Kappa
A Programming Framework for Serverless Computing

https://kappa.cs.berkeley.edu

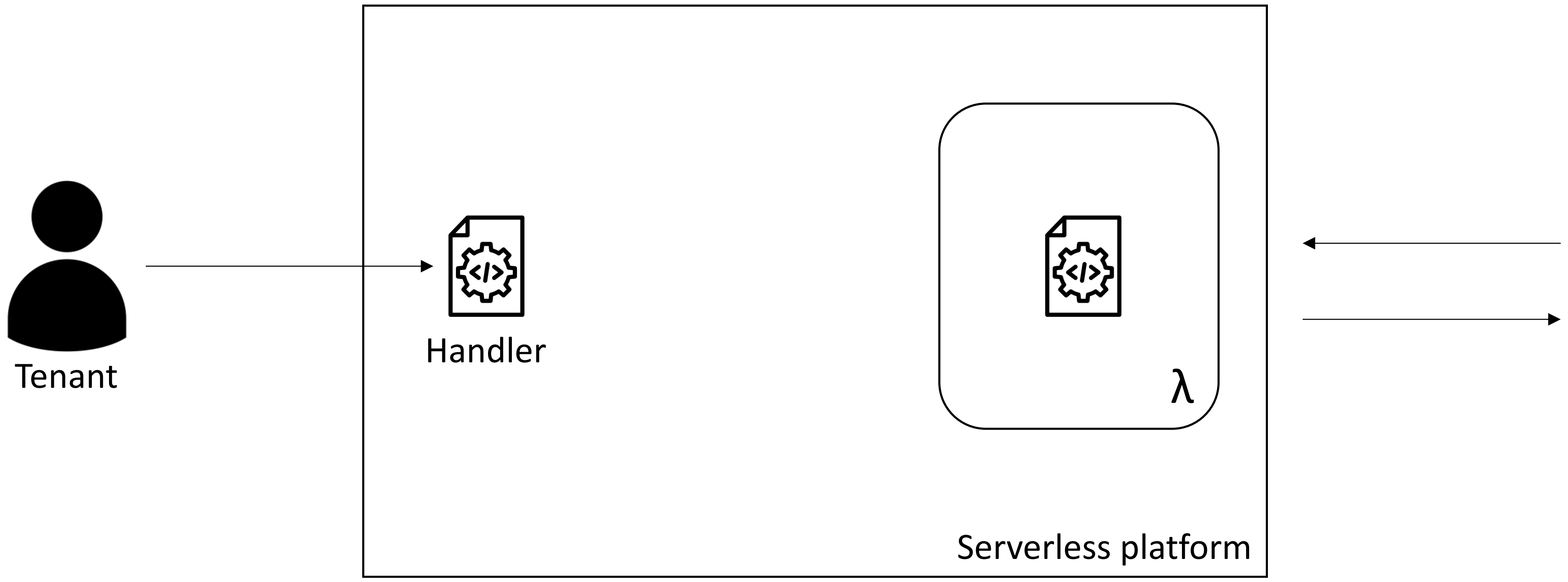
Wen Zhang UC Berkeley

Vivian Fang UC Berkeley

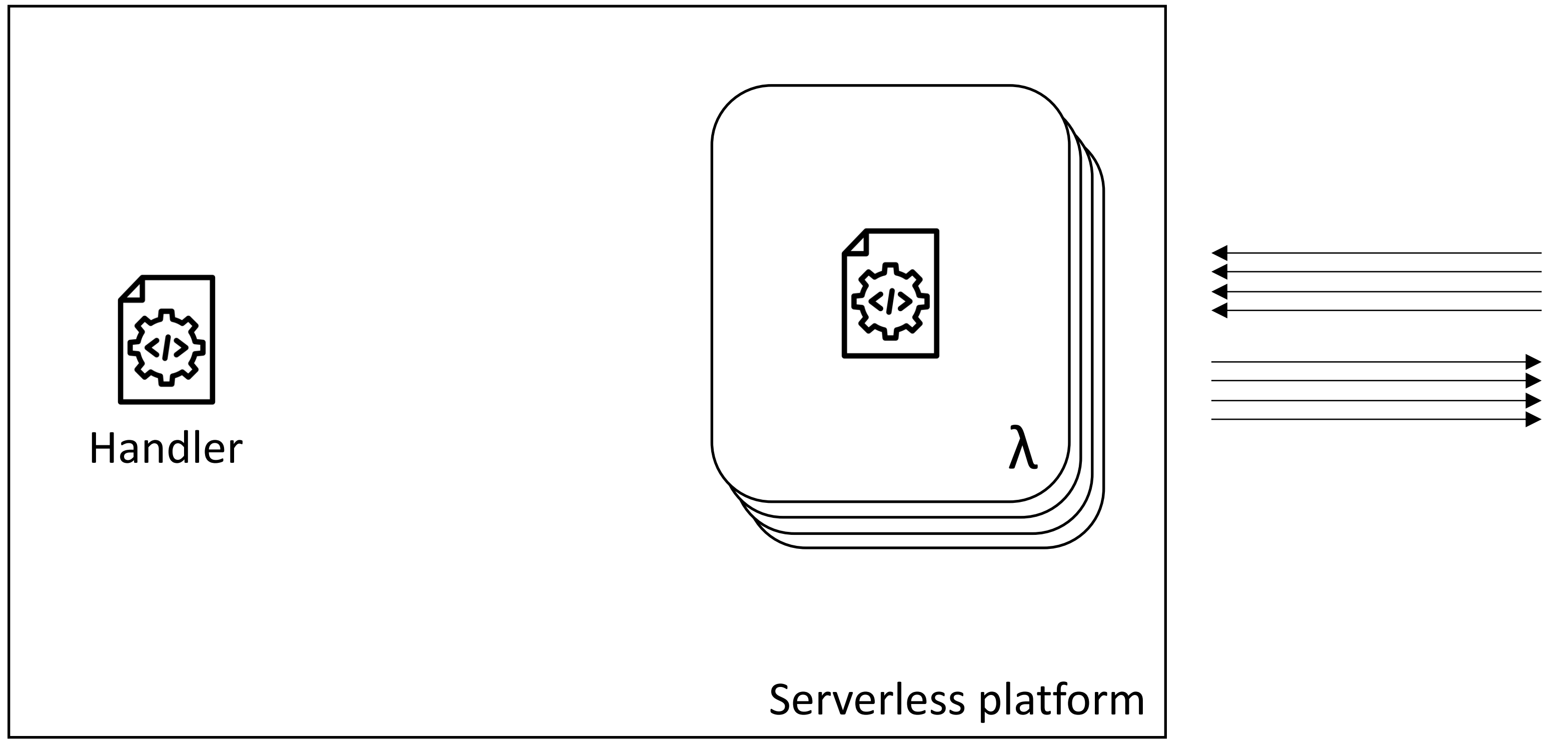
Aurojit Panda NYU

Scott Shenker UC Berkeley/ICSI

Serverless computing



Serverless computing



Serverless computing

λ

✅ Enjoys simpler autoscaling.
✅ Free from infra management.

- Stateless
- Short-lived

✅ Flexible resource management.

AWS Lambda   Azure Functions   Google Cloud Functions   IBM Cloud Functions

ExCamera (NSDI ‘17)
Sprocket (SoCC ‘18)

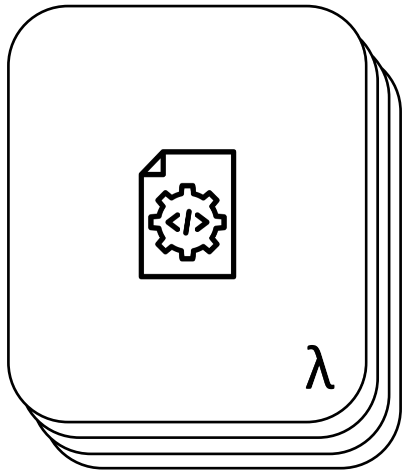
PyWren (SoCC ‘17)
Locus (NSDI ‘19)

gg (ATC ‘19)

Cirrus (SoCC ‘19)

NumPyWren (SoCC ‘20)

…



Our goal: Make serverless a **scalable substrate** for **general-purpose computing**

Challenges in general serverless computing

Computation: can be long-running.

Serverless: lambda are time-limited.

Computation: has diverse concurrency patterns.

Serverless: lacks concurrency features.

Kappa: a Framework for Serverless Computing

Checkpointing
(based on continuations)

Concurrency API
(futures, message-passing)

Fault tolerance
(in face of nondeterminism and side effects)

Requires **no modification to the serverless platform**.

Kappa enables diverse serverless applications

Big-data queries

Streaming analytics

Web crawling

Opens up possibility for many more applications on serverless!

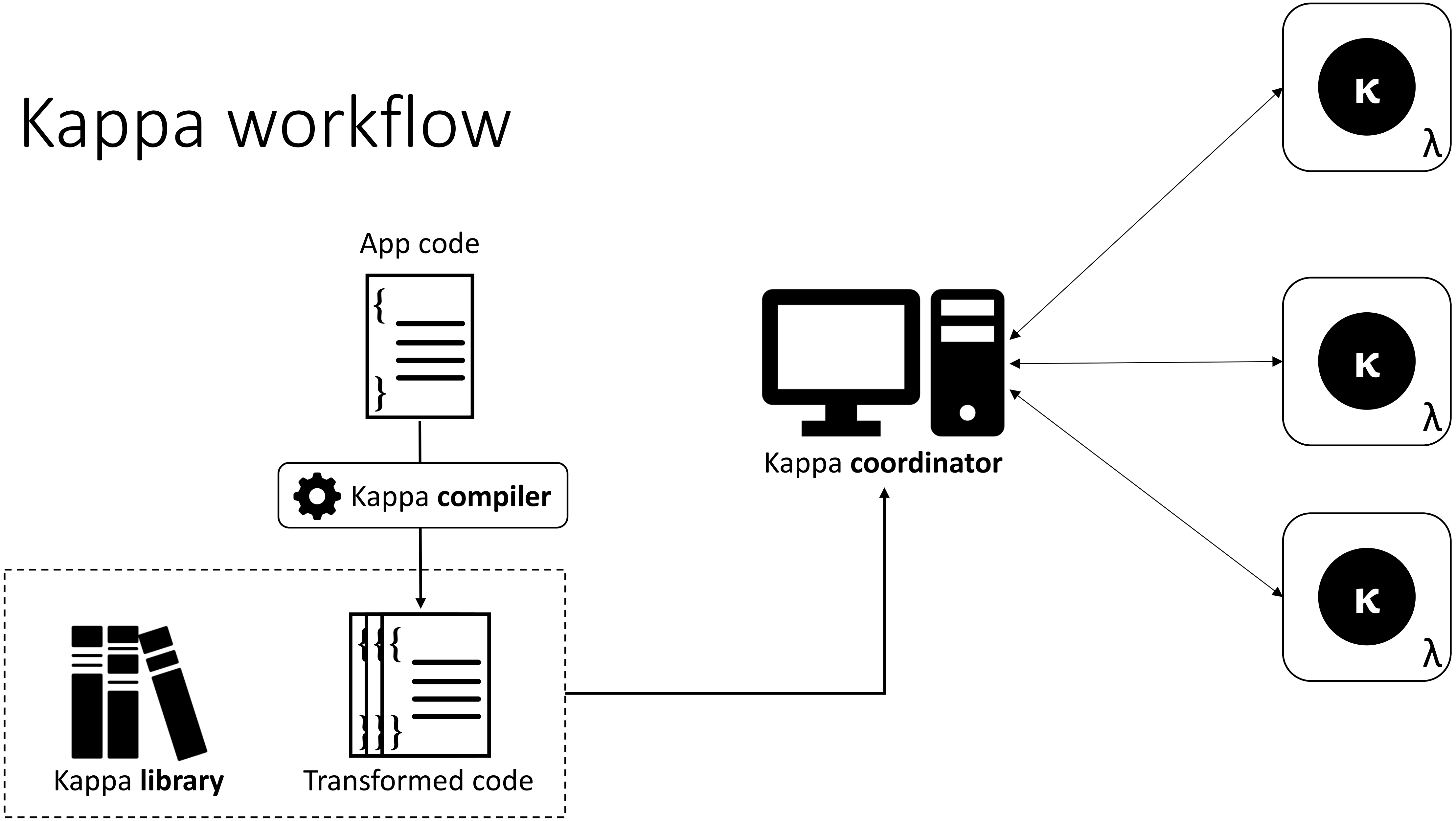
Kappa Design

Compiler

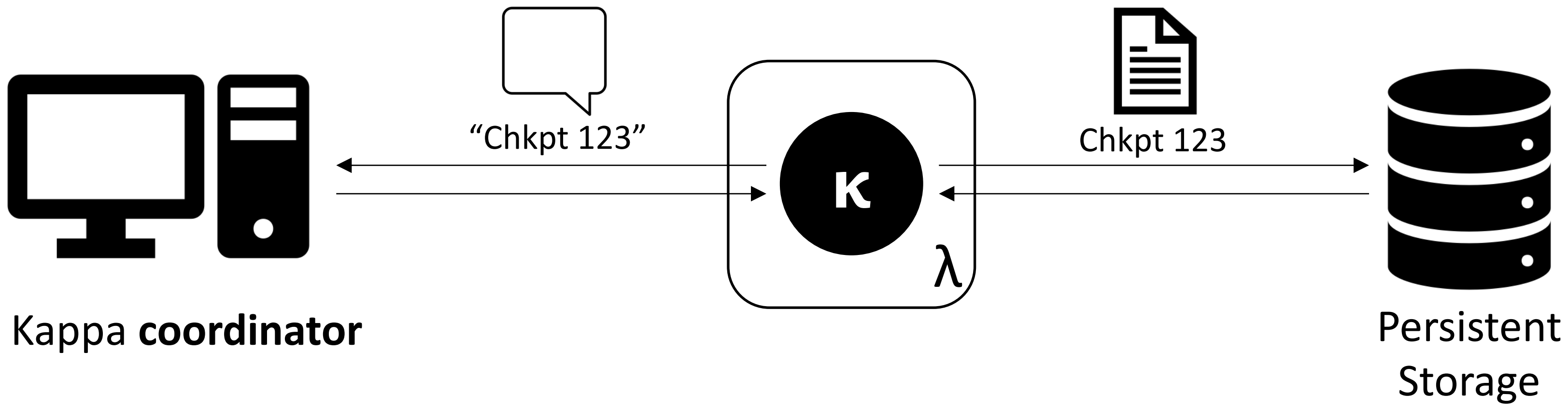
Coordinator

Library

Kappa workflow



Checkpointing



Checkpoint construction with continuations

- Language-level mechanism executed entirely in **user mode**.
- Programmer inserts `checkpoint()` calls:

```
def foo(x, y):
    z = x * y
    checkpoint()
    t = z + x
    return t
```

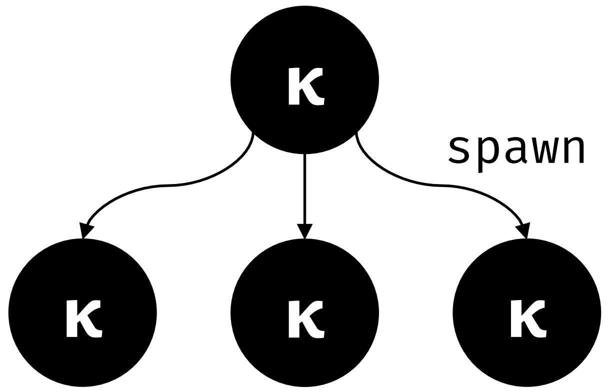
Compiler

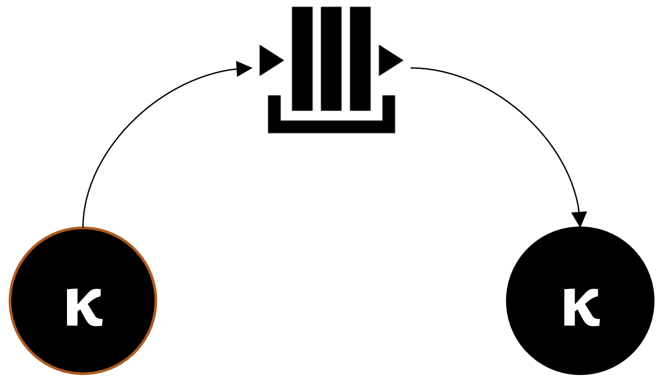
```
def cont_foo(z, x):
    t = z + x
    ...
```

- Checkpoint for this frame looks like: $\langle \text{cont\_foo}, z = 3, x = 4 \rangle$.
- Supports function calls, conditionals, loops, etc.

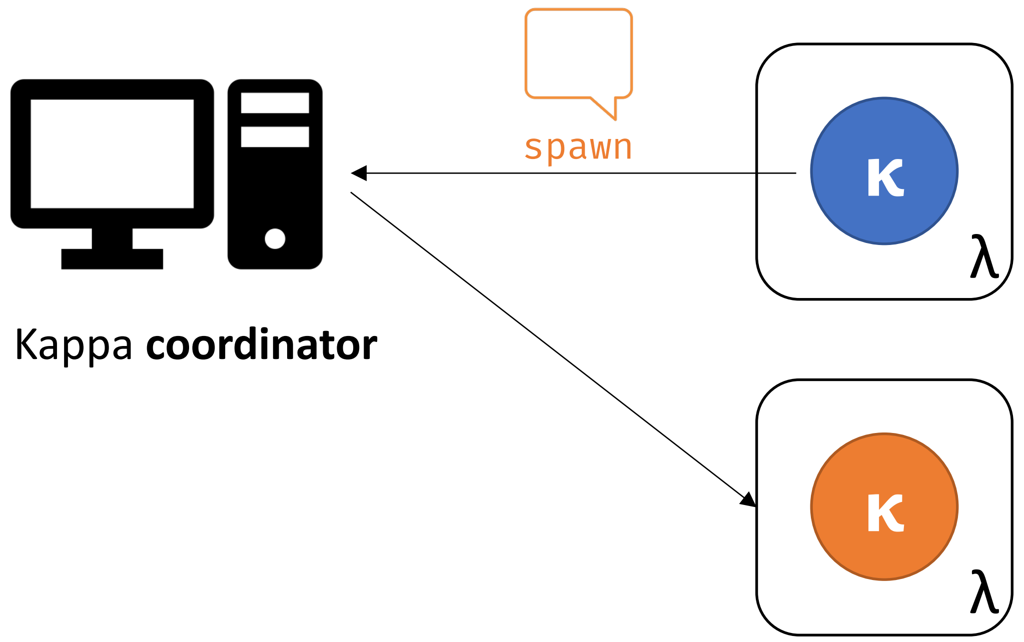
Concurrency API

- **`spawn`** Kappa task to compute in parallel; **`wait`** for task result.
- FIFO **`queue`** for communication and synchronization.



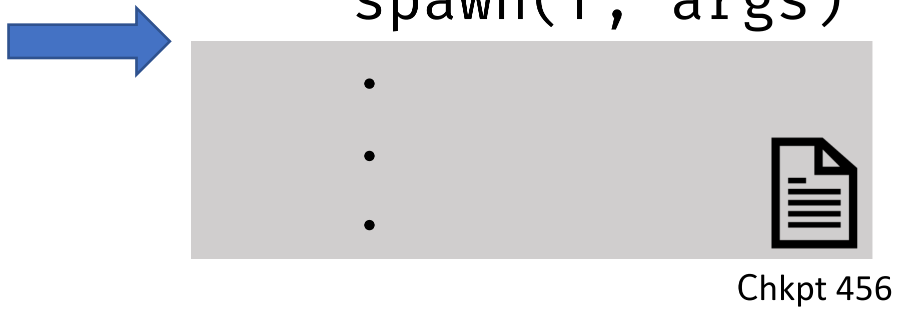


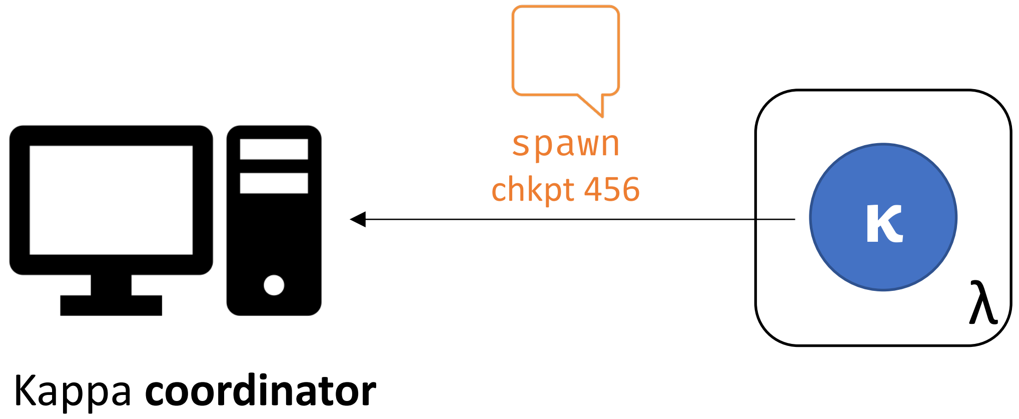
Fault tolerance for effectful operations



Fault tolerance for effectful operations

```
def foo():
    spawn(f, args)
    .
    .
    .
```





Other features

Coordinator state persistence

Checkpoint replication

External services

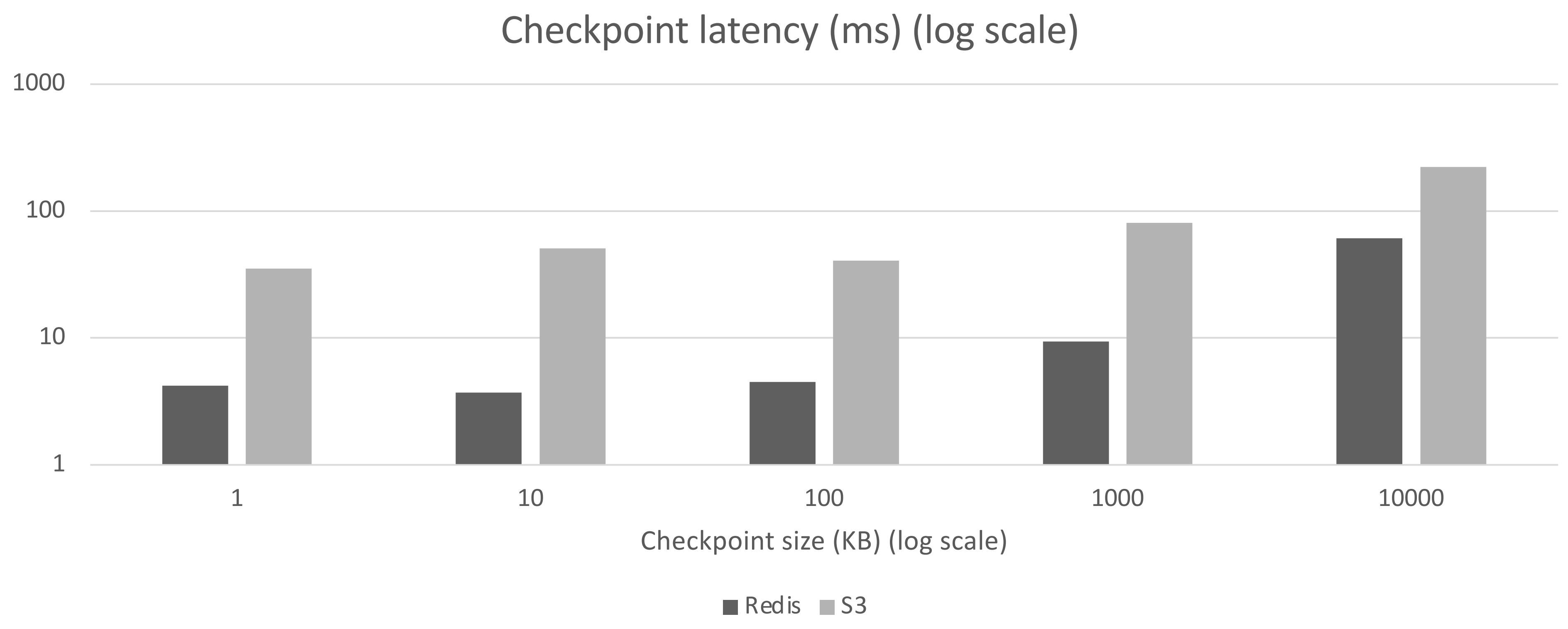
Evaluation

- How much overhead is added by Kappa’s checkpointing?
- Is the Kappa API general enough to support diverse applications on serverless?

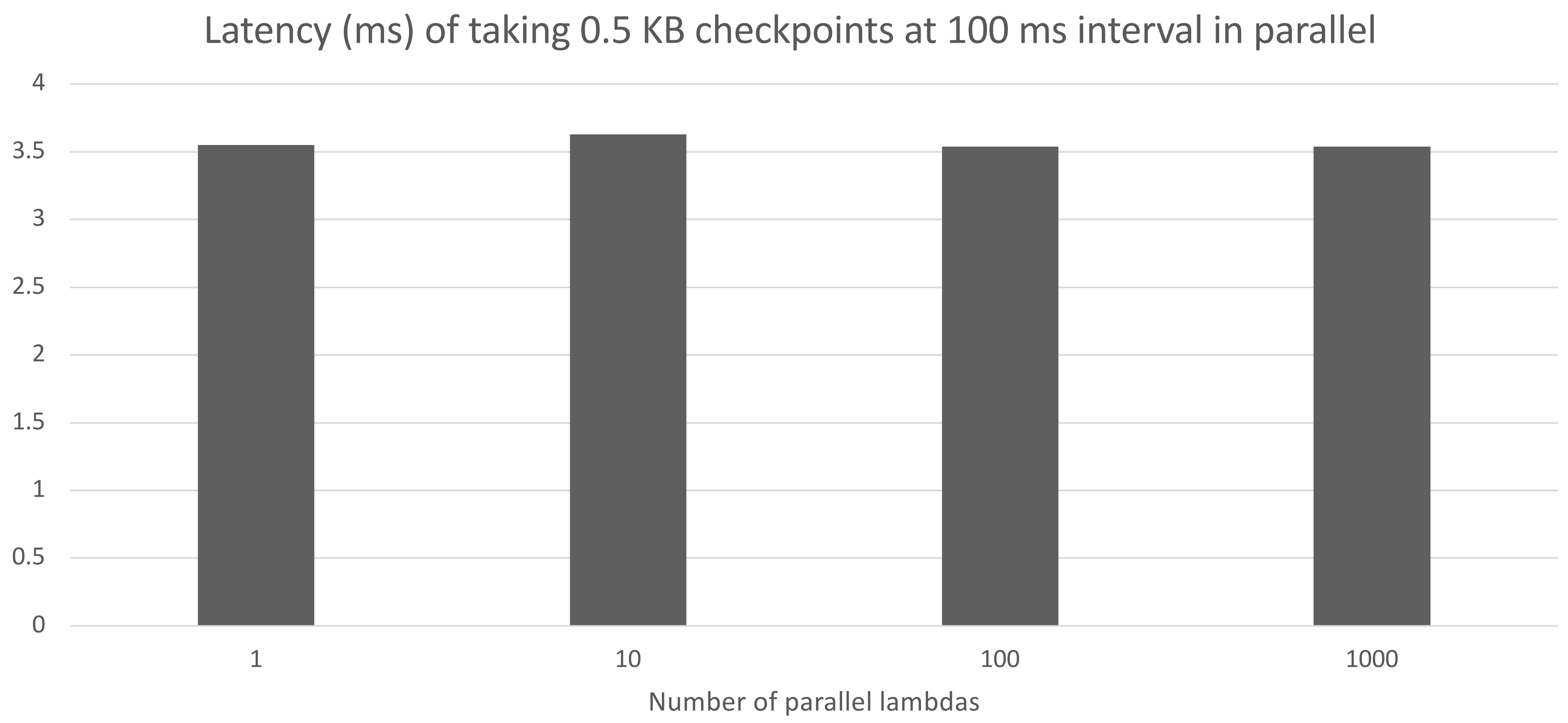
Experiment setup

- AWS Lambda with maximum lambda memory (3008 MB).
- Coordinator runs on Amazon EC2 instance (m5.4xlarge).
    - Coordinator state replicated to two Redis instances.
- Checkpoints are stored in Redis (2-way replicated).
    - Unless otherwise specified.

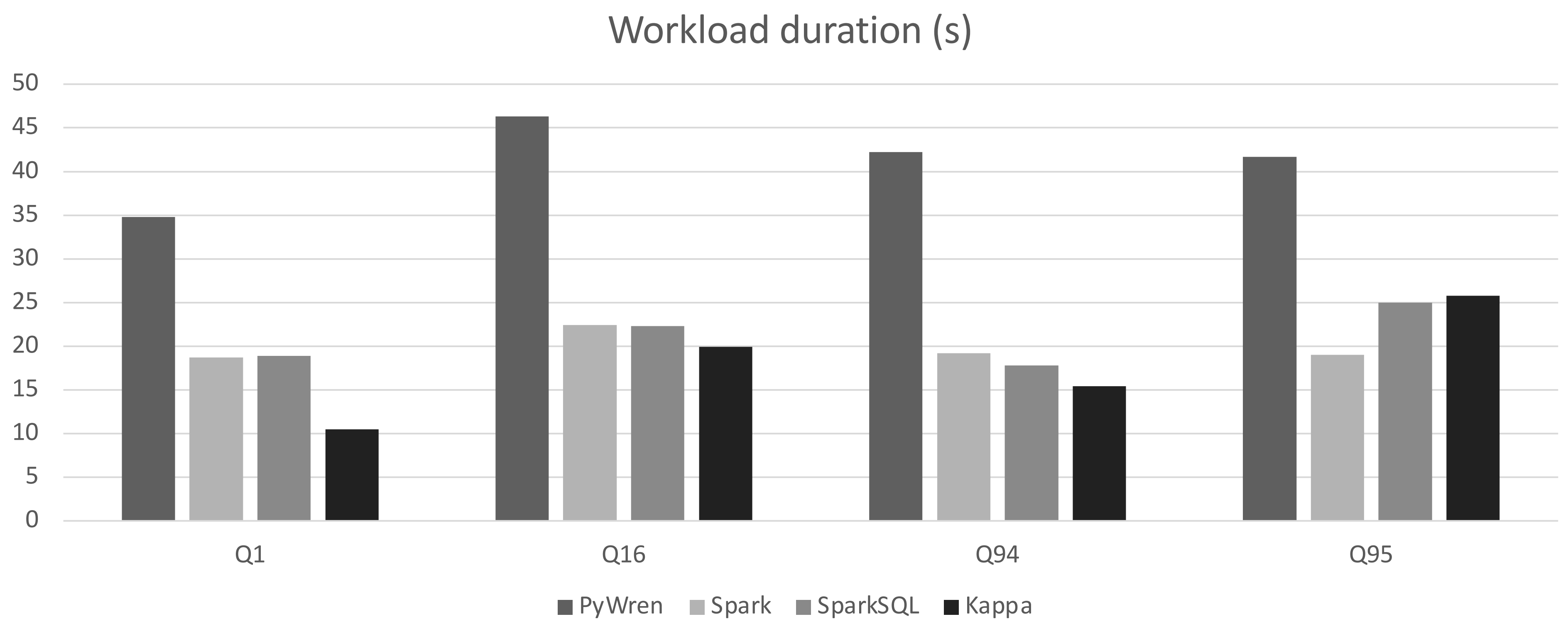
Checkpointing is fast



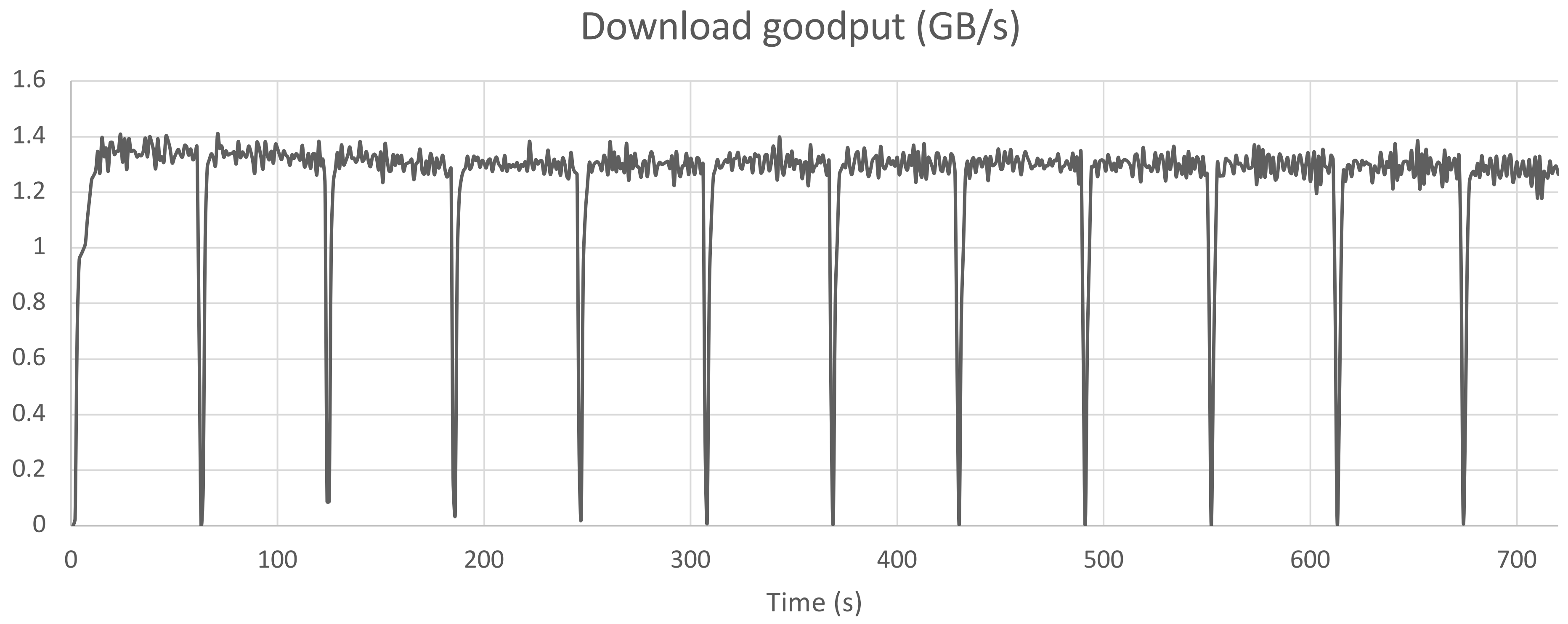
Checkpointing is scalable



TPC-DS queries



Concurrent web crawler



Thank you!

https://kappa.cs.berkeley.edu