Optimizing and Extending Serverless Platforms
A Survey

Maziyar Nazari*, Sepideh Goodarzy*, Eric Keller*, Eric Rozner§, Shivakant Mishra*
*University of Colorado Boulder, CS Department
§Facebook

SDS 2021

University of Colorado
Boulder

Introduction

- **Pros**
  - Pay-for-Use
  - No worries about servers, developers!
  - Auto-scaling
  - Thousand-way parallelism
  - ...
- **Challenges**
  - provider! Do worry about servers
  - Unlimited resource illusion
  - Optimized resource management
  - ...




Introduction (Contd.)

Define

```
exports.handler = async (event, context) => {
  return {
    statusCode: 200,
    body: "We are now split testing!"
  };
}
```

Execution

3

3

3

2

3

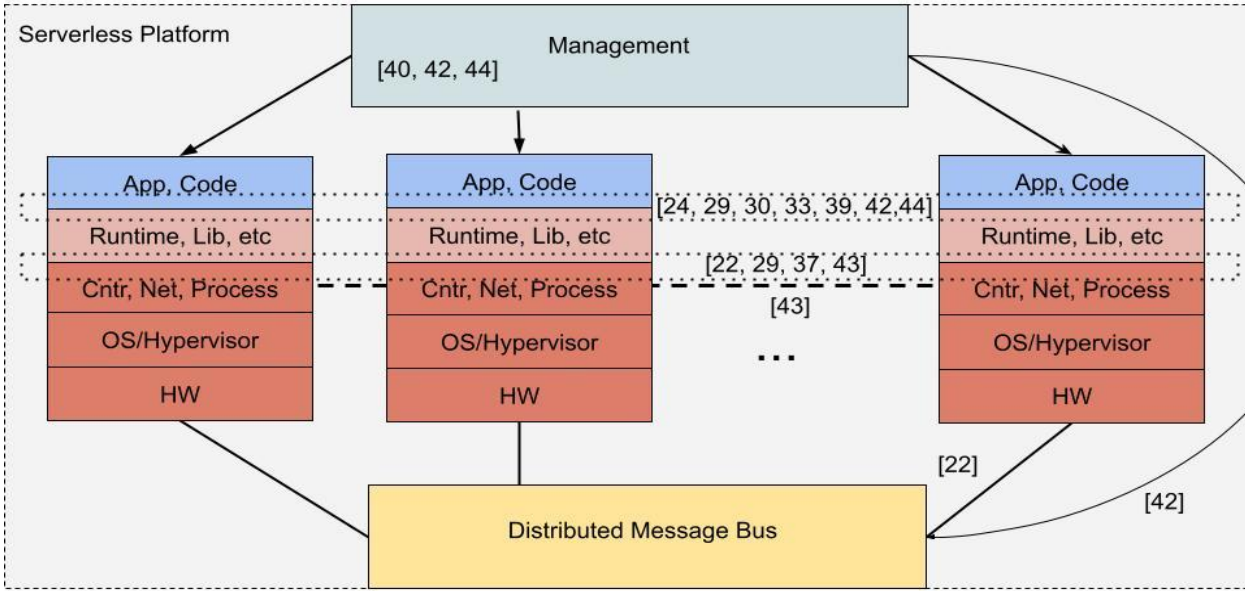
1

Events: Ex. 3 web requests

Serverless Platform

Ex. 3 web responses

In this paper ...



- Applications
- Serverless platform Optimizations
- Extensions

Applications

Categories of Applications

- Big Data Analytics
  - *ex. PyWren, IBM-PyWren*

- Cloud Offload
  - *ex. gg, ExCamera, SNF*

Big Data Analytics

- **PyWren***
  - Distributed Computing is hard for non-expert users
  - 3 main challenges in Distributed Computing world
    - Complex, Hard-to-Configure
    - Resource Management
    - Reliable execution with efficient pricing

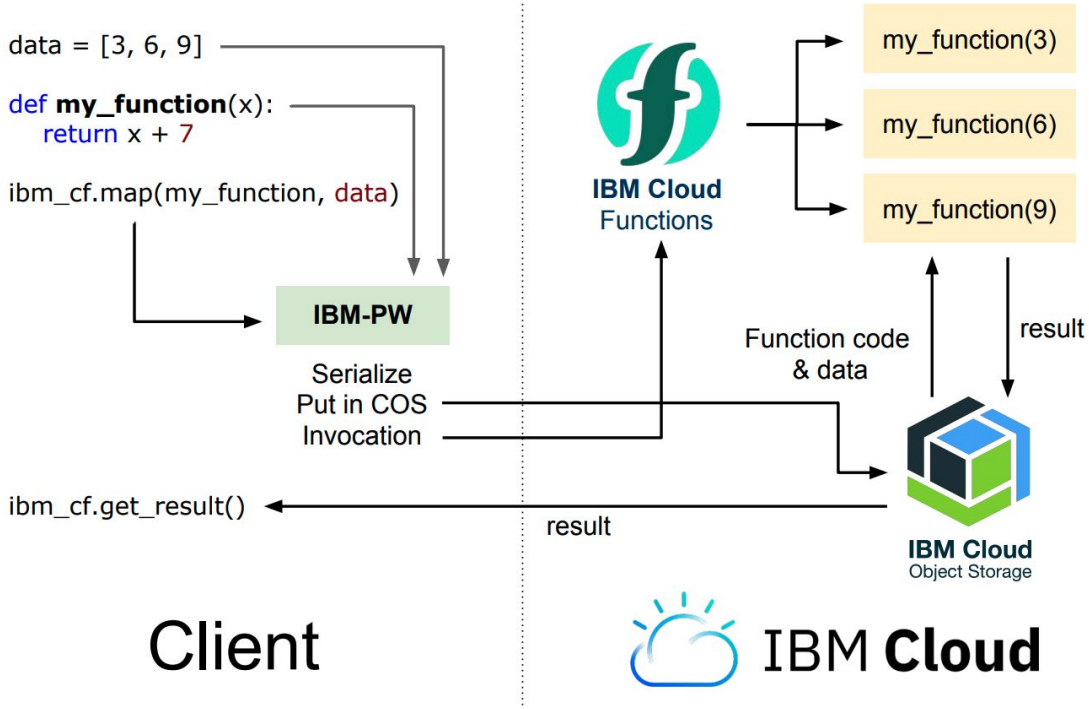
*Jonas, Eric, et al. "Occupy the cloud: Distributed computing for the 99%." *Proceedings of the 2017 Symposium on Cloud Computing*. 2017.

Big Data Analytics

- ***PyWren***
  - Data processing atop stateless functions
  - System Components
    - ***Stateless functions***
    - ***Scheduler***
    - ***Remote Storage***
  - Propose
    - ***Fault tolerance***
    - ***Simplicity***

Applications: Big Data Analytics

- ***IBM-PyWren****
  - Industry-scale PyWren implementation & extension



*Sampé, Josep, et al. "Serverless data analytics in the ibm cloud." *Proceedings of the 19th International Middleware Conference Industry*. 2018.

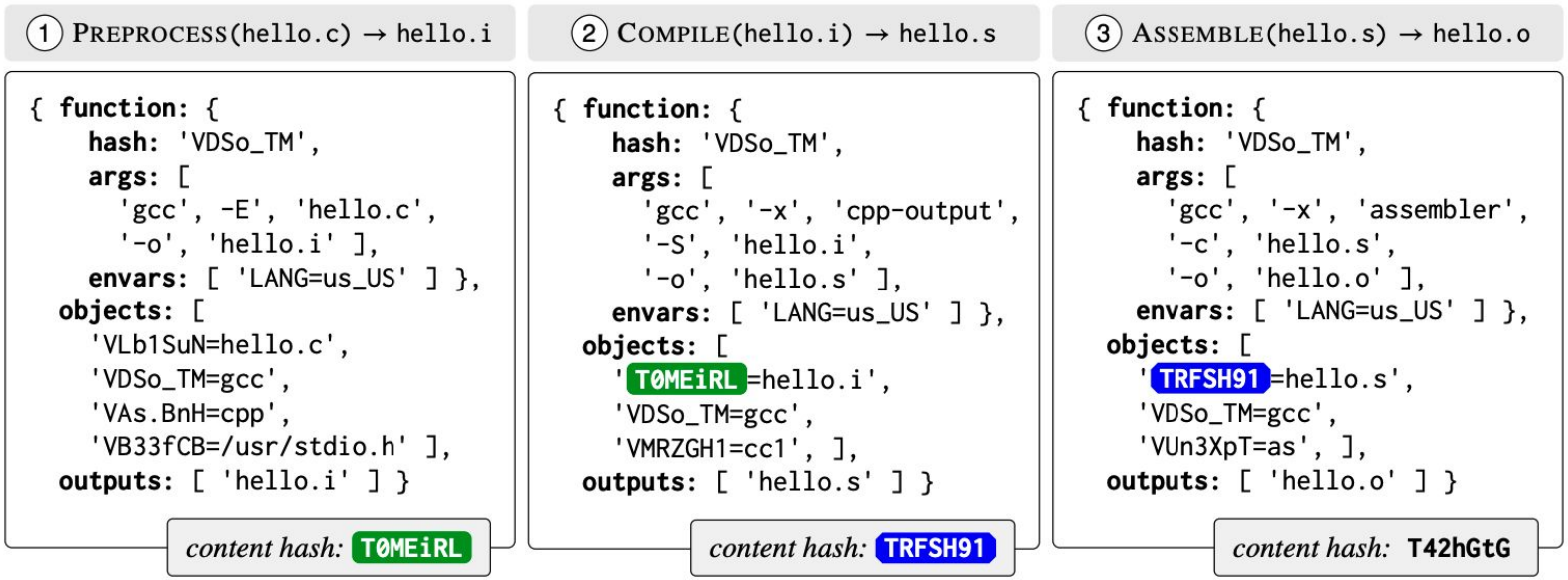
Cloud Offload

- ***gg****
  - ***Parallel Compilation*** atop AWS lambda
  - Context:
    - Offloading everyday jobs to the cloud leveraging the ability of running burst-parallel apps
  - Challenges:
    - In outsourcing applications, software dependencies must be managed
    - Local-Cloud communication must be minimized
    - Serverless functions must be easy to use

*Fouladi, Sadjad, et al. "From laptop to lambda: Outsourcing everyday jobs to thousands of transient functional containers." *2019 {USENIX} Annual Technical Conference ({USENIX}{ATC} 19)*. 2019.

Cloud Offload

- ***gg***
  - Dependency management using “thunk”s
  - Containers can reference each other’s output
    - ***gg IR (Intermediate Representation)***



Cloud Offload

- ***gg***
  - Compiles Chromium in 18 mins on AWS Lambda
    - ***2.2x faster than icecc* on 384-core cluster***
  - Compiles Inkscape^ in 87 secs
    - ***4.8X faster than icecc on 384-core cluster***
  - Running LibVPX# unit tests
    - ***gg outperforms 4-way and 48-way parallelism local runs***

*https://github.com/icecc/icecream
^https://inkscape.org/
#https://www.webmproject.org/code/

Cloud Offload

- ***SNF****
  - Network Function as a Service
  - NFaaS platform should meet the following requirements:
    - Intuitive Programming Model
    - Low Latency Packet Processing
    - Auto scaling to meet the demands
  - **Serverless has the necessary building blocks**
  - Challenges:
    - Serverless functions are stateless
    - Coupling between work allocation and billing granularity

*Singhvi, Arjun, et al. "Snf: Serverless network functions." *Proceedings of the 11th ACM Symposium on Cloud Computing*. 2020.

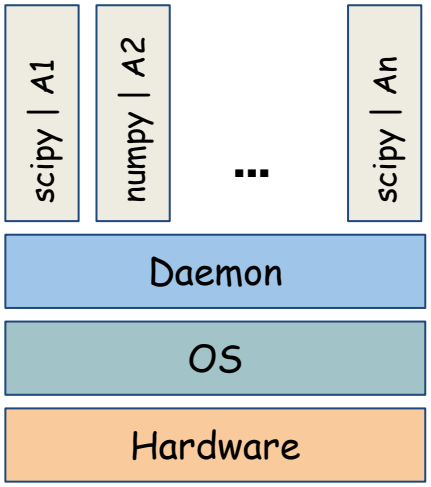
Cloud Offload

- ***SNF***
  - Decoupling Work Allocation & Billing Granularity
    - ***Middleground for Compute & State Decoupling***
    - ***Flows > Flowlets > Packets***
  - State Sharing Abstraction
    - ***Proactive State Replication***

Optimizations

Optimization Goals

- Serverless Platform Optimizations
  - Start up latency
  - Resource Utilization
  - Inter-function Communication

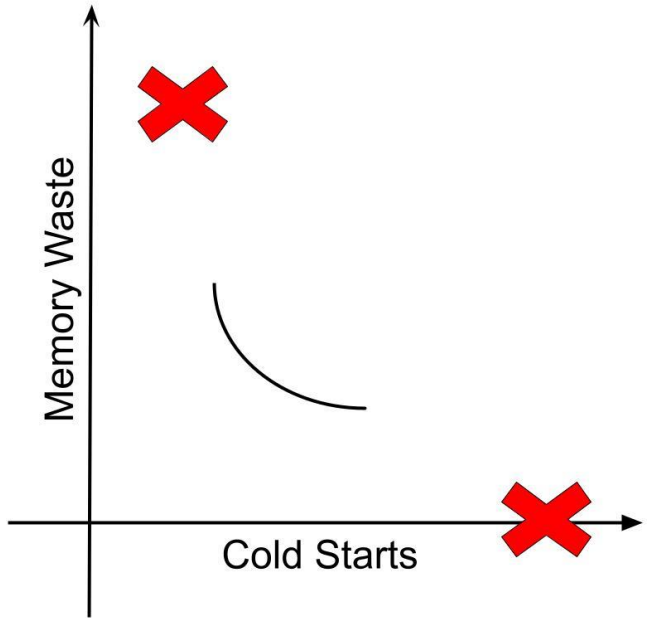


Docker Container: 400ms
Python Interpreter: 30ms
scipy:
- download 2700ms
- installl: 8200ms
- import: 88ms

Startup Latency & Cold Start

- *Trade-off*: Cold Start vs Resource Efficiency

- Startup Latency Optimization Methodologies:
  - Smart Algorithms
  - Relaxing Function Isolations



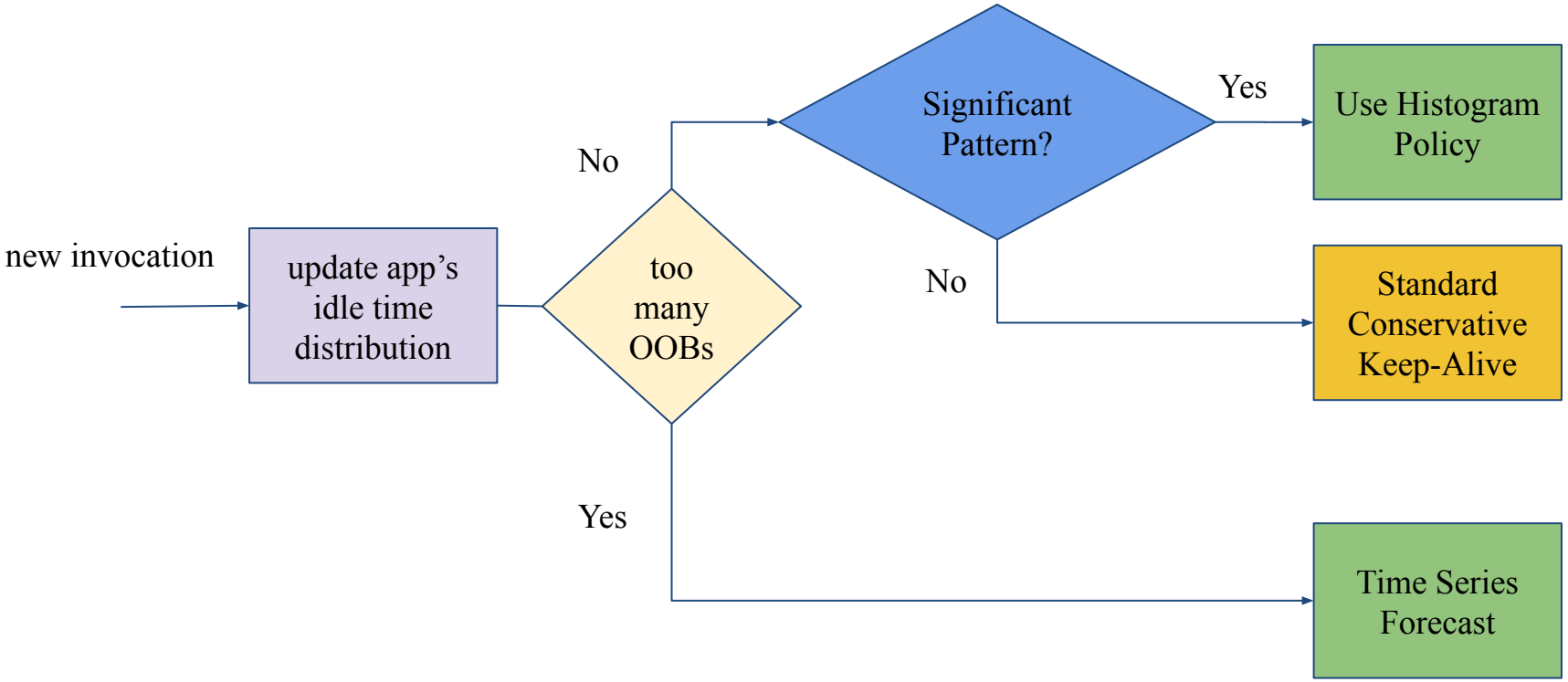
Smart Algorithms

- *Microsoft Azure Functions**
- Reducing the number of Cold Starts
- Challenges
  - Serverless Workload Types Are So Variant
  - Serverless Platforms incorporate fixed keep-alive policy

*Shahrad, Mohammad, et al. "Serverless in the wild: Characterizing and optimizing the serverless workload at a large cloud provider." *2020 {USENIX} Annual Technical Conference ({USENIX}{ATC} 20)*. 2020.

Smart Algorithms

- Put it in a nutshell:

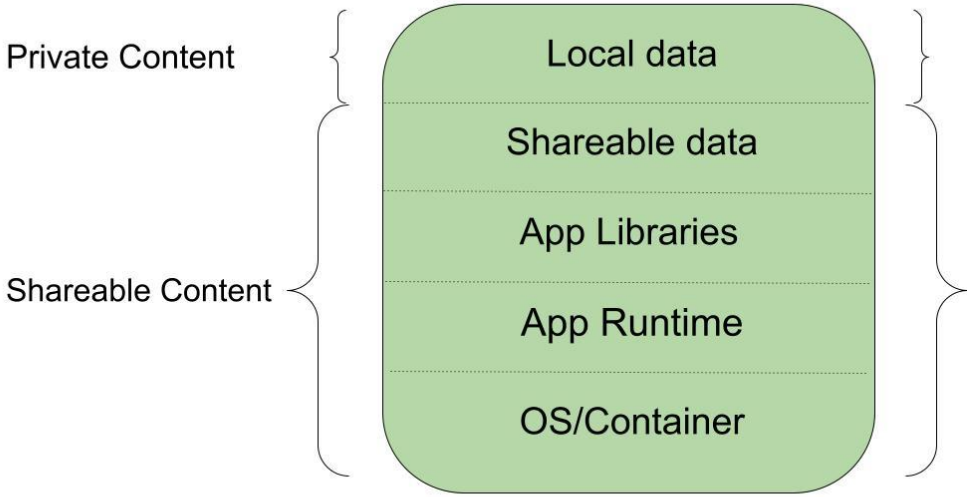


Smart Algorithms

- Evaluation
  - Implemented on OpenWhisk
    - ***Memory Reduction: 15.6%***
    - ***99th-percentile execution time reduction: 82.4%***
    - ***Overhead: < 1ms To The Critical Path Latency***

Relaxing Function Isolation

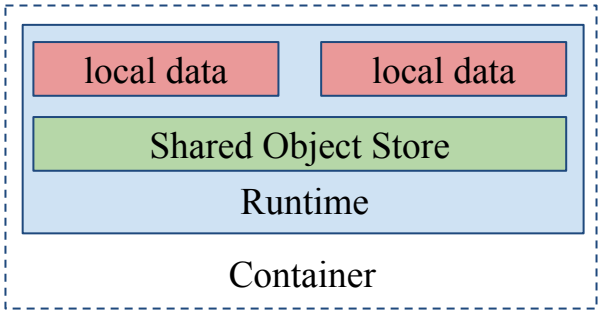
- *Photons**
  - 70-90% of the memory footprint for the concurrent functions (in the same app) is shareable



*Dukic, Vojislav, et al. "Photons: Lambdas on a diet." *Proceedings of the 11th ACM Symposium on Cloud Computing*. 2020.

Relaxing Function Isolation

- Strong isolation is not necessary for the concurrent functions of the same application
- Run concurrent functions in the same JVM
  - Using language level isolation
- Shared Object Store with a GET/SET API
  - Inter-function communication
  - Synchronization Mechanism



Relaxing Function Isolation

- Running 100 concurrent functions
  - memory consumption: ⇩5X
- more concurrency => better performance
  - Using Azure workload
    - ***30% less memory***
- Keep more containers warm
  - 52% less cold starts

Relaxing Function Isolation

*Particle**

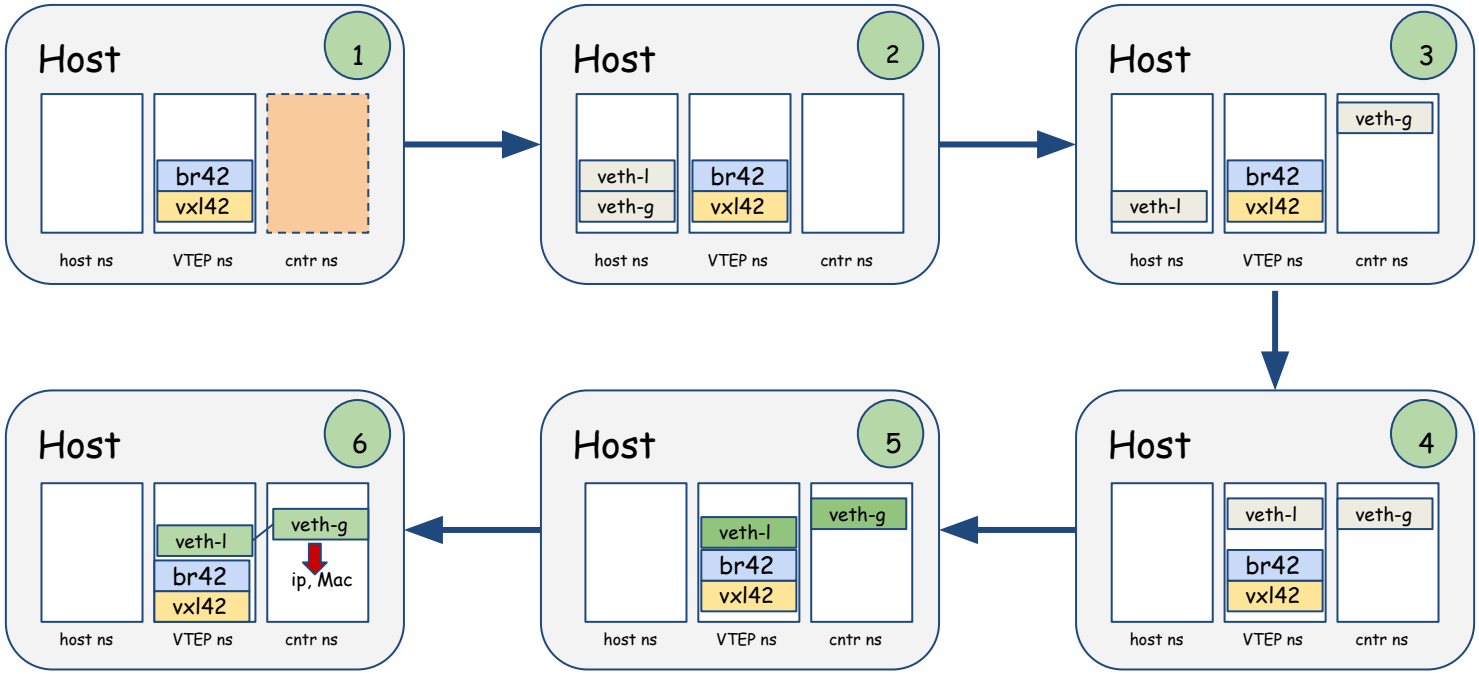
- Network Startup Time
- Using Storage-based communication is not ideal
- Overlay network for container communication
  - Flannel, Weave, Docker Swarm

	User Code	Container Startup	Network Startup
Linux Overlay	6%	26%	68%
Docker Swarm Overlay	4%	16%	80%
Weave Overlay	6%	25%	69%

Interconnecting 100 Tasks with Overlay Network

*Thomas, Shelby, et al. "Particle: ephemeral endpoints for serverless networking." *Proceedings of the 11th ACM Symposium on Cloud Computing*. 2020.

Relaxing Function Isolation



Relaxing Function Isolation

- Profiling with eBPF
- Steps 3,4
  - `dev_change_net_namespace`
    - Startup/cleanup
    - Hold locks
    - twice (steps 3, 4) for each container

Step	Time	Percent
1	0.1	0.92%
2	0.1	0.92%
3	5.18	47.71%
4	4.77	43.95%
5	0.49	4.45%
6	0.22	2.03%

Relaxing Function Isolation

- Consolidate Network Interfaces
  - Share Network Namespace between the functions of a single tenant
    - ***Share VETH devices*** => only perform 2 locks per app

- Batch commands if possible

Relaxing Function Isolation

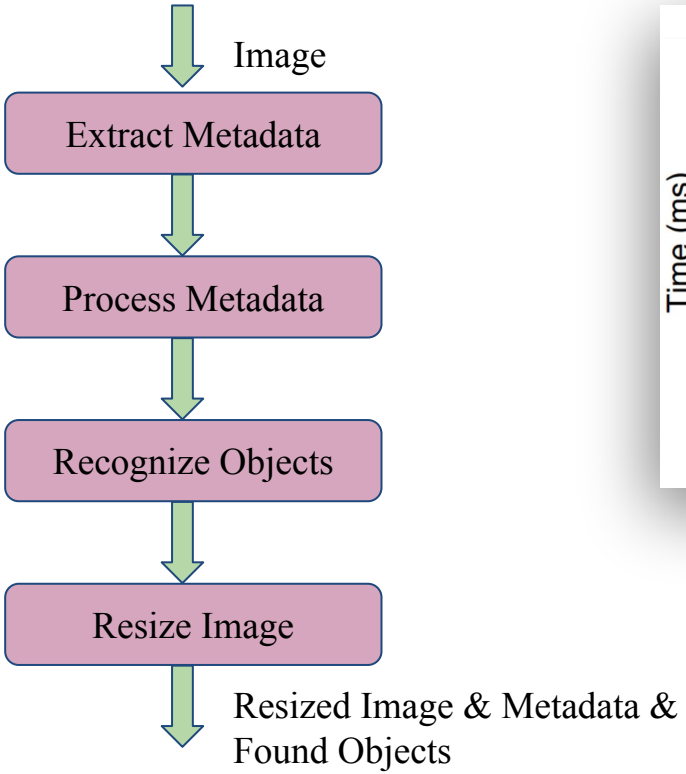
- Reduced total runtime of burst-parallel workload
  - 2.5x better than Linux Overlay
  - 2.8x better than Weave Overlay
  - 5x better than Docker Swarm
- Running Sprocket*
  - 3-stage video processing pipeline
  - Using Particle, results are skewed towards processing

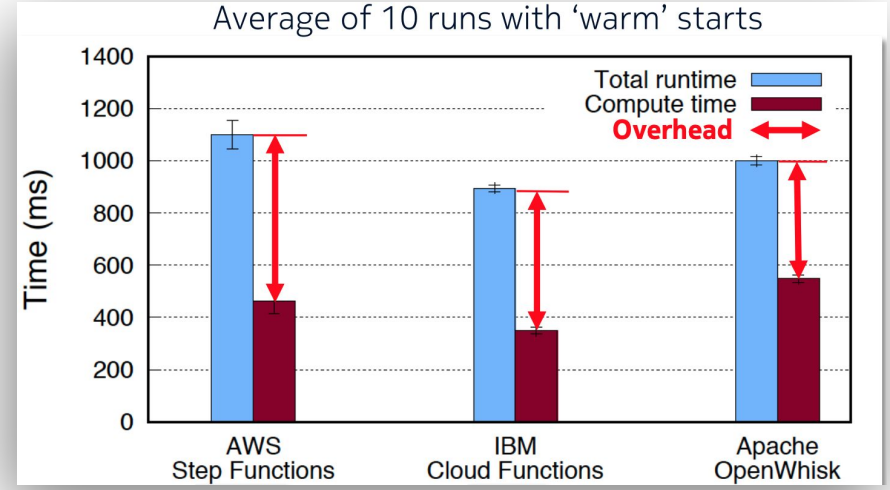
	Startup	Transfer	Process
Docker Swarm	69.86%	1.89%	28.25%
Linux Overlay	62.12%	2.50%	35.38%
Particle	17.77%	10.92%	71.31%

*Ao, Lixiang, et al. "Sprocket: A serverless video processing framework." *Proceedings of the ACM Symposium on Cloud Computing*. 2018.

Relaxing Function Isolation

- SAND*

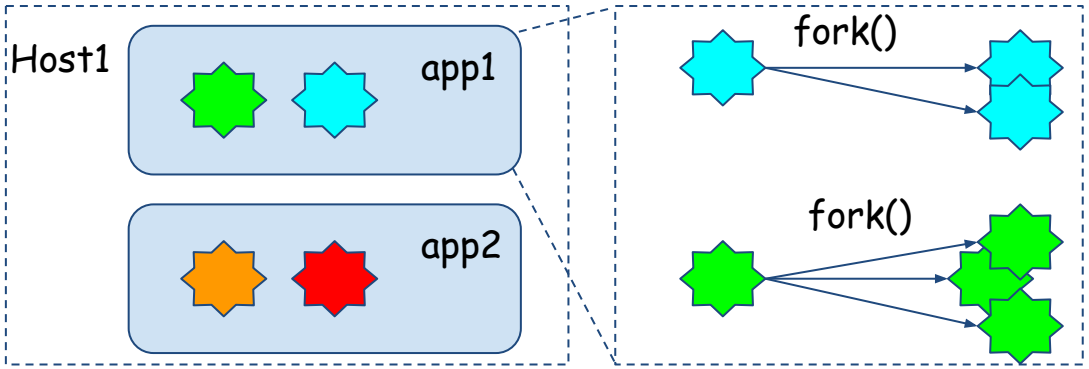




*Akkus, Istemi Ekin, et al. "{SAND}: Towards High-Performance Serverless Computing." *2018 {Usenix} Annual Technical Conference ({USENIX}{ATC} 18)*. 2018.

Relaxing Function Isolation

- Different levels of isolation
  - Application >> Container
  - Function >> Process
- Second Contribution?
  - Next Section!

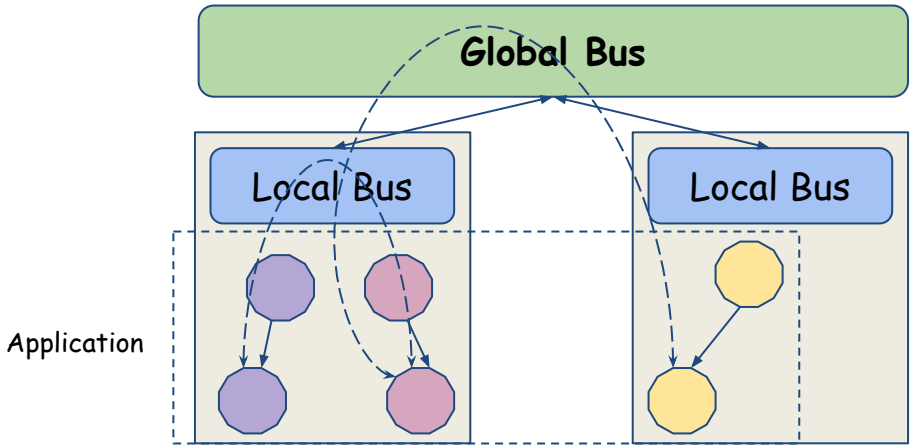


Advantages:
- Low Execution Footprint
- Automatic Resource Deallocation
- Copy-on-Write

Inter-function Communication

SAND

- First Contribution ✅
  - ***Application-level Sandboxing >> Different levels of isolation***
- Second Contribution
  - ***Hierarchical Message Bus***



Inter-function Communication

- Image processing pipeline:
  - 43% reduction in total runtime compared against OpenWhisk
- local message bus is 3-5x faster than the global one
- Function interaction is faster
  - 6.3x better than AWS Greengrass
  - 8.3x better than OpenWhisk

Inter-function Communication

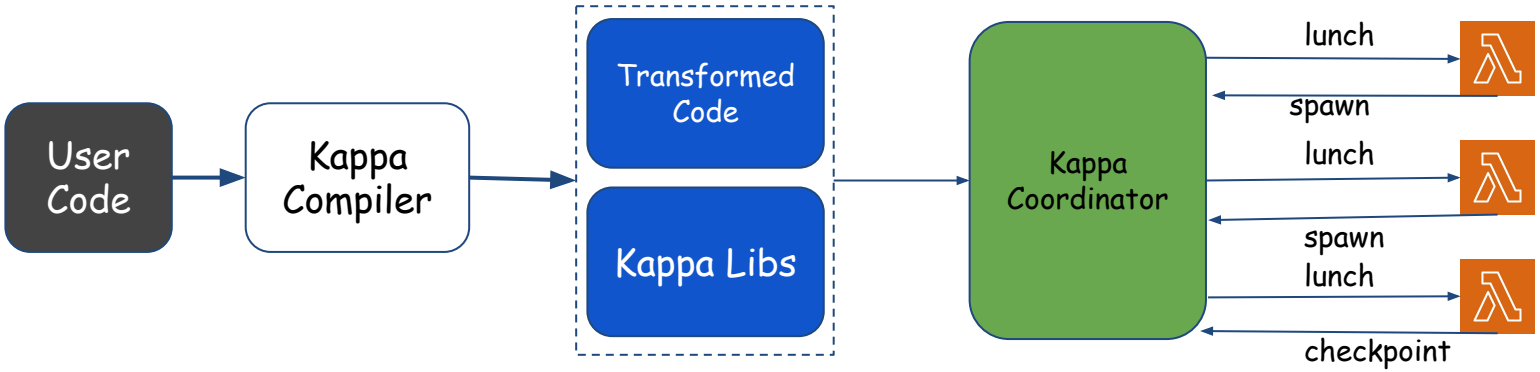
- Summary of inter-function communication methods
  - *Photons*
    - **Shared Object Store**
  - *Particle*
    - **Overlay Network**
  - *PyWren*, ... (Most Common)
    - ***External Storage Service***
  - *SAND*
    - ***Hierarchical Message Bus***

Extensions

Extensions

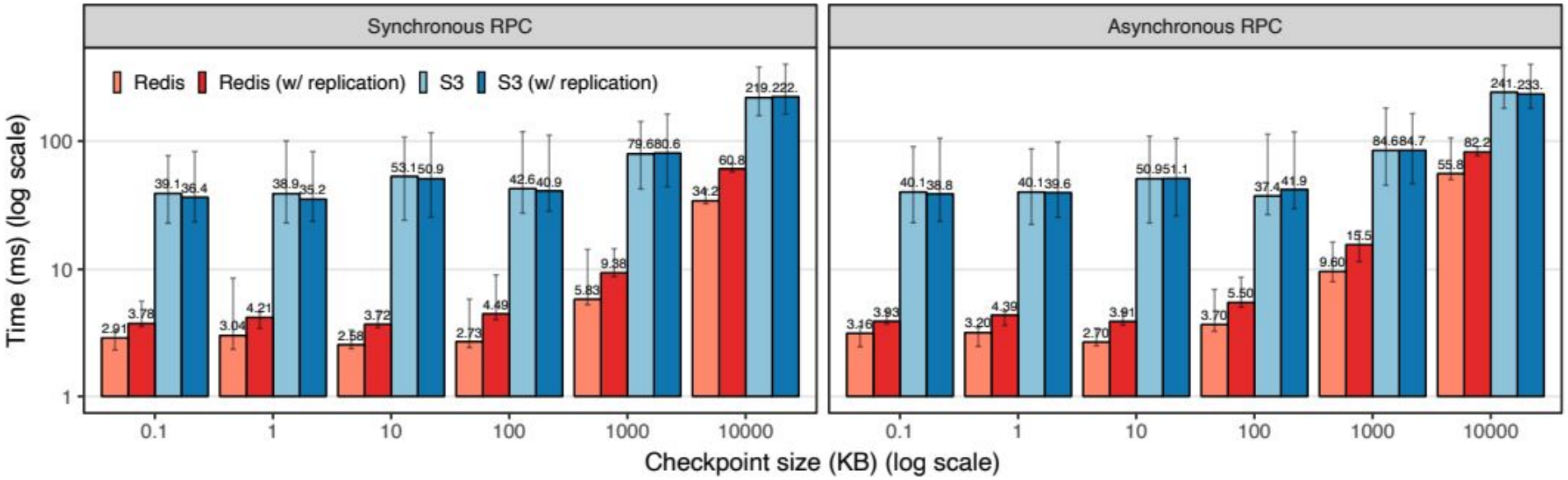
Kappa*

- Challenges:
  - Lambda functions are not suitable for long running jobs
  - Existing FaaS platforms lacks concurrency primitives
- Solution
  - Checkpointing mechanism
  - Provide an API



*Zhang, Wen, et al. "Kappa: A programming framework for serverless computing." *Proceedings of the 11th ACM Symposium on Cloud Computing*. 2020.

Extensions



Using Redis, upto 100KB chekpointing every 1s, Kappa adds < 1% overhead

Chekpointing (size: 0.5KB) every 100ms: 1000 parallel lambdas ~ 1 lambda

Discussion & Future Directions

- Resource Disaggregation + Serverless

- Serverless + Non-volatile memory
  - Faster than Disk & Remote Storage
  - Cheaper than in-memory cache
  - According to literature*
    - ***No support for specialized HW***


?
University of Colorado
Boulder