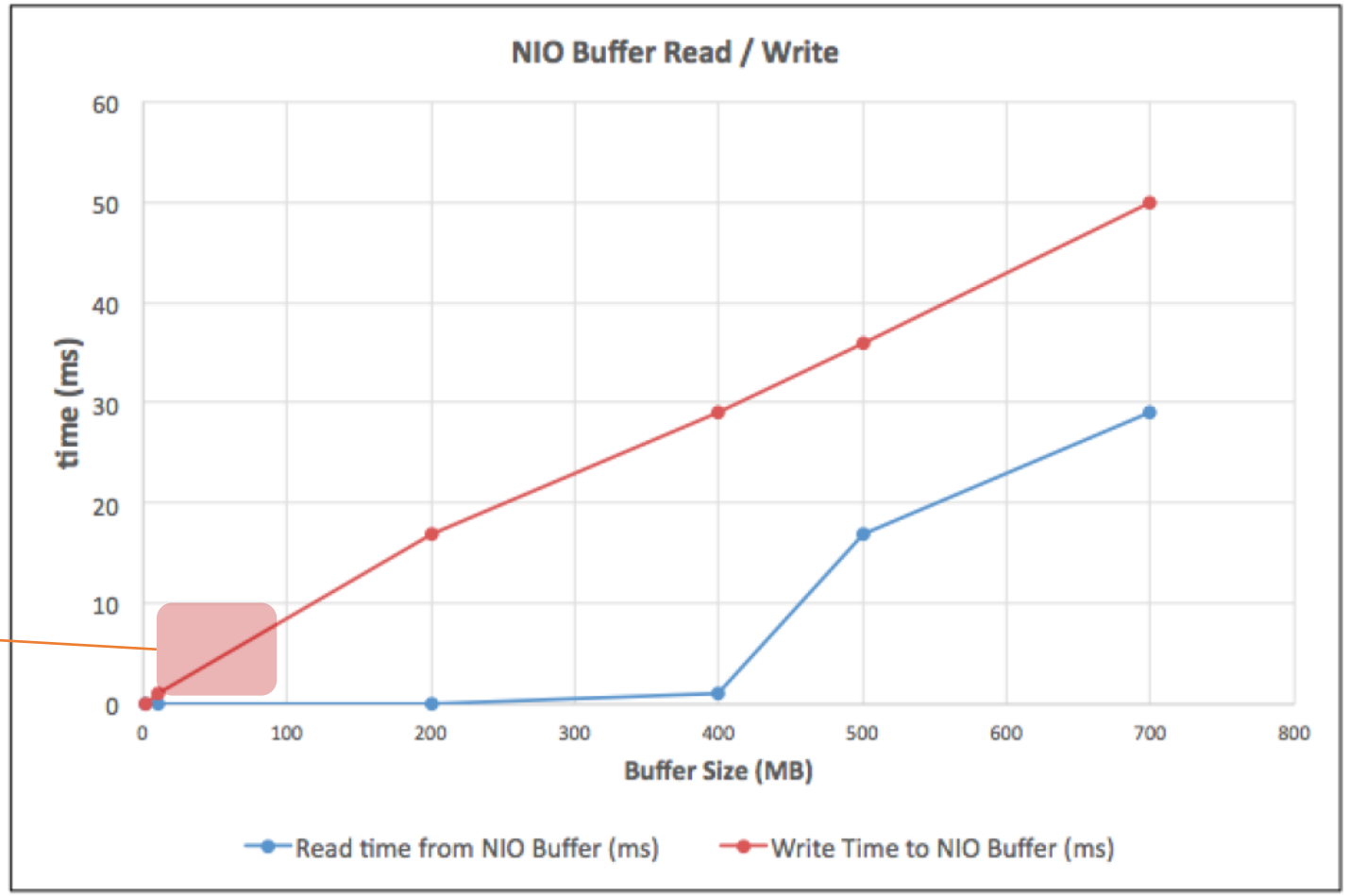


CPU-FPGA Thread Co-existence

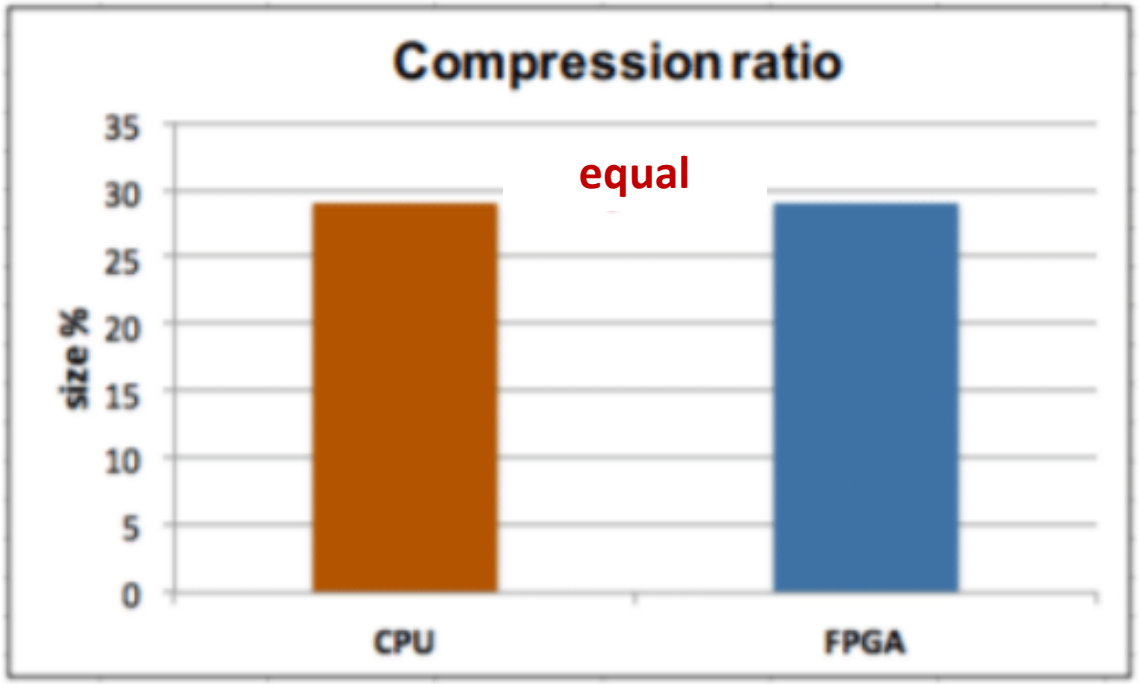
- Keep FPGA busy all time with tasks
- Stateless
- Accelerator management, (buffer size, int./clean-up, etc.)

Buffer R/W performance

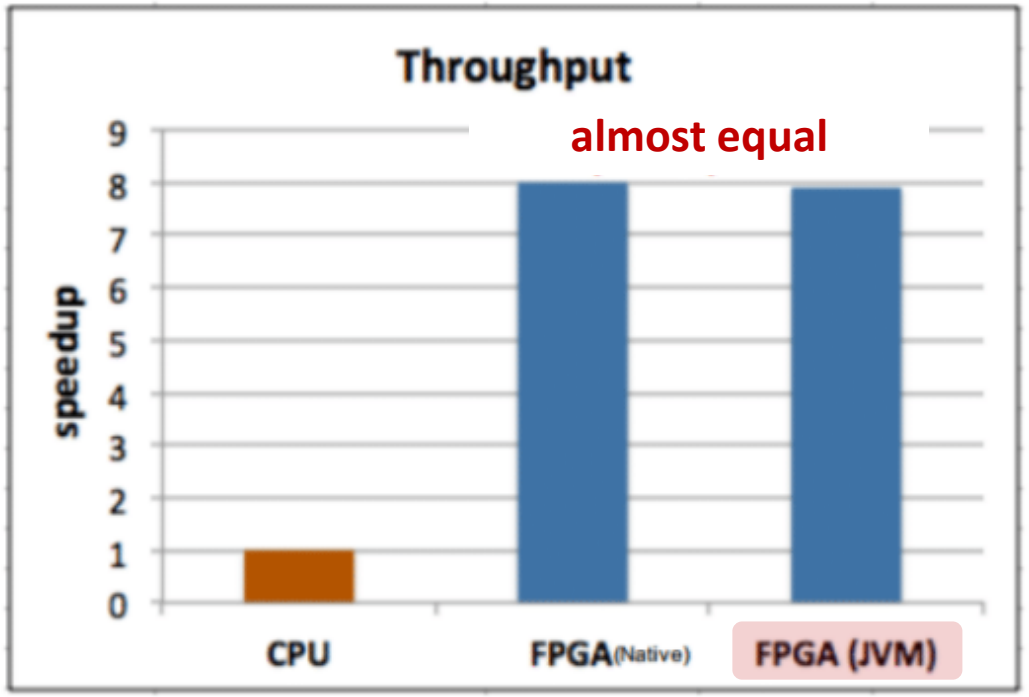
Sweet spot – resulting in almost no performance loss (next slide) for the runtime system.



Compression: Raw Performance



Functional Evaluation



Performance Evaluation

Conclusions & Future Work

- FPGAs compliment CPUs in certain compute-intensive task
  - 3.2X speedup, 4X memory footprint reduction in Spark (single-node deployment!)
  - Potentially increased benefit in larger-scale scenarios
- FPGA accelerators can leverage microservices architecture
  - Key to efficiency is CPU-memory-FPGA interaction; accelerator management
  - Optimizations to both low-level and high-level stack help improve performance
- Future work:
  - Performance evaluation with cloud and machine learning workloads