Faster and Cheaper Serverless Computing on Harvested Resources

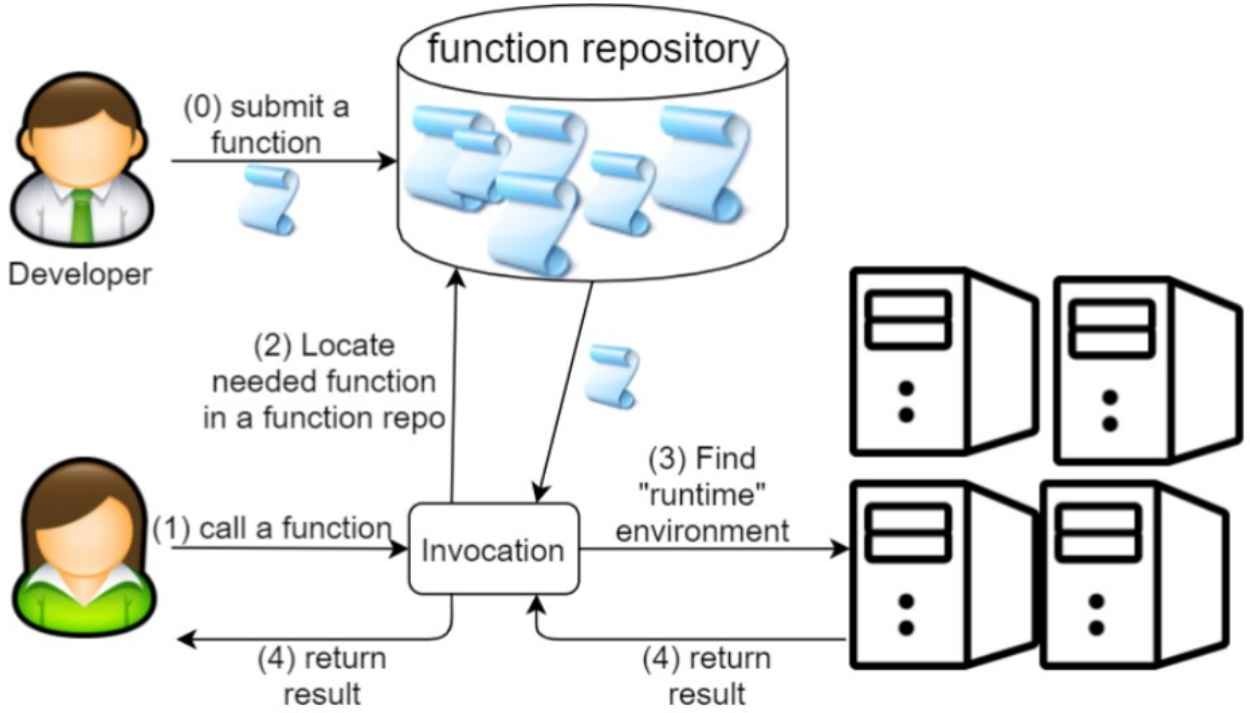
Yanqi Zhang et al.

SOSP '21: Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles October 2021 Pages 724–739

https://doi.org/10.1145/3477132.3483580

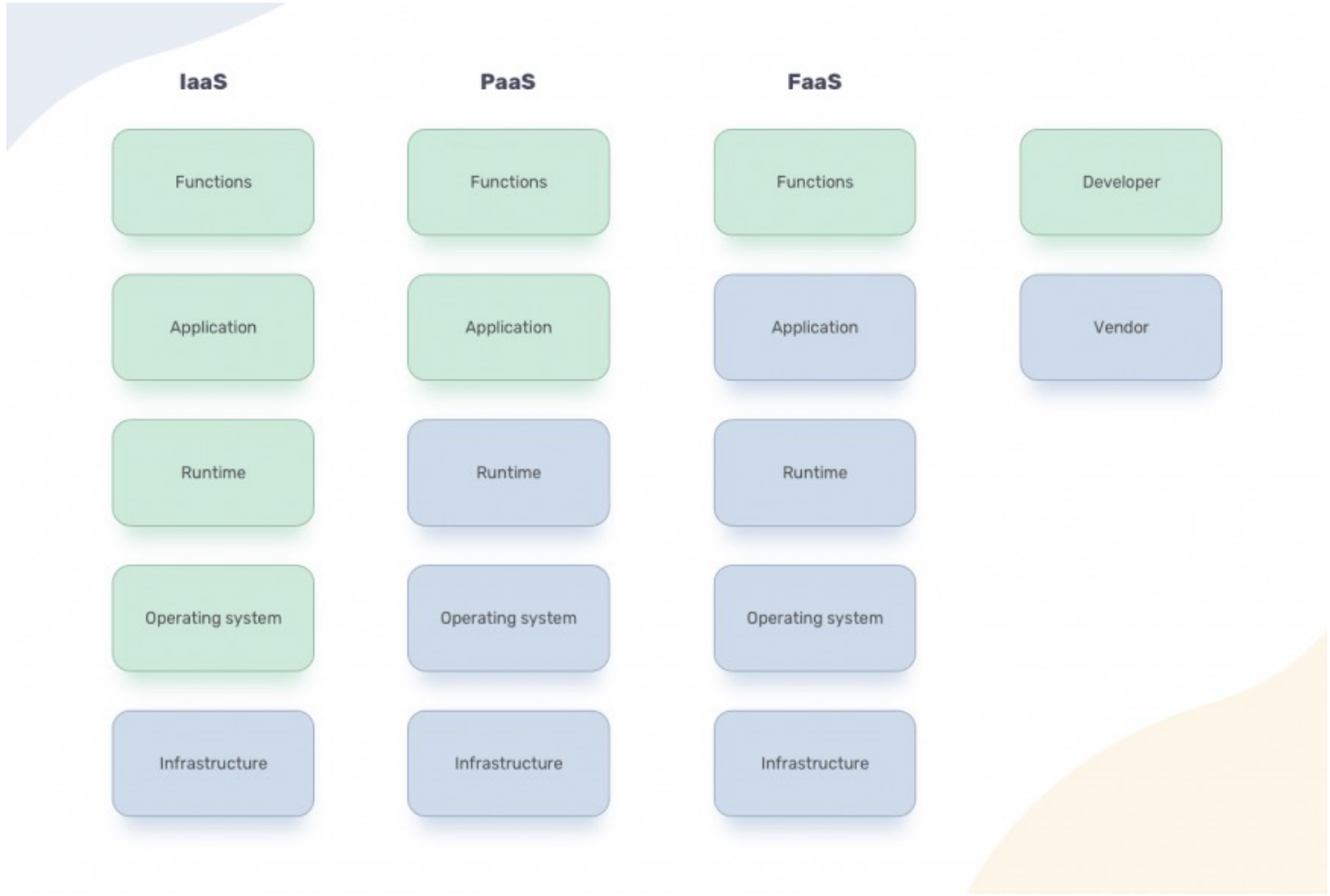
Presented by Bocheng(Noah) Cui
Master's in UNH
noahcui.github.io/info

Serverless Computing

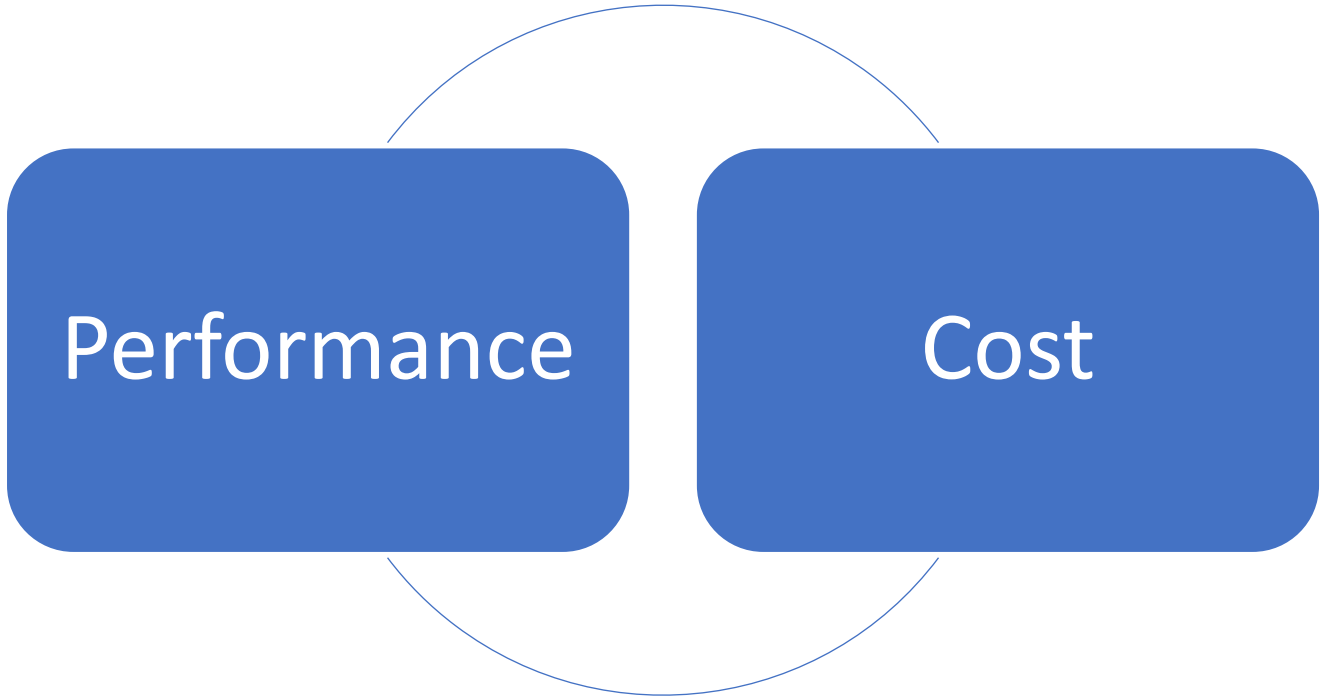


Serverless Computing
aka Function-as-a-Service(FaaS)

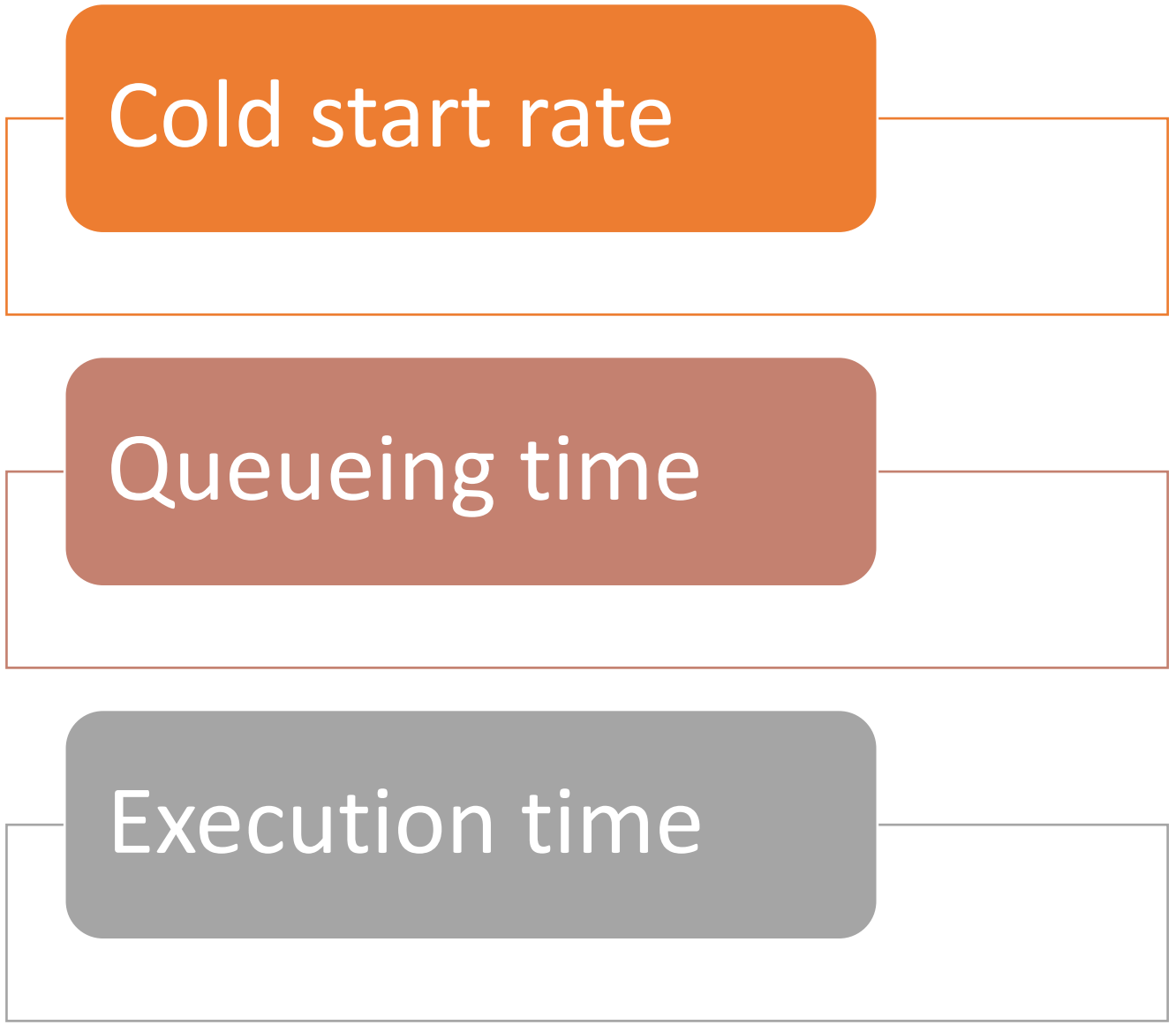
Blackbox for developer & user



Balance



Latency



Harvested Resources

- Can use idle resources

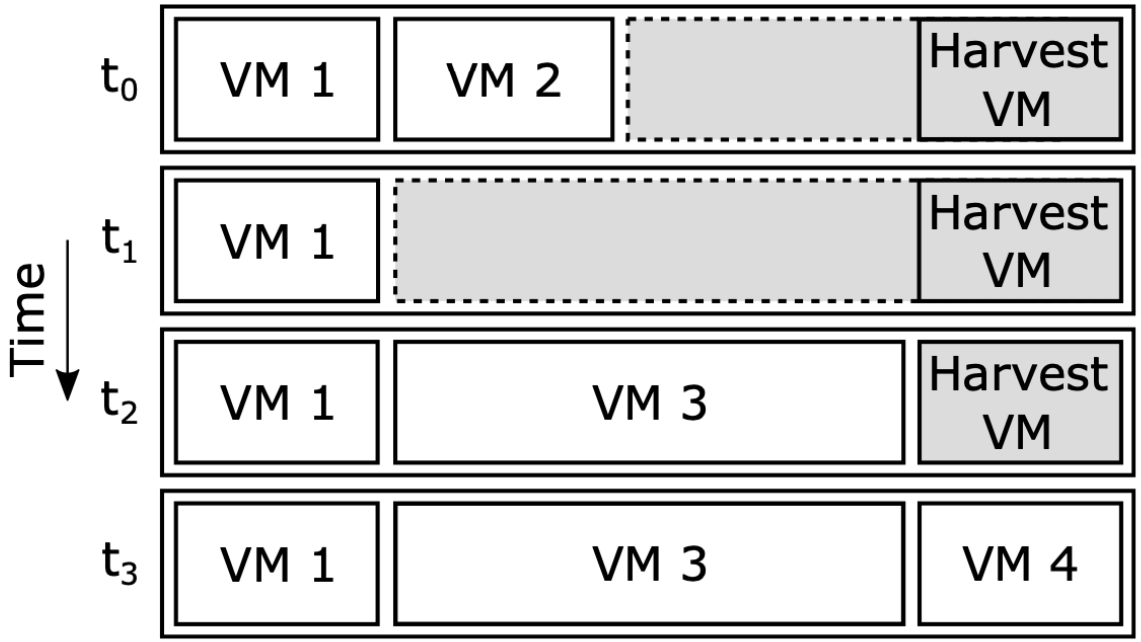


Figure 8: Harvest VM dynamically changing sizes over time.

Pic from: Providing SLOs for Resource-Harvesting VMs in Cloud Platforms

Serverless computing using harvest resources

Typically, functions don’t consume a lot of resources.

FaaS can easily adjust to harvest resources

Many of FaaS functions can be packed on one VM

An eviction may affect more computations.

Eviction vs Function run time

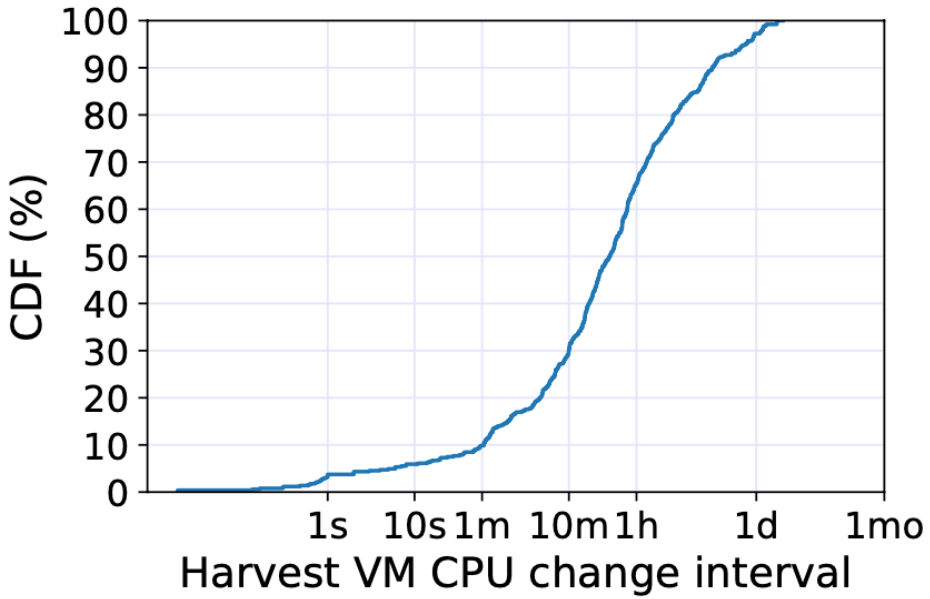


Figure 2: Intervals between Harvest VM CPU changes.

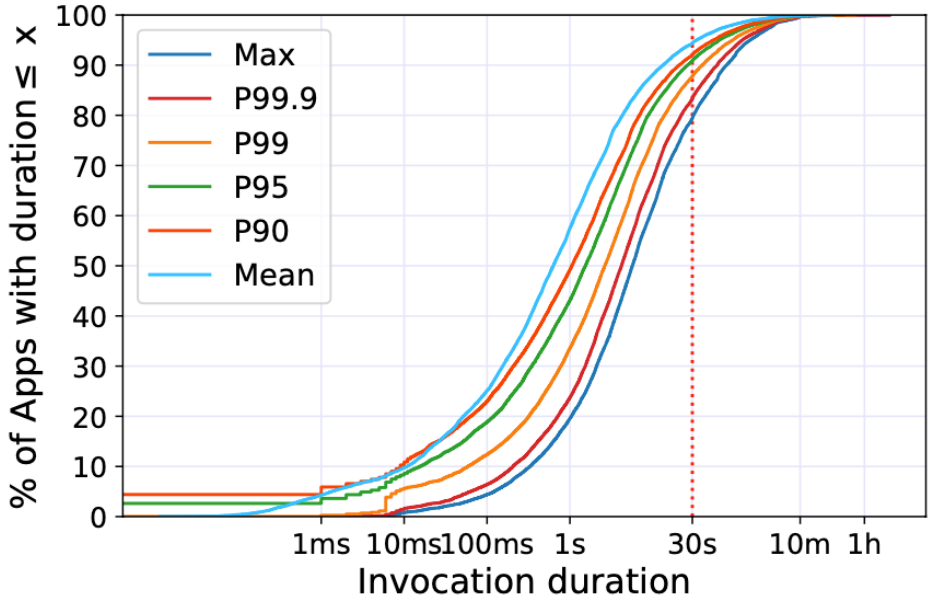


Figure 4: CDFs of the average and top percentiles of the invocation durations per application in the F_{Large} trace.

Handling Resource Variability

Join-the-Shortest-Queue(JSQ)

- Pretty good at reducing Queuing time for a heavy loaded system
- Can potentially increase cold start rate
- Need to track the min, O(log(n))

Min-Worker-Set(MWS)

- Can reduce cold start rate
- Reduce the queuing time, but in some cases may slightly imbalance the system
- Just blindly iterating, O(1)

Evaluation

- Multiple Python serverless functions from FunctionBench

Functions	Description
Floatop	Sine, cosine & square root
Matmult	Square matrix multiplication
Linpack	Linear equation solver
Chameleon	HTML table rendering
Pyaes	AES encryption & decryption
Image processing	Flip, rotate, resize, filter & grayscale images
Video processing	Grayscale video
Image classification	MobileNet inference
Text classification	Logistic regression

Table 2: The examined serverless functions from FunctionBench [32] and their description.

Evaluation

- On OpenWisk
- 10 invokers, 32CPUs and 128GB memory for each invoker
- CPU ranges from 5-25

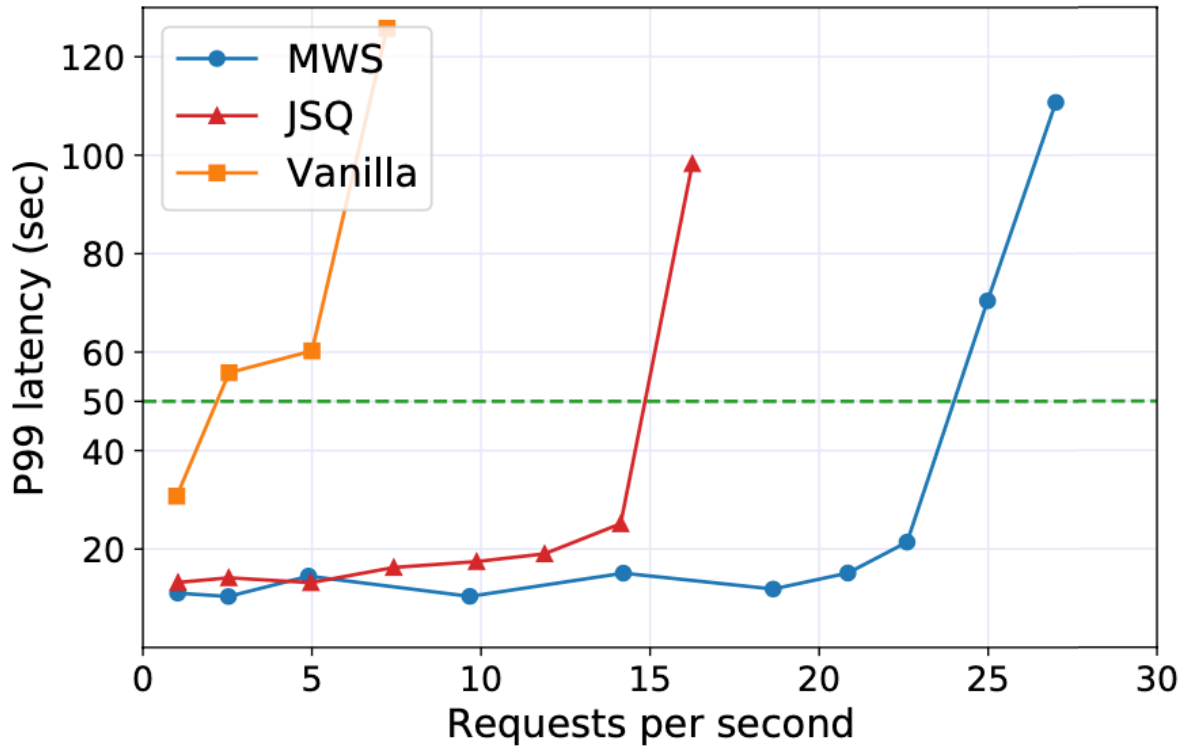


Figure 12: P99 latency across load balancing algorithms.

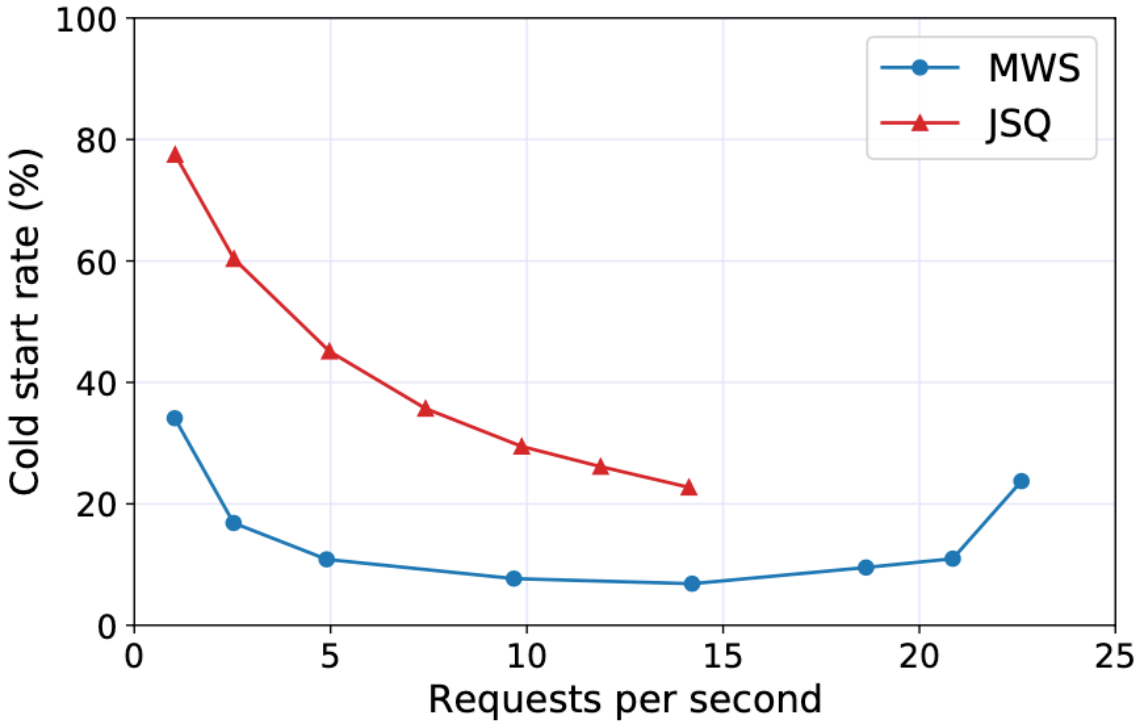


Figure 13: Cold start rate of MWS vs. JSQ.

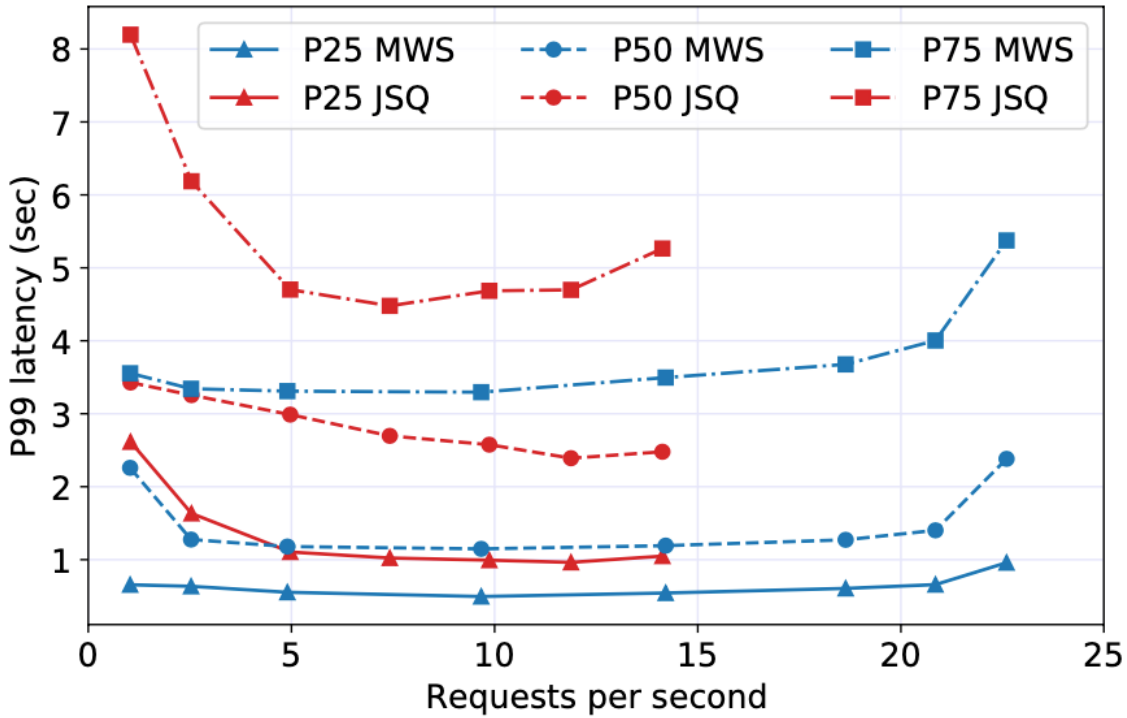


Figure 14: Low percentile latency of MWS vs. JSQ.

Actually, it is fixed # cores

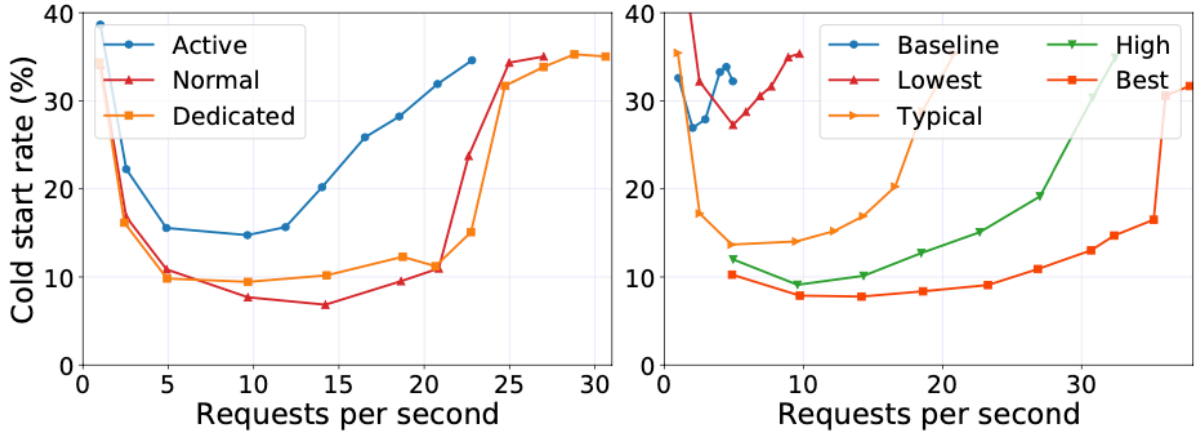


Figure 16: Cold start rate against load for fixed budget.

Discount	$d_{evict}(\%)$	$d_{harv}(\%)$	#VMs
Baseline (dedicated)	0	0	2
Lowest	48	48	6
Typical	70	80	12
High	80	90	18
Best	88	90	21

Table 3: Number of Harvest VMs with the same budget, based on the discount level.

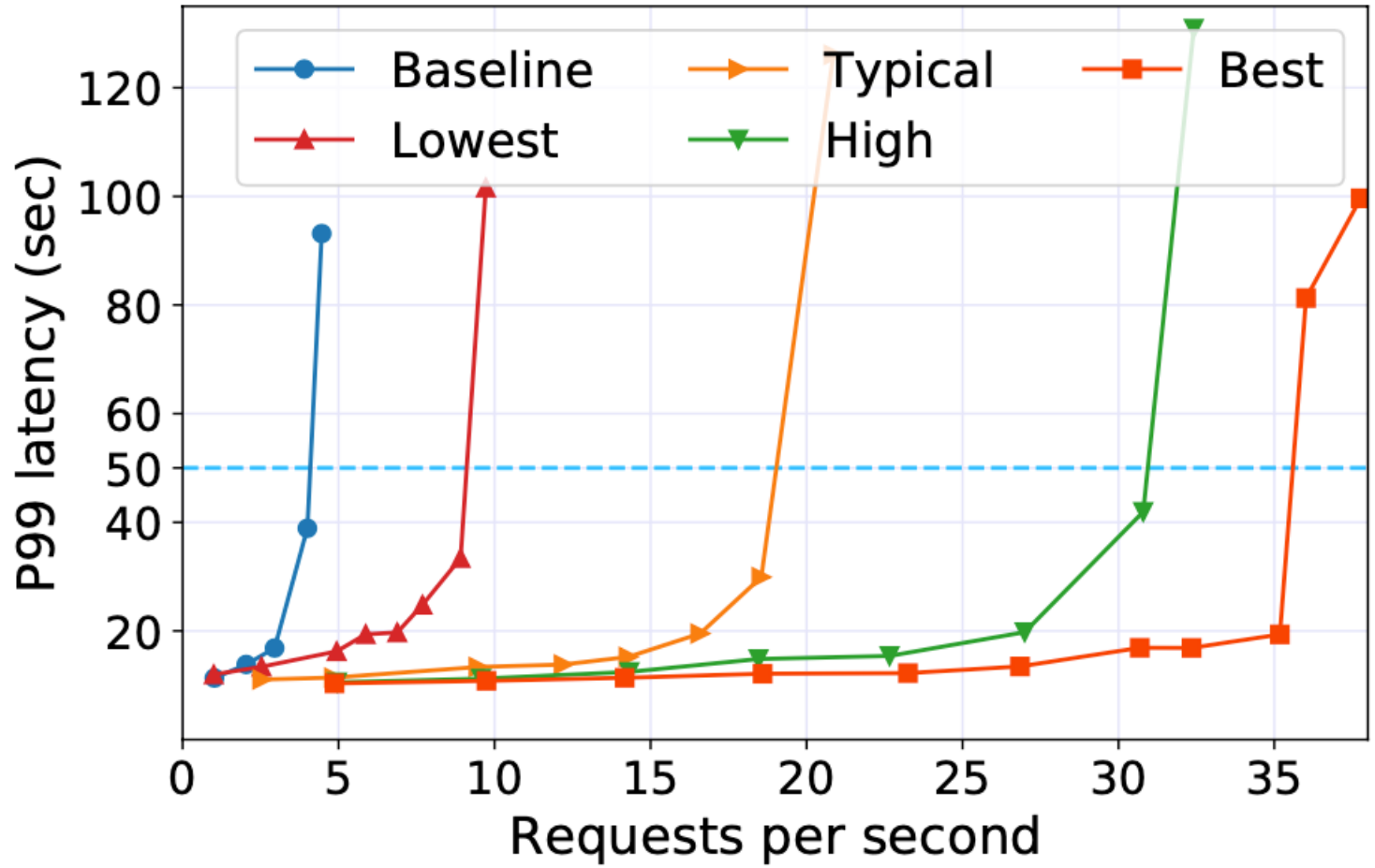


Figure 17: Regular vs Harvest VMs with same budget.

Conclusion

Propose to host serverless platforms on harvested resources

Quantify the challenges of using harvested resources for FaaS

Demonstrate the reliability of hosting serverless workloads on harvested resources with trace-driven simulation

Designed a harvesting-aware serverless load balancer on OpenWhisk

Thank you

Presented by Bocheng(Noah) Cui

Master’s in UNH

noahcui.github.io/info