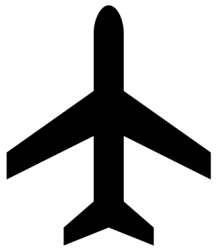
AFT: A Serverless Fault-Tolerance Shim

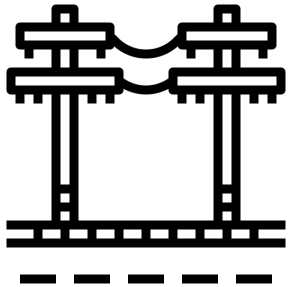
Vikram Sreekanti, Chenggang Wu, Saurav Chhatrapati,
Joseph E. Gonzalez, Joseph M. Hellerstein, Jose M. Faleiro
RISE Lab, UC Berkeley
04/29/2020

Fault-Tolerance in Serverless Computing

- FaaS programs with shared state raise concerns about faults



What happens when functions fail mid-flight?



What happens when infrastructure fails between functions?

What is the contract with the user?

Semantic Goals for Stateful FaaS

- Understandable: **exactly-once** executions
- State of play for commercial FaaS: **at-least once** execution
  - Advice: Roll your own idempotence – difficult to reason about!

- But idempotence is not enough!
  - **Fractional** executions can leak partial side effects
- What else do we need? Atomicity!

Partial Executions: 0.5?

- Retries – even if idempotent – can expose partial executions
- Make some results of a function visible but not all

Request 1	Request 2
$W(A_1)$	
$W(B_1)$	
	R(A)
	R(B)

Partial Executions: 0.5?

- Retries – even if idempotent – can expose partial executions
- Make some results of a function visible but not all

Request 1	Request 2
$W(A_1)$	
ERROR	
	R(A)
	R(B)

Partial Executions: 0.5?

- Retries – even if idempotent – can expose partial executions
- Make some results of a function visible but not all

Request 1	Request 2
W(A_1)	
ERROR	
	R(A)
	R(B)

AFT: A Serverless Fault-Tolerance Shim

- Goal: Exactly-once transactions for FaaS with minimal code changes

- Design
  - Transparent fault-tolerance for FaaS runtimes
  - Implements new protocols for **read atomic** isolation

- Results
  - Low overheads compared to standard cloud deployments
  - Highly scalable

The Bigger Picture

- Part of a broader stack in the RISE Lab: the Hydro Project
- Check out our long talk for more details!

hydro-project.github.io