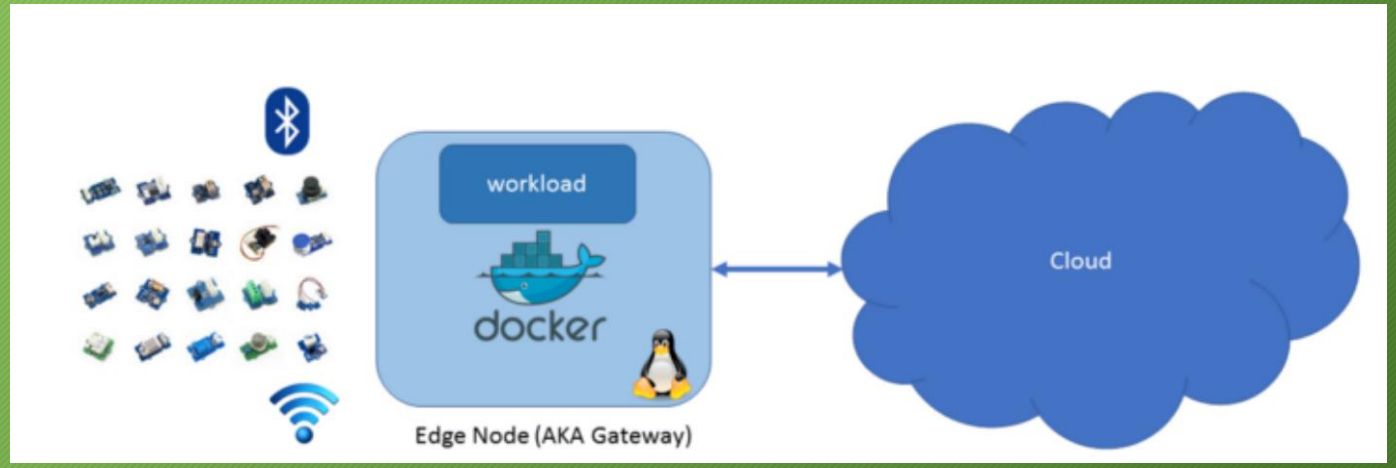
Serverless On Edge Lean-Openwhisk

Σταυρακάκης Κωνσταντίνος
Kostantinosst23@gmail.com
https://github.com/razkey23/Serverless-On-Edge

IoT & Serverless

Why Serverless on Edge?

- Necessity to process data on Edge nodes.
- Edge workloads are event-driven
- serverless programming model fits to event-driven applications like a glove



Serverless Frameworks

- Microsoft Azure Functions
- Amazon Lambda
- Apache Openwhisk

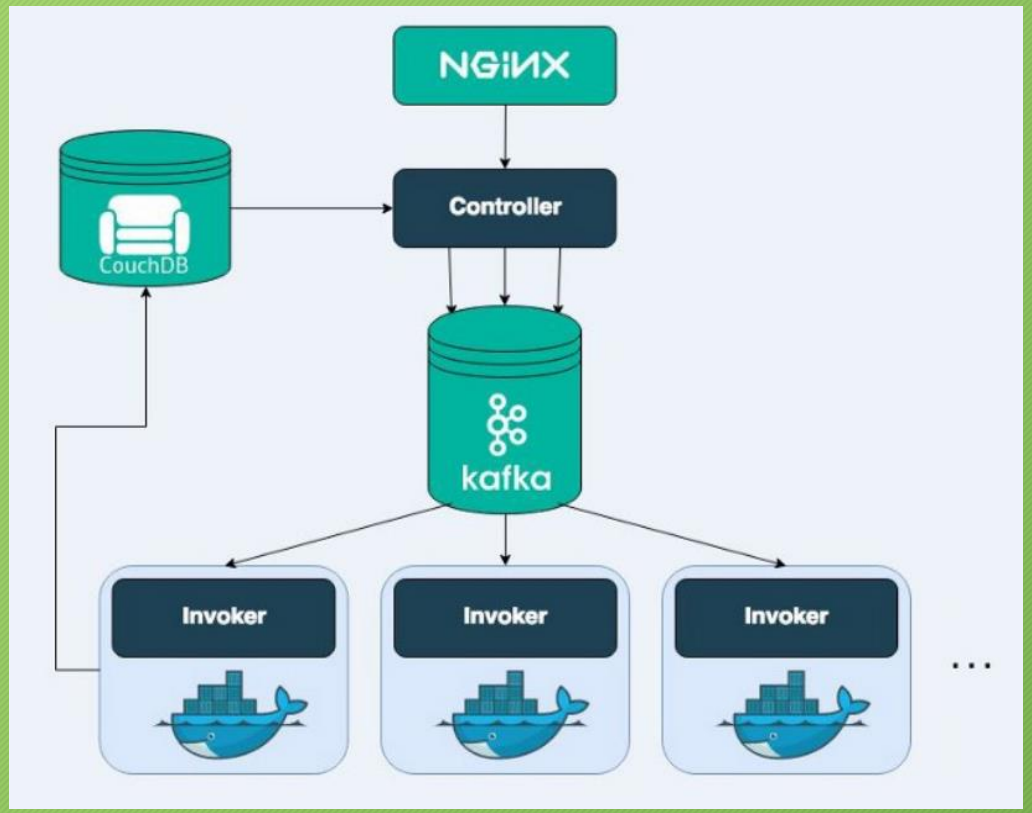
Can we use one of those on edge devices?

- Big Overhead
- Edge devices are resource constrained

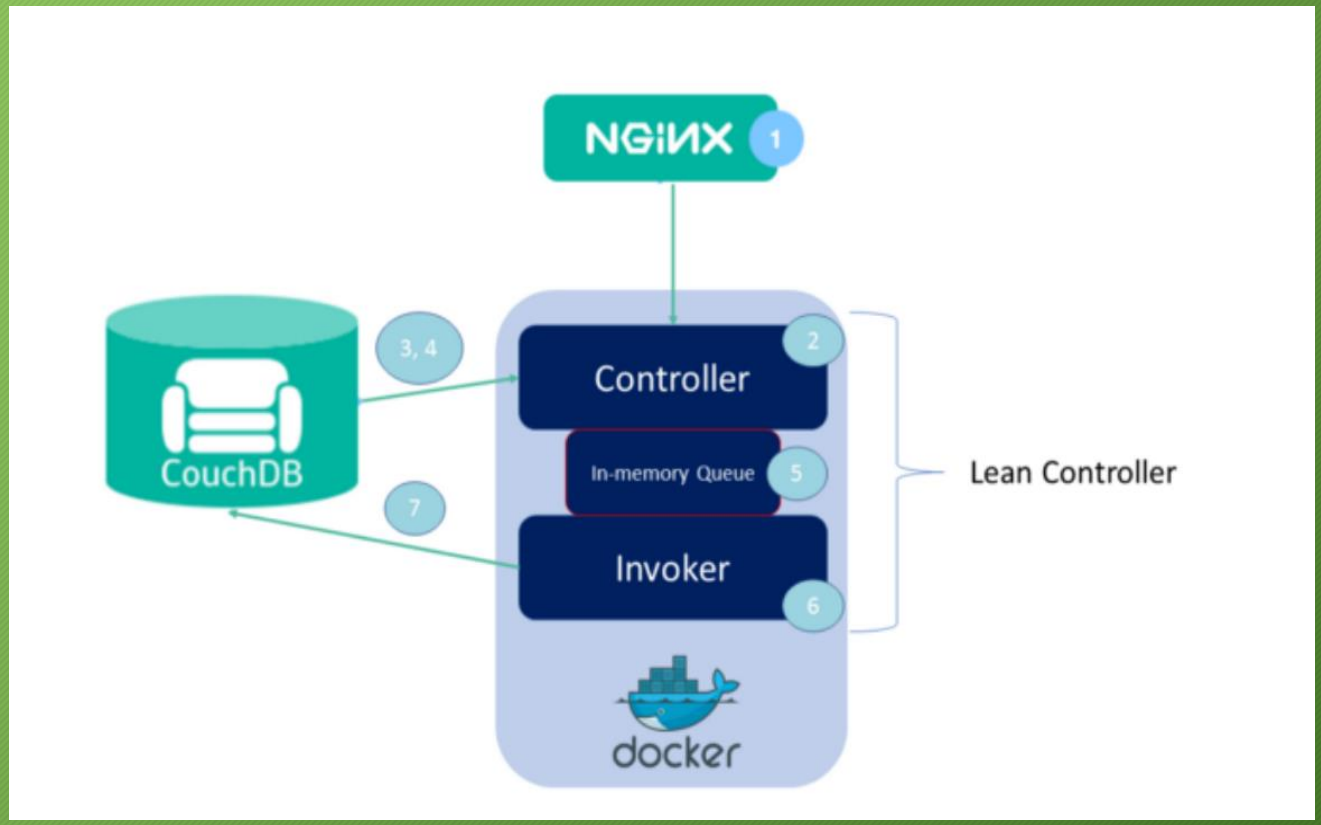
Alternative? Try to downsize them

Lean - Openwhisk

Openwhisk



Lean-Openwhisk



The Project

Goals

- Install Lean Openwhisk on Raspberry Pi 4
- Test our setup with Serverless Benchmarks
- Try to see if it is a viable option for a serverless architecture on edge nodes

Project Setup

Step 1 (Install Prerequisites)

- pip 19.1
- ansible 2.7.9
- Docker (Fresh install recommended)
- Clone github repo of Lean-Openwhisk

Step 2 (Pre-Install Configuration)

- Export needed enviroment variables
- Configure the metadata database
- pull docker images for nodejs runtime

Step 3 (Install Lean-Openwhisk)

- Install Lean-Openwhisk using ansible-playbook as shown on github

Step 4 (Install Wsk-Client)

- Download wsk-client from github
- Export wsk-cli binary enviroment variable
- Update ~/.wskprops file for authentication

How to use Lean-Openwhisk

Javascript action sample:

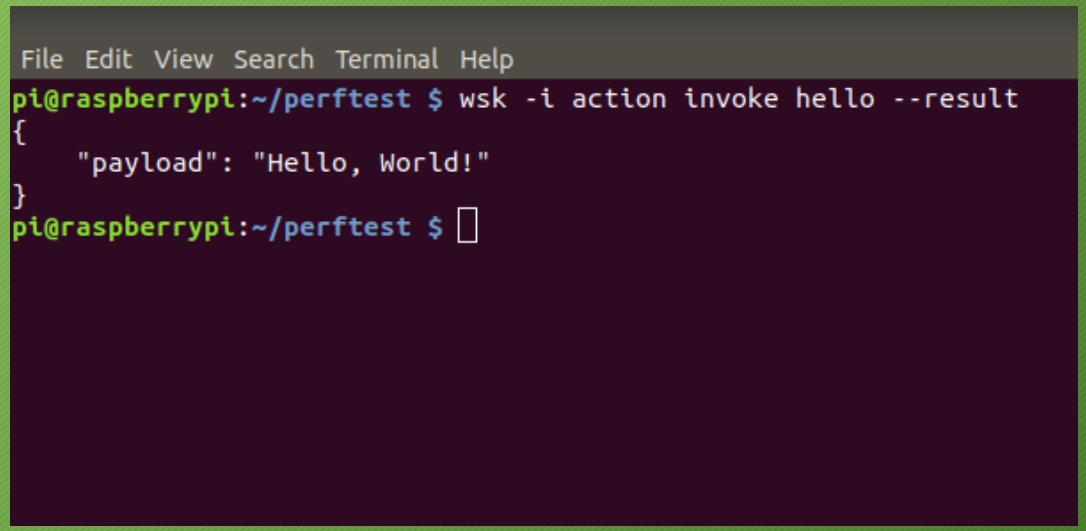
```
/**
 * Hello world as an OpenWhisk action.
 */
function main(params) {
 var name = params.name || 'World';
 return {payload: 'Hello, ' + name + '!'};
}
```

Now create action using wsk-cli:

```
$wsk action create hello hello.js
```

Now Invoke the action

```
$wsk action invoke hello --result
```



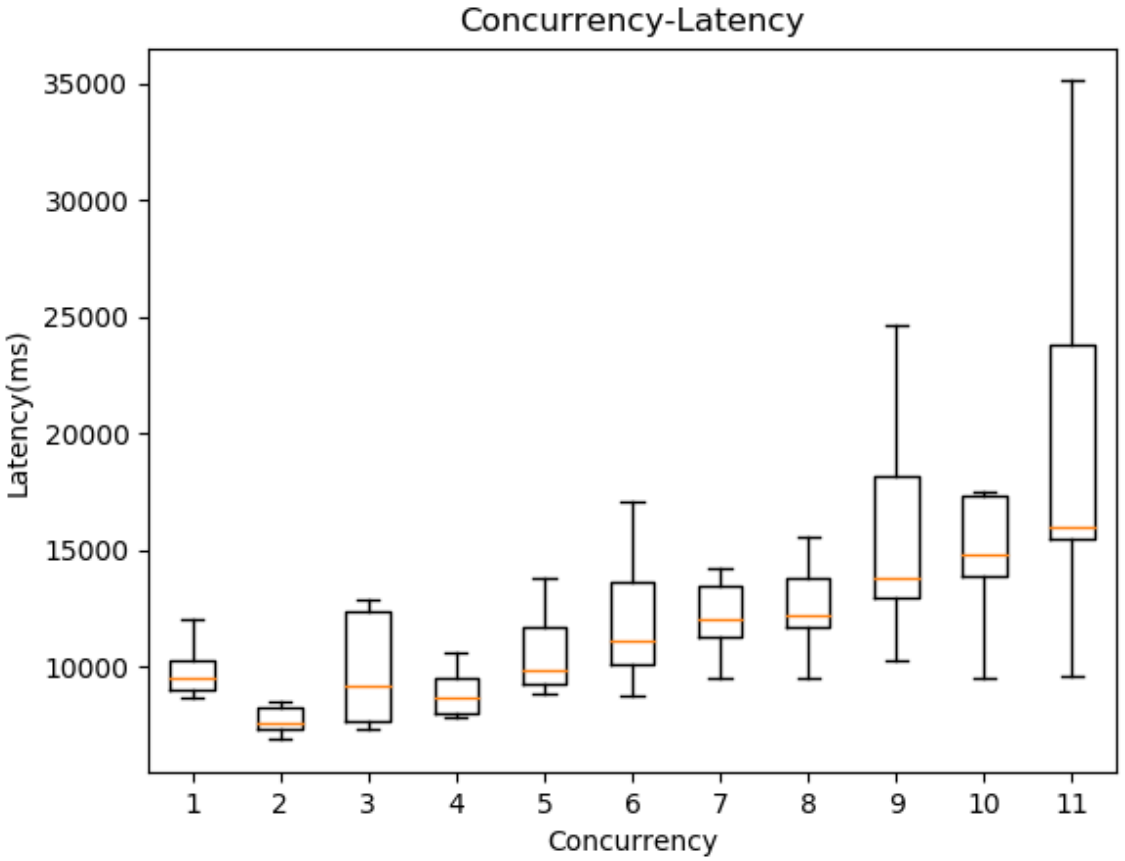
Lean-Openwhisk Experiments

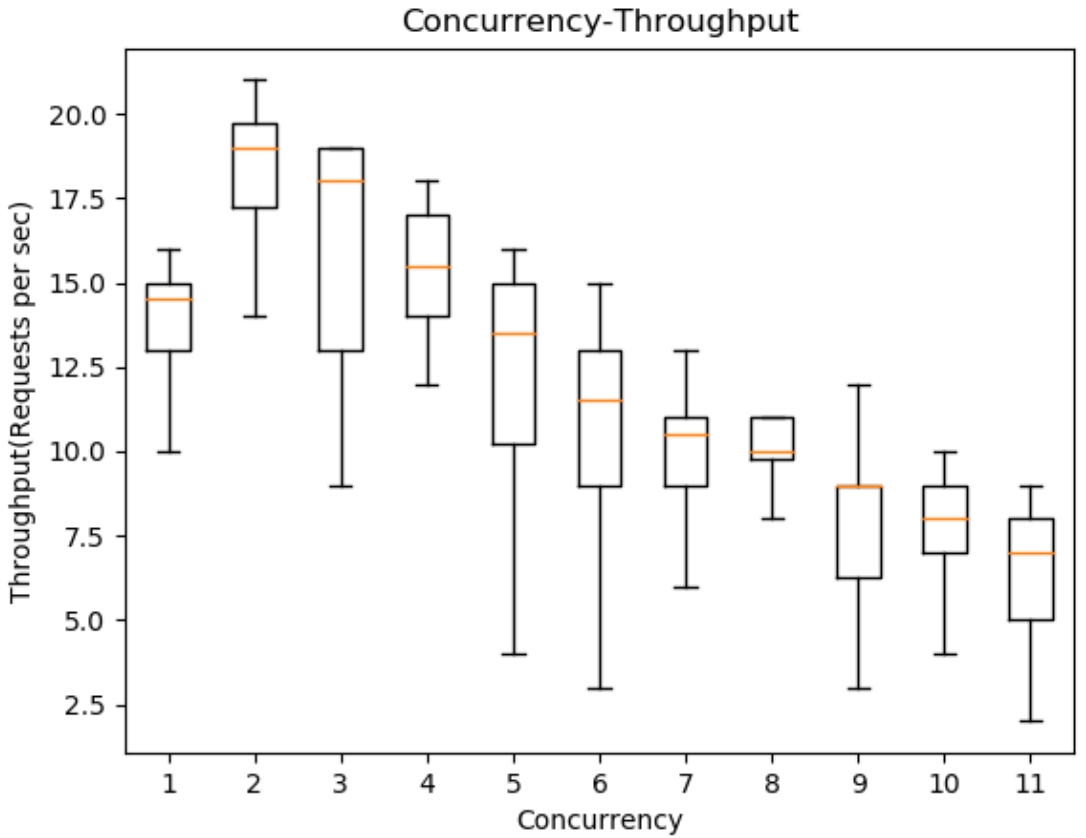
Experiment A : How concurrency affects latency & throughput

Experiment B : How different payloads affects latency & throughput

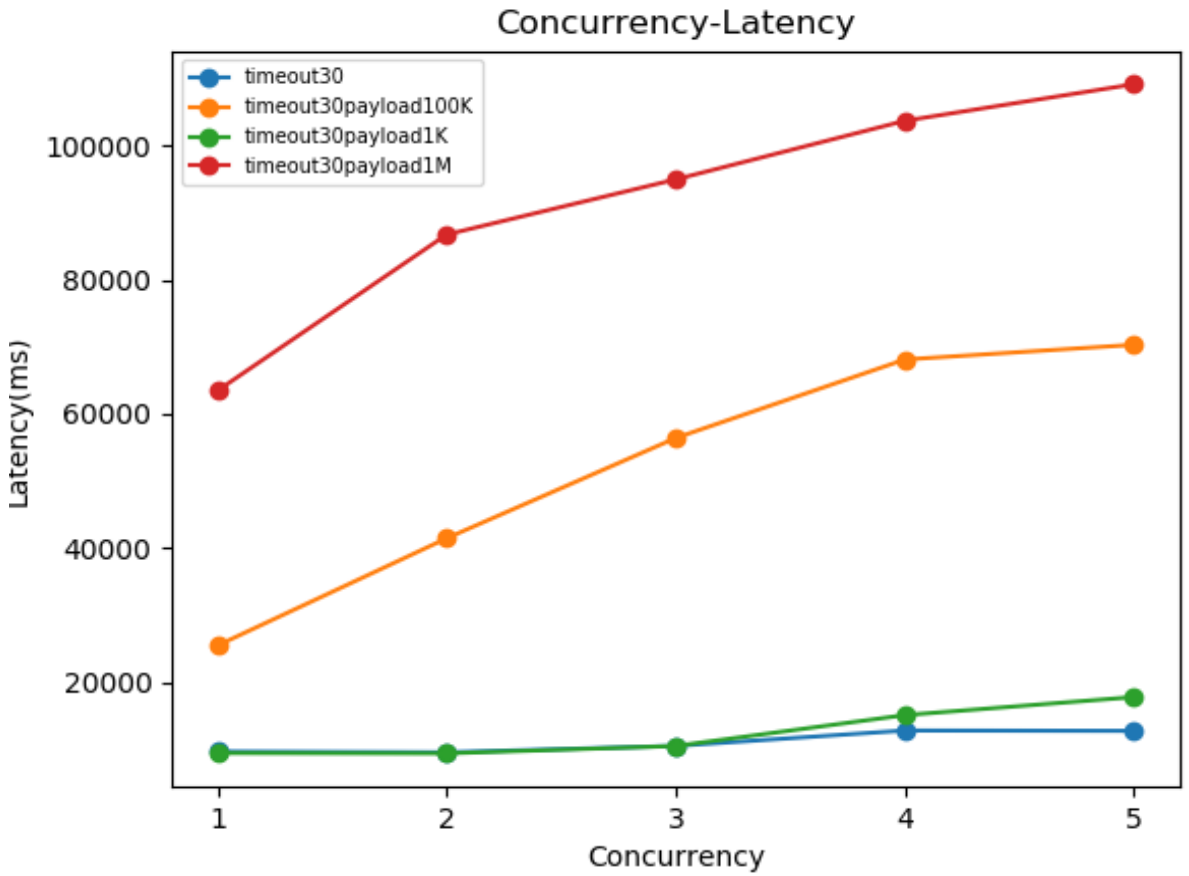
Experiment C : How transfering different payloads affects latency

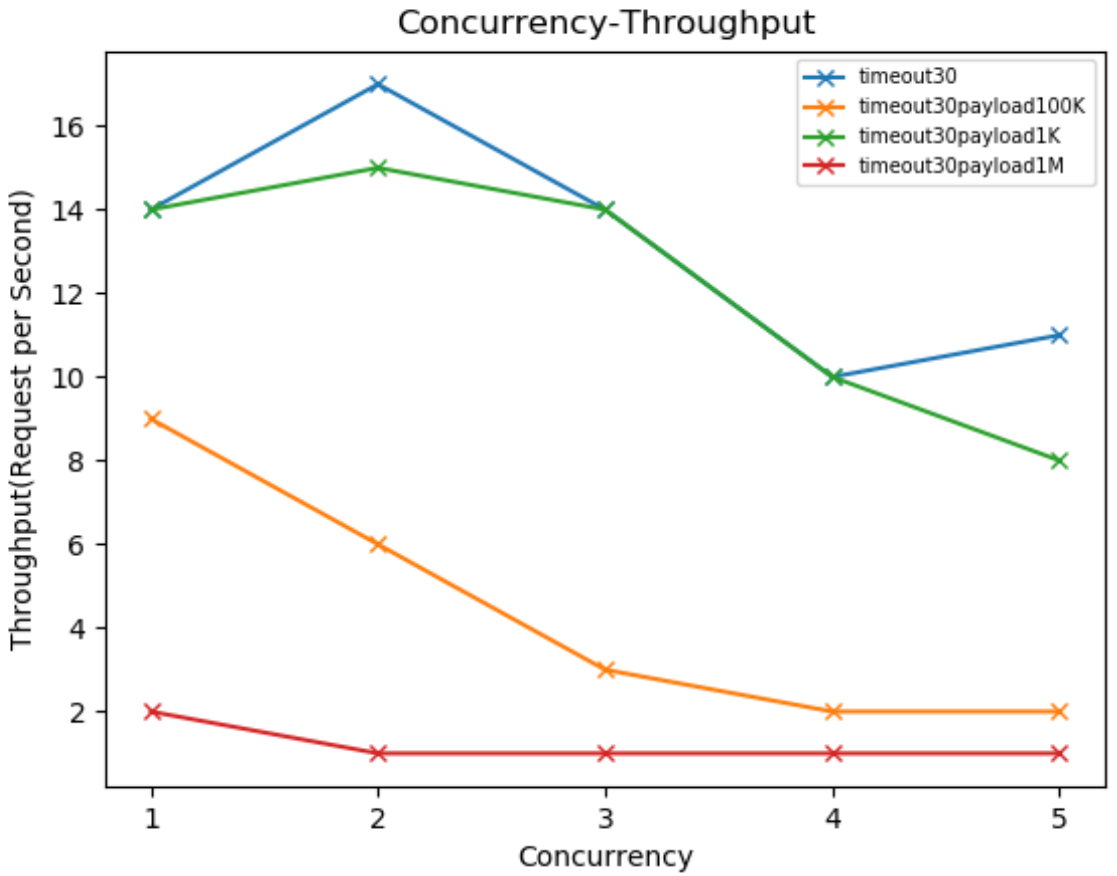
For Experiment A,B Lean-Openwhisk-Performance github repo was used ,while for Experiment C ServerlessBench Testcase5 was used



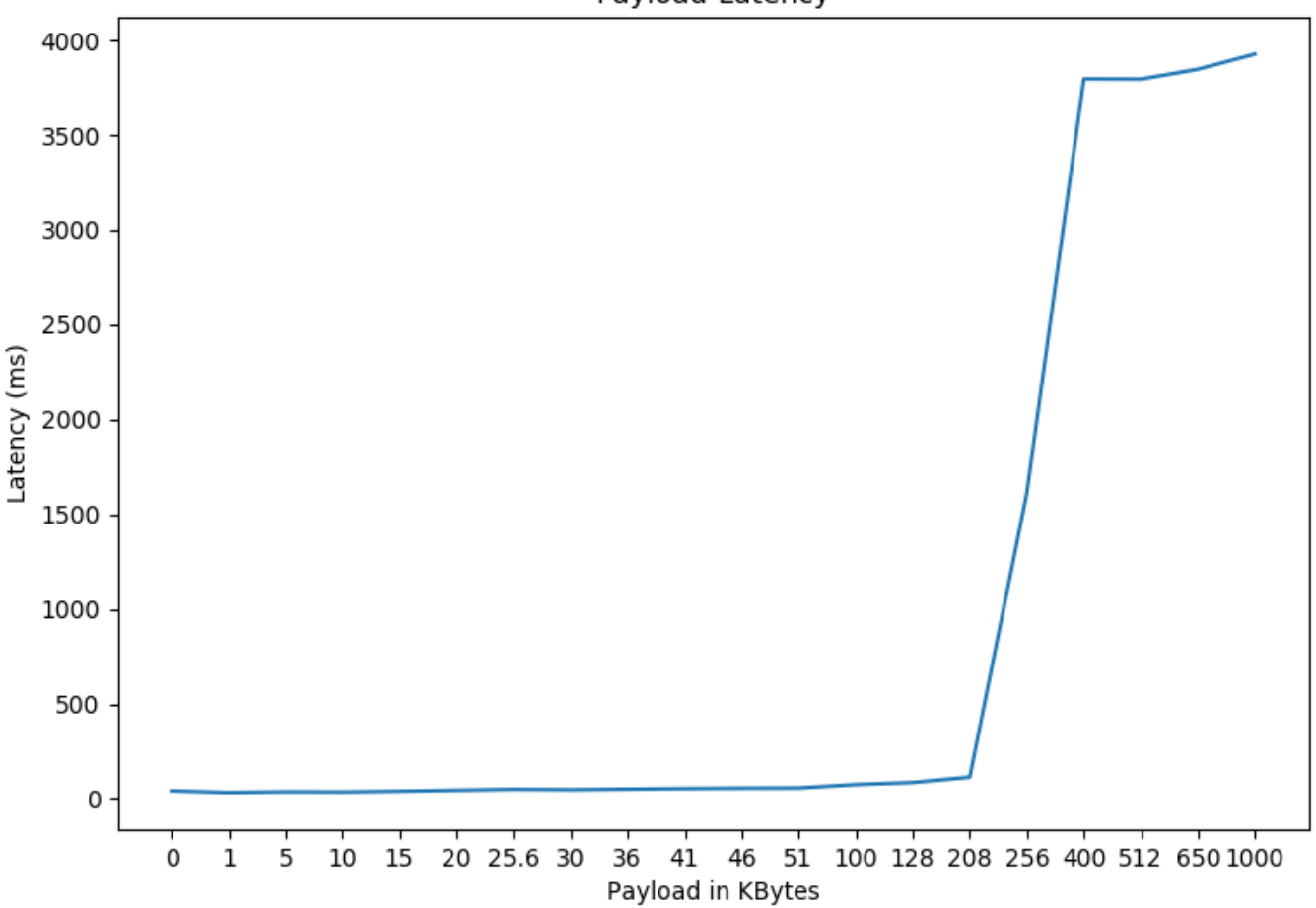


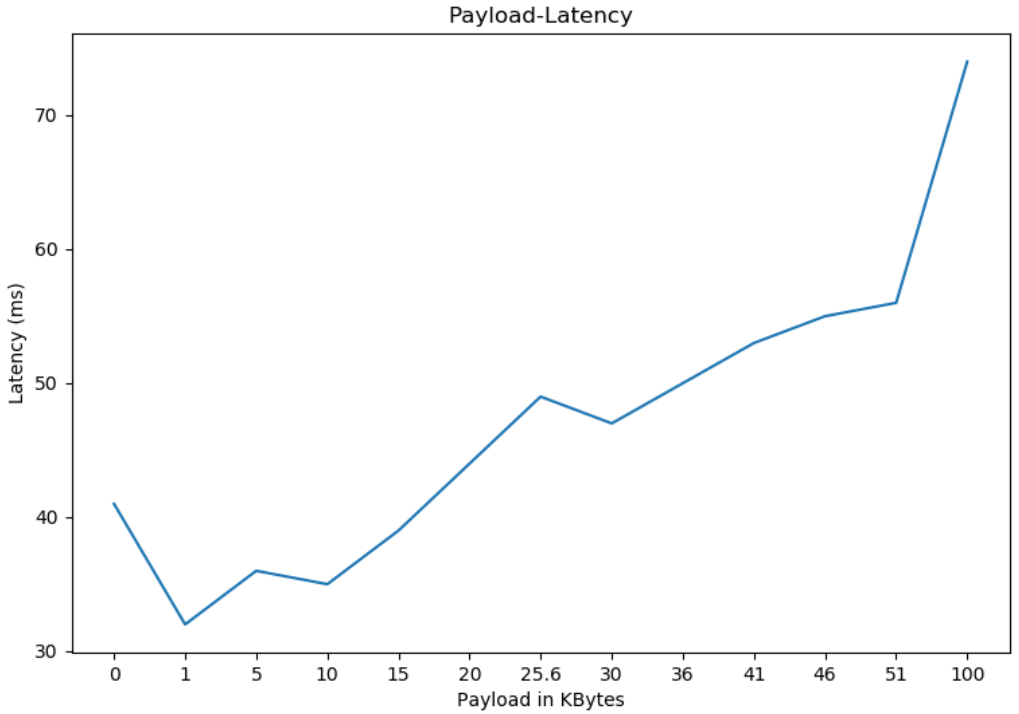
Experiment A





Experiment B





Experiment C

Conclusion

- Backend Concurrency is advised to be kept between 1 and 5.Otherwise latency is raised to unwanted levels and throughput plummets
- Payloads above 100KB-200KB are discouraged since latency skyrockets and the achieved throughput gets significantly low
- Data transfer for up to 100K doesn't cause significant increase in latency which seems promising for future optimizations in scheduling the functions (maybe partition big ones into smaller?)

Thank you

References:

https://medium.com/openwhisk/lean-openwhisk-open-source-faas-for-edge-computing-fb823c6bbb9b

https://medium.com/openwhisk/apache-openwhisk-meets-raspberry-pi-e346e555b56a

https://github.com/SJTU-IPADS/ServerlessBench

https://github.com/kpavel/lean-openwhisk-performance

https://serverlessbench.systems/socc20-serverlessbench.pdf

Here is a cookie if you got here