F3: An FPGA-accelerated FaaS Framework

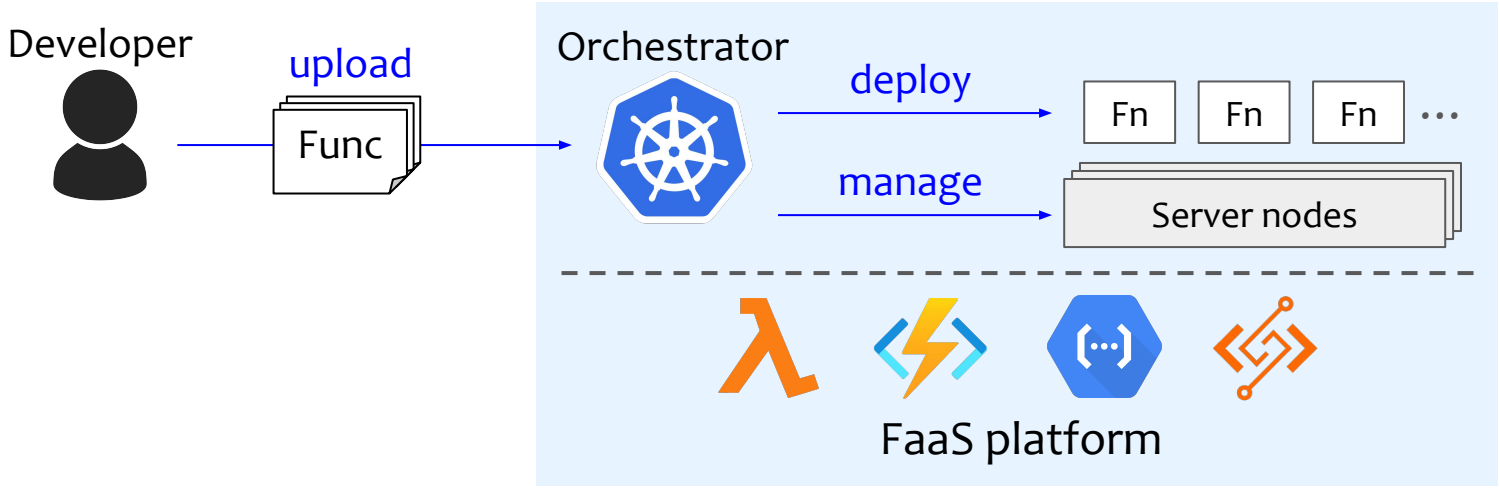
Charalampos Mainas, Martin Lambeck, Bruno Scheufler,
Laurent Bindschaedler, **Atsushi Koshiba**, Pramod Bhatotia

MAX PLANCK INSTITUTE
FOR SOFTWARE SYSTEMS

HPDC 2025

Function-as-a-Service (FaaS)

Function-as-a-Service (FaaS) simplifies the cloud computing model as ***serverless***

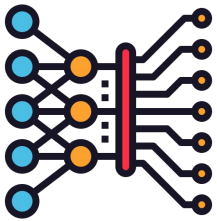


FaaS achieves easy deployment, high scalability, and low cost (pay as you go)

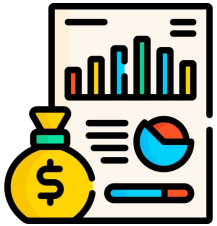
Performance limitations of FaaS platforms

Compute-intensive tasks are being deployed on FaaS platforms

Compute-intensive workloads



Machine learning



Data analytics

LLM

CPU-centric FaaS platforms

Amazon

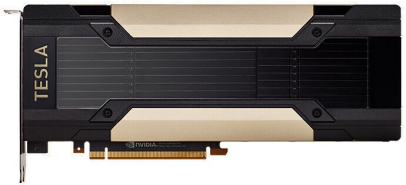
Microsoft

Google

Running compute-heavy workloads **only on CPUs** are not cost-/power-efficient

Accelerators in the cloud

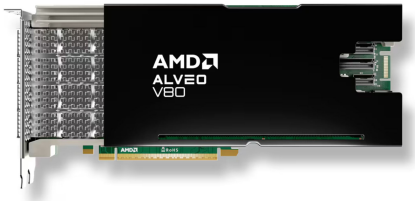
Various types of accelerators are being adopted in the cloud



GPUs

Tensor Processing Units (TPUs)

Our target



Field Programmable Gate Arrays (FPGAs)

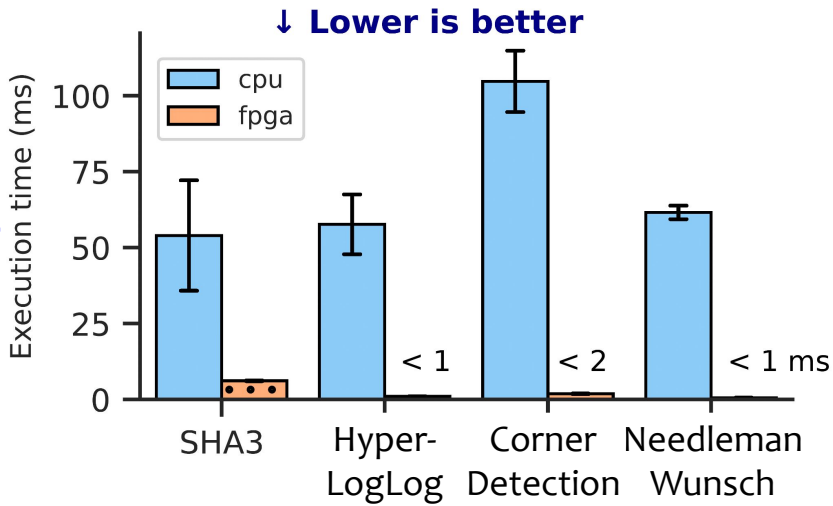
Can we leverage **FPGAs** to accelerate serverless functions?

FPGAs for cloud workloads

FPGAs are a good fit to accelerate various, ever-changing cloud workloads

- Reconfigurability
- Optimizable for specific computations
- Potential performance benefits

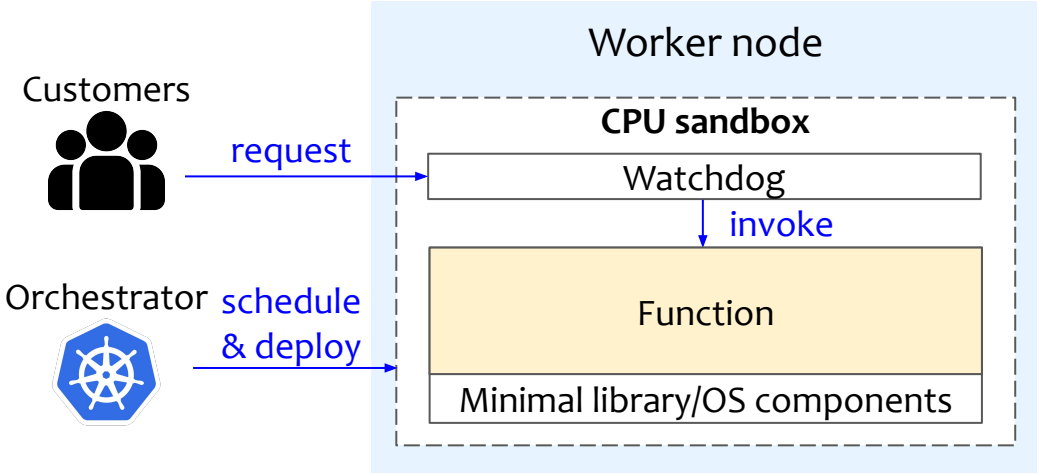
10-100x speedups



FPGA promises accelerating cloud workloads, ***but…***

FPGA and serverless model mismatches

There are ***hard-to-reconcile gaps*** between FPGA and serverless computing model

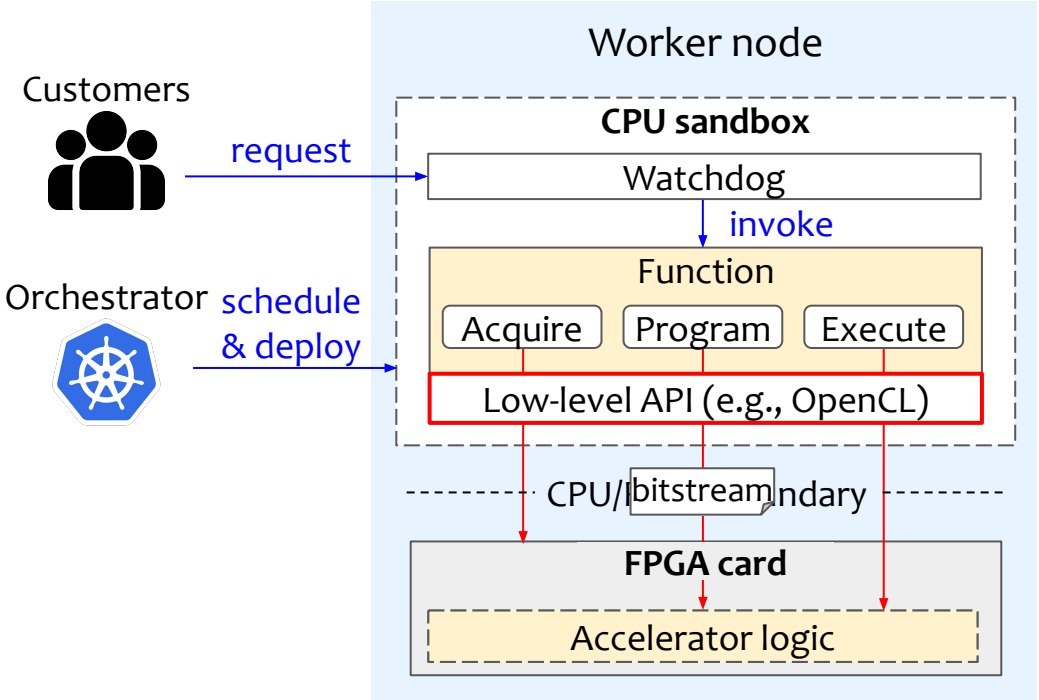


FPGA adoption may lose serverless benefits:

- #1 Easy-to-develop
- #2 CPU isolation
- #3 Low invocation latency
- #4 CPU orchestration

FPGA and serverless model mismatches

There are ***hard-to-reconcile gaps*** between FPGA and serverless computing model



FPGA adoption may lose serverless benefits:

- #1 Development complexity
- #2 Lack of isolation
- #3 High invocation latency
- #4 No orchestration for FPGAs

Research question

How do we enable **FPGA acceleration for serverless functions** by reconciling mismatches between the serverless computing model and FPGA execution model?

Our proposal

F3: FPGA-accelerated FaaS Framework
The first end-to-end FaaS framework for FPGA acceleration and orchestration

Properties:

- Programmability
  - High-level APIs for FPGA control
- Multi-tenancy
  - Hypervisor-enforced isolation
- Low latency
  - Fast FPGA reconfiguration & CPU-FPGA communication
- Resource efficiency
  - FPGA-aware orchestration

Outline

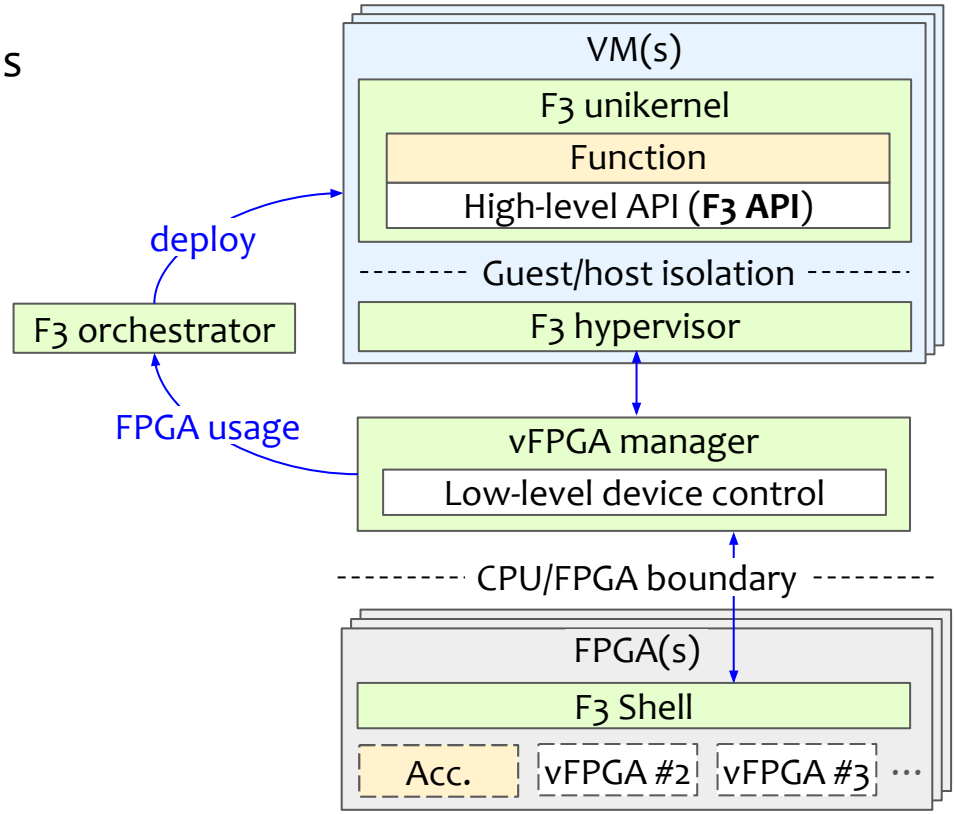
- ~~Motivation~~
- Overview
- Evaluation

F3 overview

An end-to-end HW/SW co-design bridges a gap between FPGA and serverless

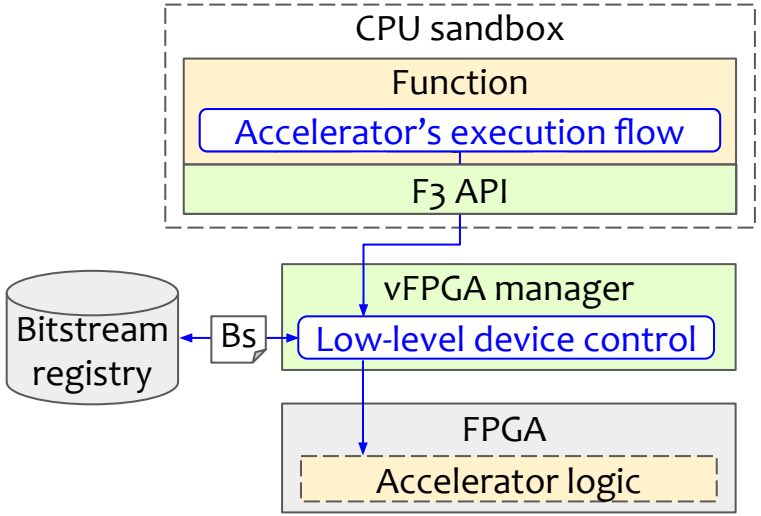
Key components:

#1 F3 API
#2 F3 unikernel & hypervisor
#3 F3 Shell & vFPGA manager
#4 F3 orchestrator

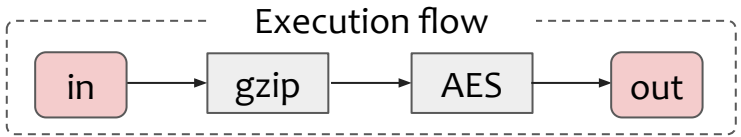


#1: F3 API

Accelerator-agnostic API that delegates low-level jobs to the backend




```
in   = f3_alloc_buffer(INPUT_SIZE);
tmp  = f3_alloc_buffer(INPUT_SIZE);
out  = f3_alloc_buffer(OUTPUT_SIZE);
id1  = f3_call_fpga('gzip', in, tmp);
id2  = f3_call_fpga('aes', tmp, out);
f3_wait_fpga(id1, id2);
```

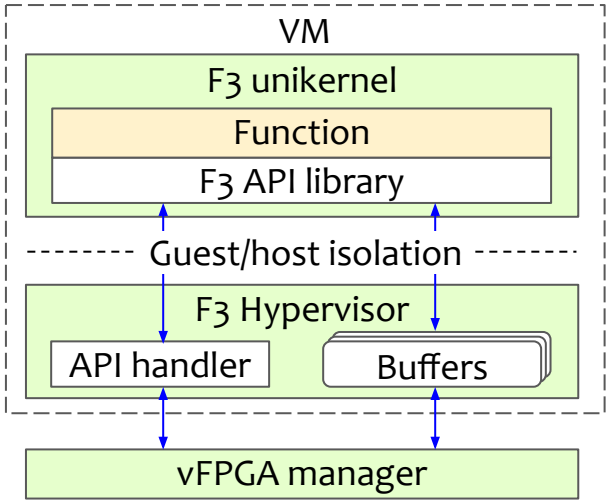


F3 API abstracts the complex FPGA execution flow from developers

#2: F3 unikernel and hypervisor

A lightweight isolation mechanism for latency-sensitive serverless functions

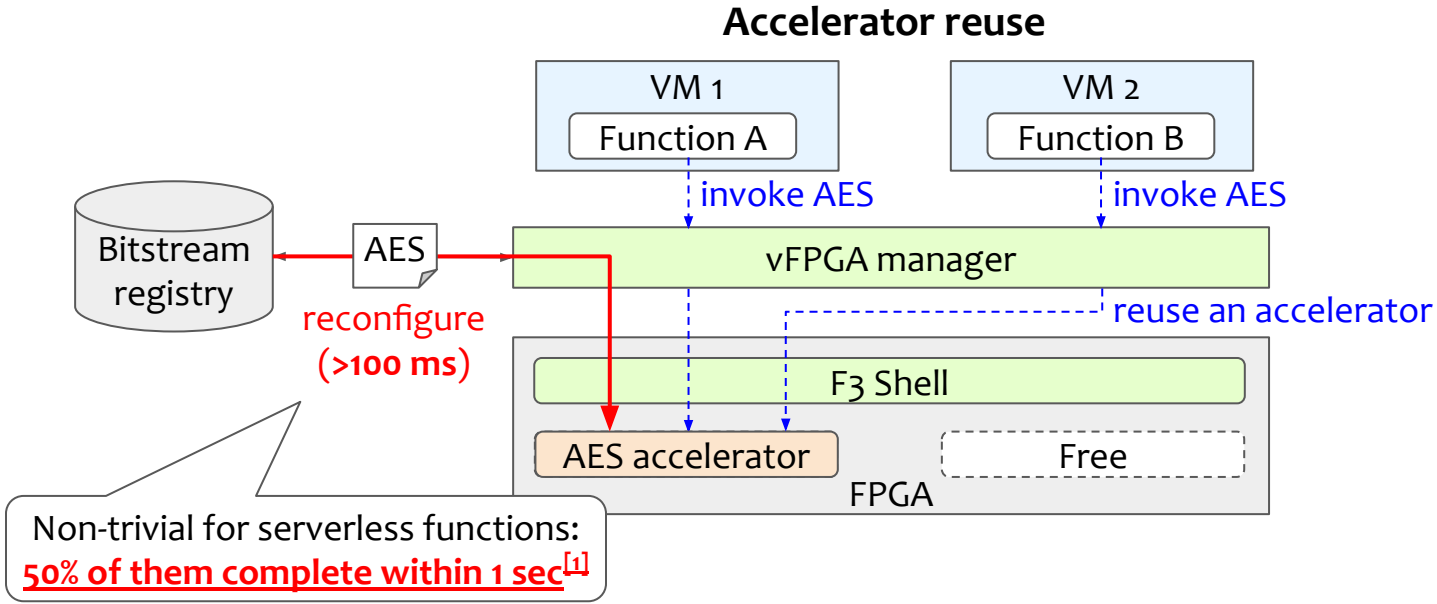
- F3 unikernel
  - ensures isolation for CPU contexts
- F3 hypervisor
  - bridges guest functions and FPGAs
- vFPGA manager
  - restricts access to FPGA resources



F3 guarantees multi-tenant isolation while retaining the lightweightness

#3: F3 Shell and vFPGA manager

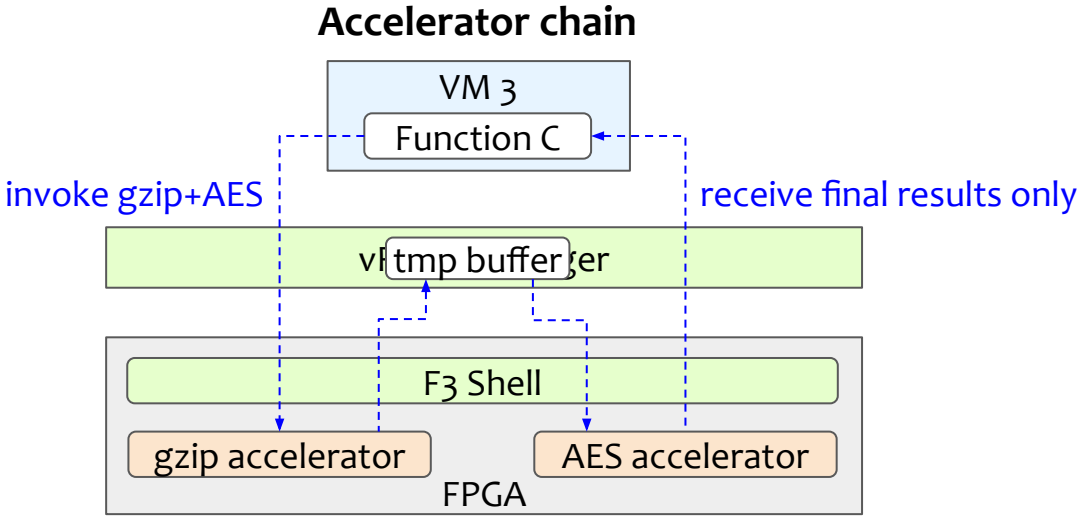
Low-level device management ***mitigating FPGA-related overheads***



[1] "Serverless in the Wild: Characterizing and Optimizing the Serverless Workload at a Large Cloud Provider", USENIX ATC'20

#3: F3 Shell and vFPGA manager

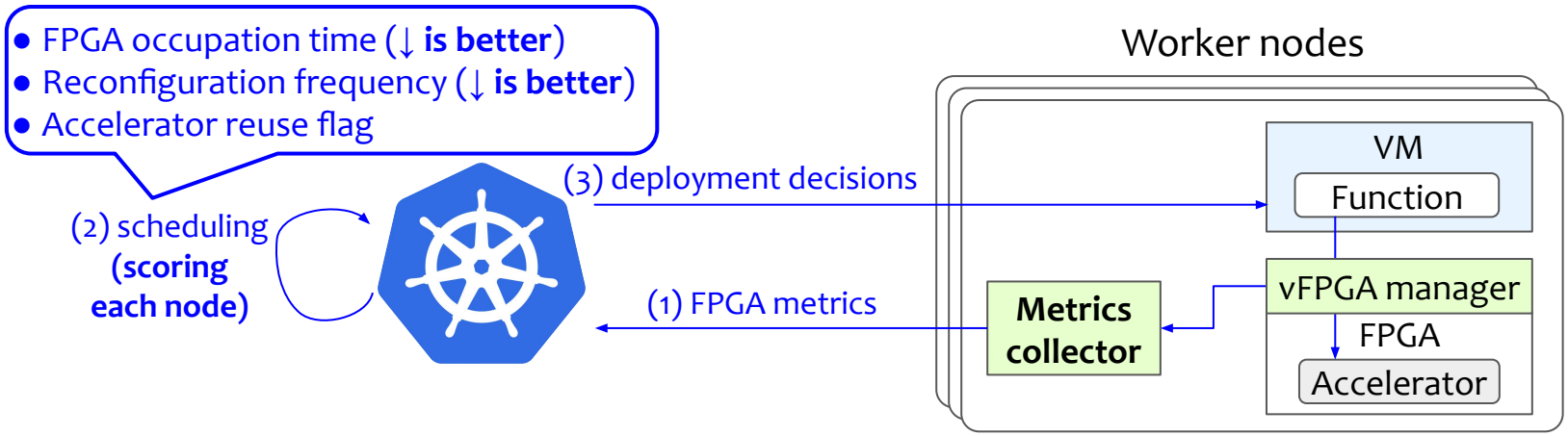
Low-level device management ***mitigating FPGA-related overheads***



F3 leverages Shell features to reduce FPGA's start-up/communication latencies

#4: FPGA-aware orchestrator

Collecting FPGA-related metrics for ***FPGA-centric orchestration***



F3 orchestrator achieves fair, resource-efficient function scheduling

Outline

- ~~Motivation~~
- ~~Overview~~
- Evaluation

Evaluation

Questions:

- What is the end-to-end performance gain of F3?
- How effective is FPGA function (accelerator) chaining?

See our paper for more results!

Experimental setup:

- F3 prototype built upon OpenFaaS, Kubernetes, containerd, and Kata-runc
- Cluster : 1x leader (Xeon G5317@3.0GHz), 3x workers (Xeon G6238R@2.2GHz)
- FPGA : each worker node equipped with 1x AMD FPGA (Alveo U50)

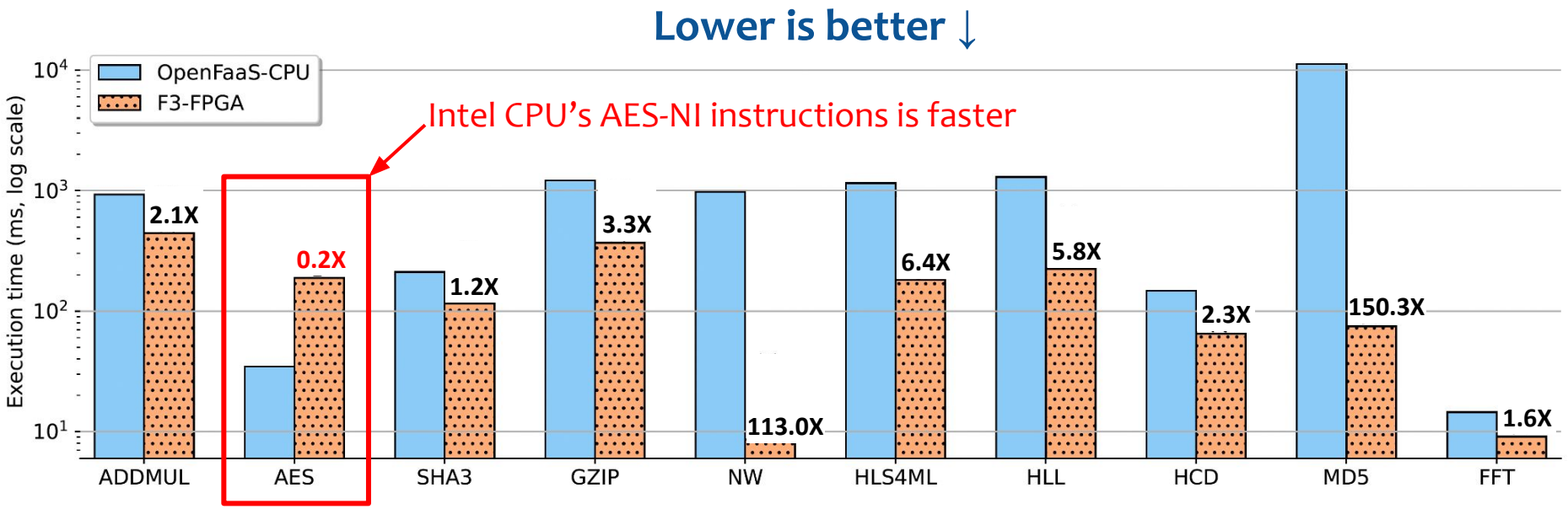
Evaluation

Applications:

- 10 compute-intensive workloads suitable for FPGA acceleration
  - Encryption (AES-128-ECB)
  - Hashing (SHA3-512, MD5)
  - Compression (GZIP)
  - Convolutional neural network (HLS4ML[2])
  - Data analytics (HyperLogLog)
  - Etc.

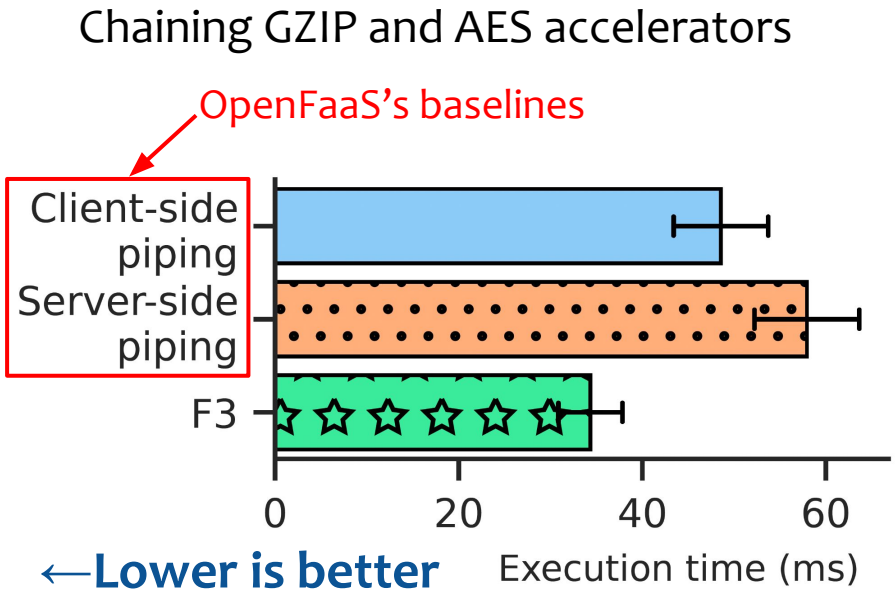
[2] HLS4ML: Ultra fast machine learning for FPGAs., https://opensource.web.cern.ch/HLS4ML

End-to-end performance



F3 achieves **28.6x average speedups (up to 150.3x)** over a CPU-only baseline

FPGA accelerator chaining



F3 function chaining achieves **1.4x to 1.7x speedups** against OpenFaaS baselines

Summary

Hard-to-reconcile differences between serverless and FPGA execution models

- ***Development complexity*** due to low-level device management
- ***Lack of isolation*** for user code/data distributed across CPU & FPGA
- ***High invocation latency*** due to FPGA reconfiguration & communication overheads
- ***No orchestration support*** for FPGAs

F3: FPGA-accelerated FaaS Framework

- ***High-level API*** for easy FPGA execution
- ***Multi-tenant isolation*** for CPU & FPGA contexts
- ***Low invocation latency*** assisted by the FPGA Shell
- ***FPGA-aware orchestration***

Paper

Code

https://github.com/TUM-DSE/F3