Serverless FPGA

Utilizing dynamic partial reconfiguration in
an FPGA-accelerated FaaS architecture

Martin Lambeck
Advisor: Dr. Atsushi Koshiba
Chair of Computer Systems
https://dse.in.tum.de/

15.07.2023 – 15.07.2024
(part time)

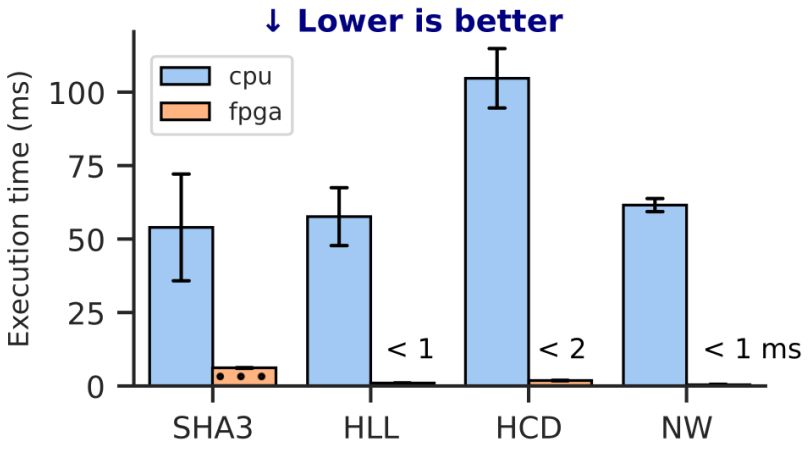
Outline

- Motivation
- Design
- Benchmark applications
- Evaluation
- Conclusion

Motivation

FPGAs

- Offer great computing performance
  - Excel at parallelizable/pipelinable tasks
- Difficult to program and integrate into system



Motivation

FPGAs

- Offer great computing performance
  - Excel at parallelizable/pipelinable tasks
- Difficult to program and integrate into system

Serverless functions (FaaS)

- Popular cloud deployment model
- Simplifies deployment greatly
- No infrastructure management by developer

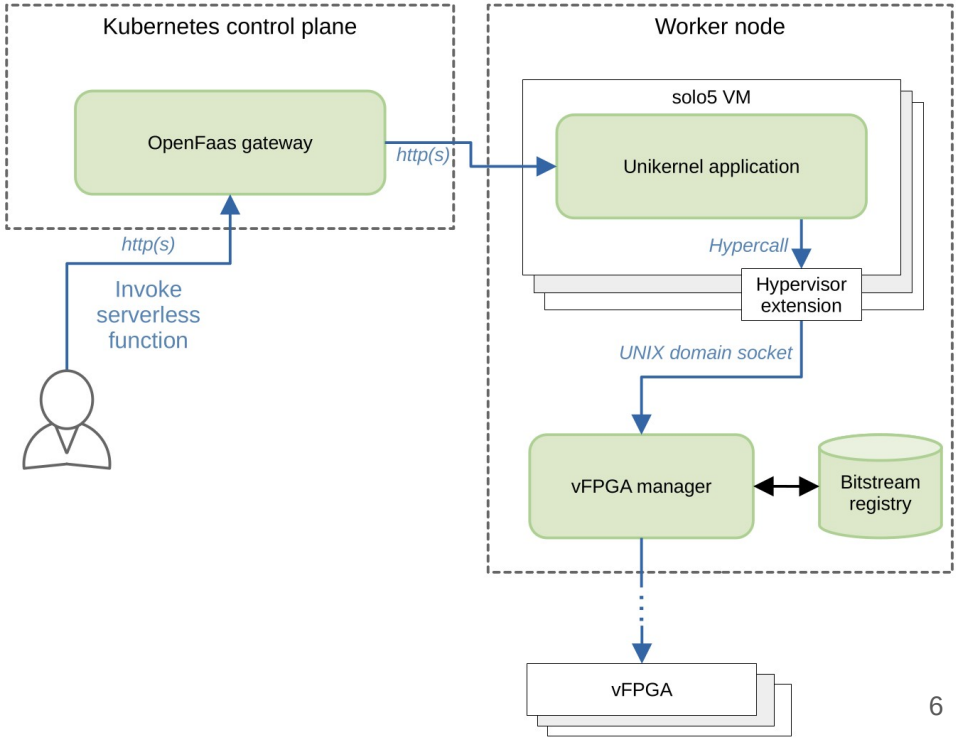
Can we combine them?

Outline

- ~~Motivation~~
- Design
- Benchmark applications
- Evaluation
- Conclusion

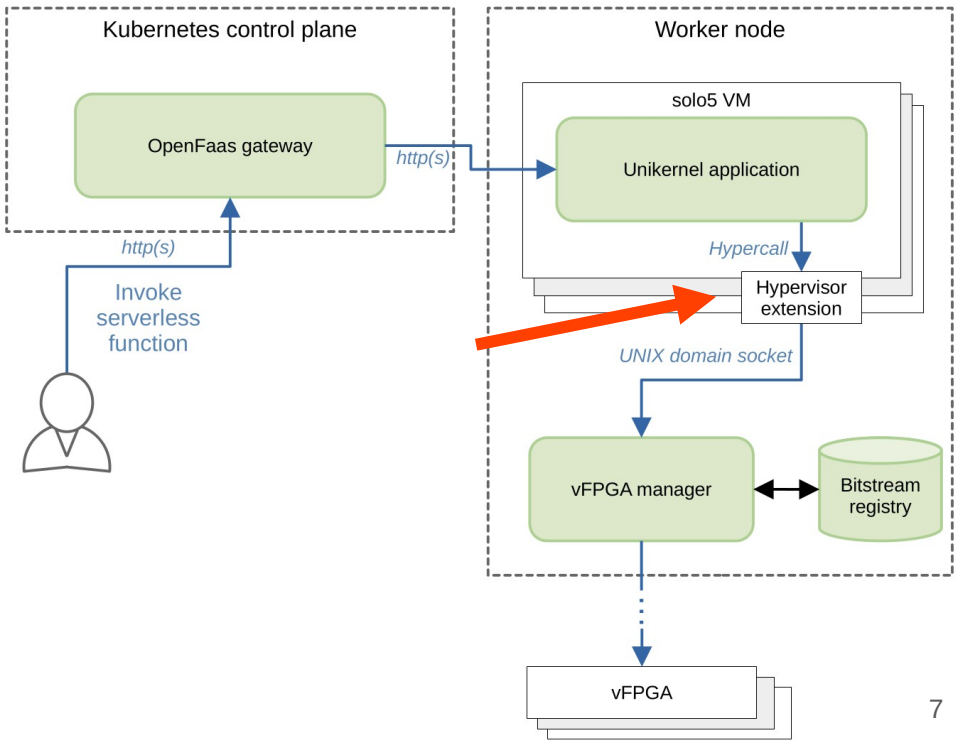
Design

- **FPGA:** Use Coyote shell as runtime
  - Multiple isolated slots (vFPGAs)
  - Multi tenancy
- **Host:** A minimal program on the host side is necessary
  - Relays input/output data between FPGA and invoker
  - Unikernel, confined to VM
- **Cluster:** OpenFaaS/Kubernetes for orchestration



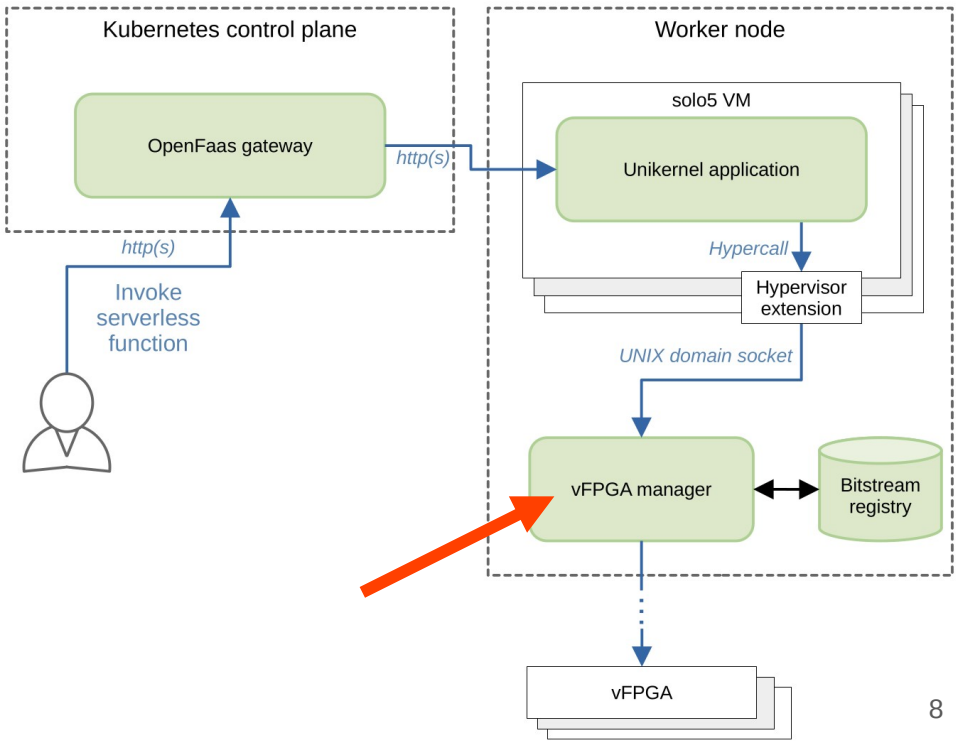
Design

- Hypercalls to exit VM
-
```
solo5_result_t solo5_serverless_set_csr(uint32_t offset, uint64_t value);
solo5_result_t solo5_serverless_load_bitstream(uint32_t config);
solo5_result_t solo5_serverless_map_memory(
      void *addr,
      size_t len,
      size_t input_len,
      size_t output_len);
solo5_result_t solo5_serverless_exec(void);
```
- Communicate with vFPGA manager



Design

- Client/server architecture
- Receives invocation request from unikernel app
- Schedules invocations
- Manages reconfiguration
- Invoke user logic on FPGA
- Return completion signal



Outline

- ~~Motivation~~
- ~~Design~~
- Benchmark applications
- Evaluation
- Conclusion

Benchmarks

- Select popular algorithms
  - Port/implement FPGA implementation to Coyote
  - Baseline: CPU implementation
- 12 benchmark applications:

AddMul	AES (ECB mode)
SHA256	SHA3
GZIP	MATMUL (64x64)
Needleman-Wunsch	hls4ml
Hyperloglog	Harris Corner Detection
MD5 brute force	FFT Auto-Correlation

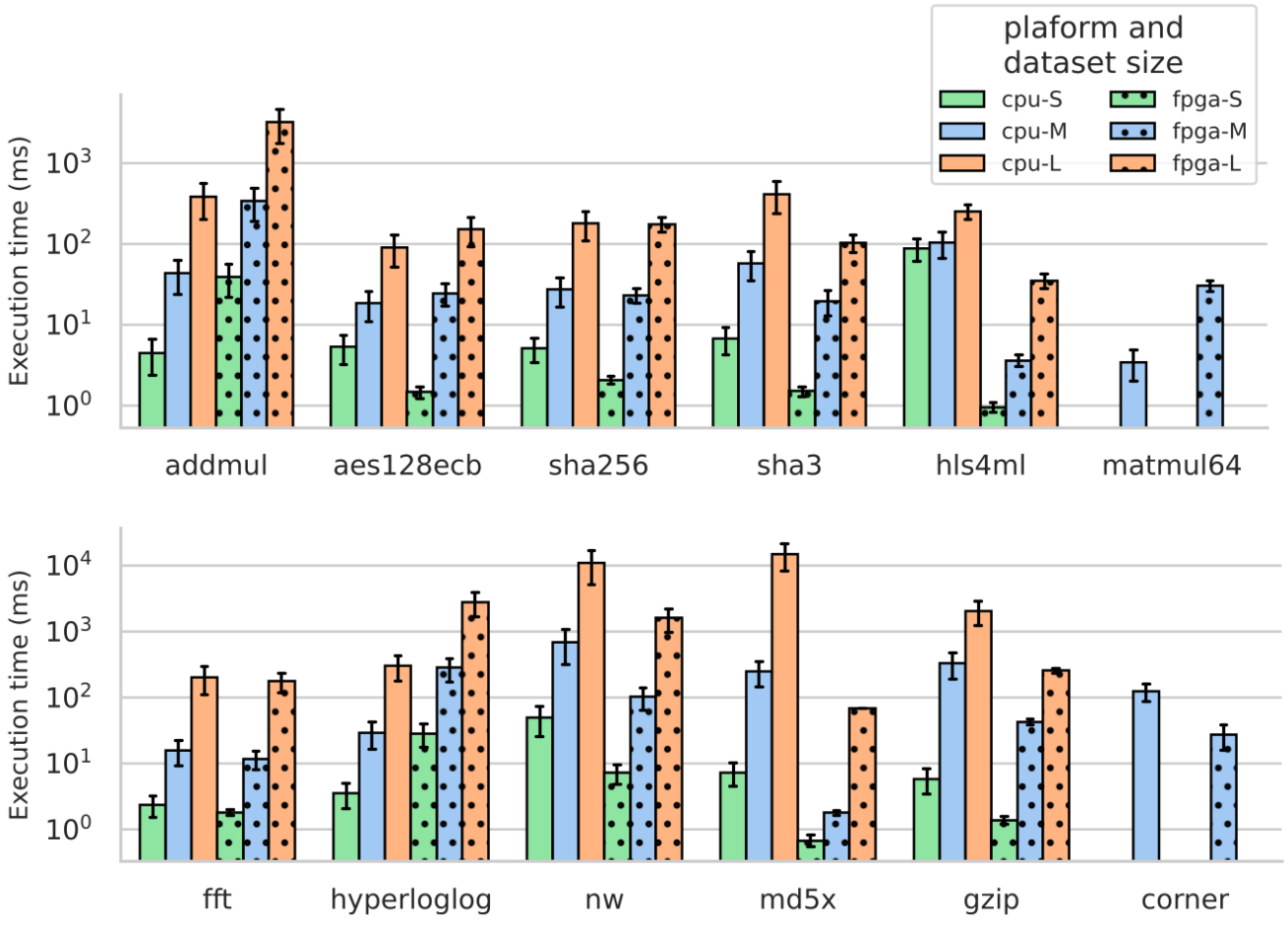
Outline

- ~~Motivation~~
- ~~Design~~
- ~~Benchmark applications~~
- Evaluation
- Conclusion

Evaluation

- End-to-end evaluation
  - Evaluate all benchmarks
  - Measure duration from moment input data has been received until result is ready
  - No network overhead!
- Micro-benchmarks
  - Measure impact of huge pages
  - Measure parallel efficiency
  - Measure reconfiguration overhead

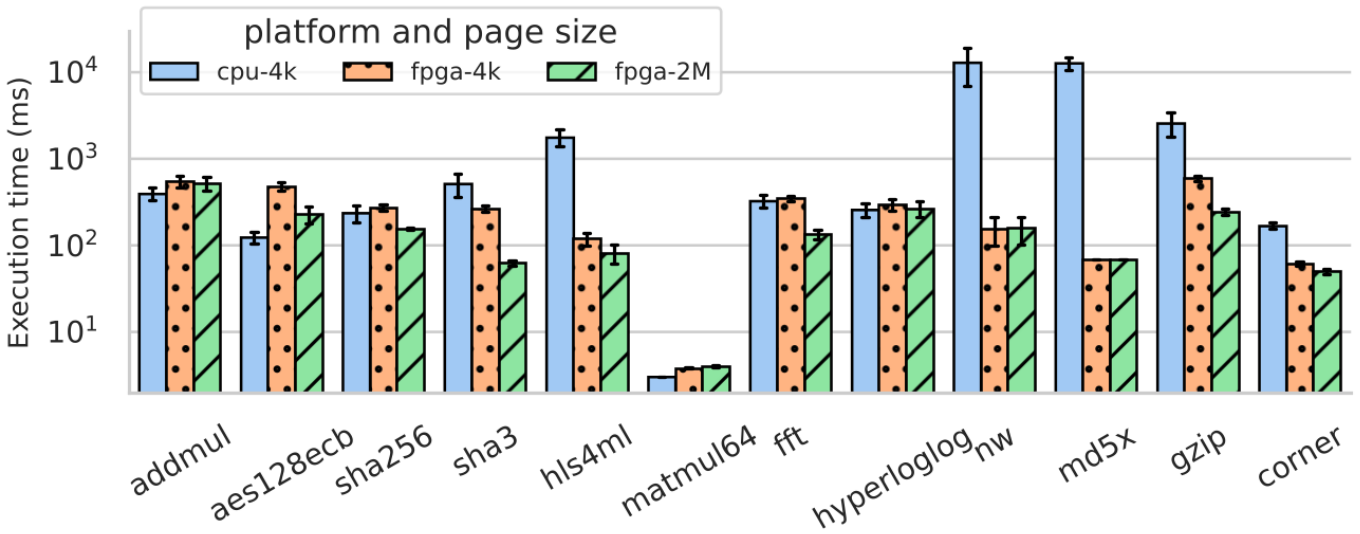
Results (End-to-end)



FPGA speed-up factors

	S	M	L
addmul	0.12	0.13	0.12
aes128	3.7	0.76	0.6
sha256	2.4	1.2	1.0
sha3	4.5	2.9	4.0
hls4ml	91	29	7.2
matmul64		0.11	
fft	1.3	1.4	1.1
hll	0.12	0.10	0.11
nw	6.9	6.7	6.9
md5x	11	140	220
gzip	4.2	7.8	8.0
corner		4.6	

Results (Huge pages)

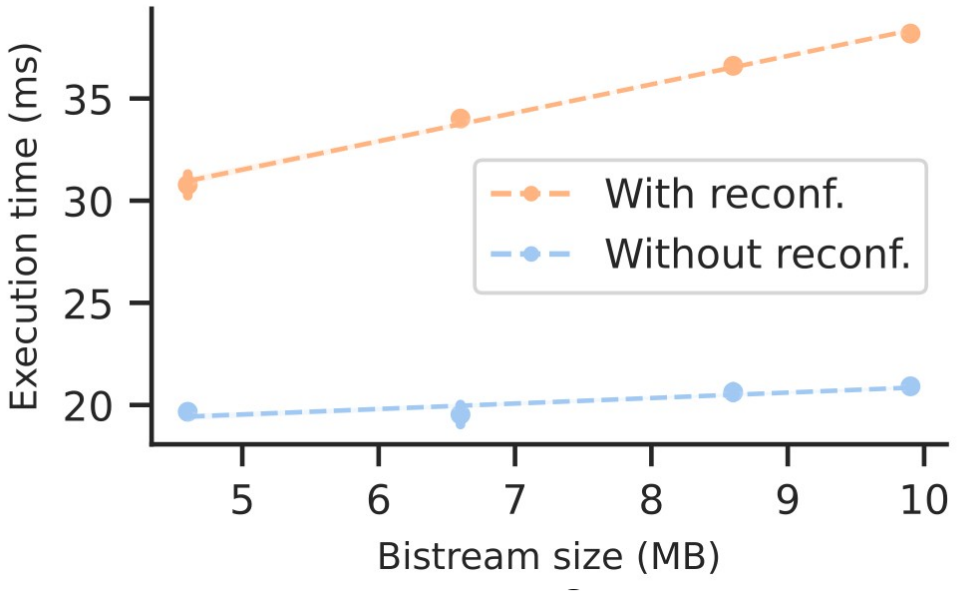


FPGA + huge pages
speed-up factors

Baseline:	**cpu-4k**	**fpga-4k**
addmul	0.76	1.1
aes128	0.54	2.1
sha256	1.5	1.7
sha3	8.3	4.2
hls4ml	22	1.5
matmul64	0.76	0.95
fft	2.4	2.6
hll	0.97	1.1
nw	82	0.98
md5x	180	1.0
gzip	11	2.4
corner	3.4	1.2

Results (Reconfiguration)

- Dynamic partial reconfiguration induces overhead
- Linear correlation between bitstream size and duration



Clock regions	8	16	24	32
Bitstream size	4.6 MB	6.6 MB	8.6 MB	9.9 MB

Outline

- ~~Motivation~~
- ~~Design~~
- ~~Benchmark applications~~
- ~~Evaluation~~
- Conclusion

Conclusion

- FPGAs offer excellent performance for some workflows
- Coyote shell enables multi-tenancy on the FPGA
- FaaS simplifies infrastructure management

- => FaaS + FPGA is a feasible deployment model

Thank you!
Questions?