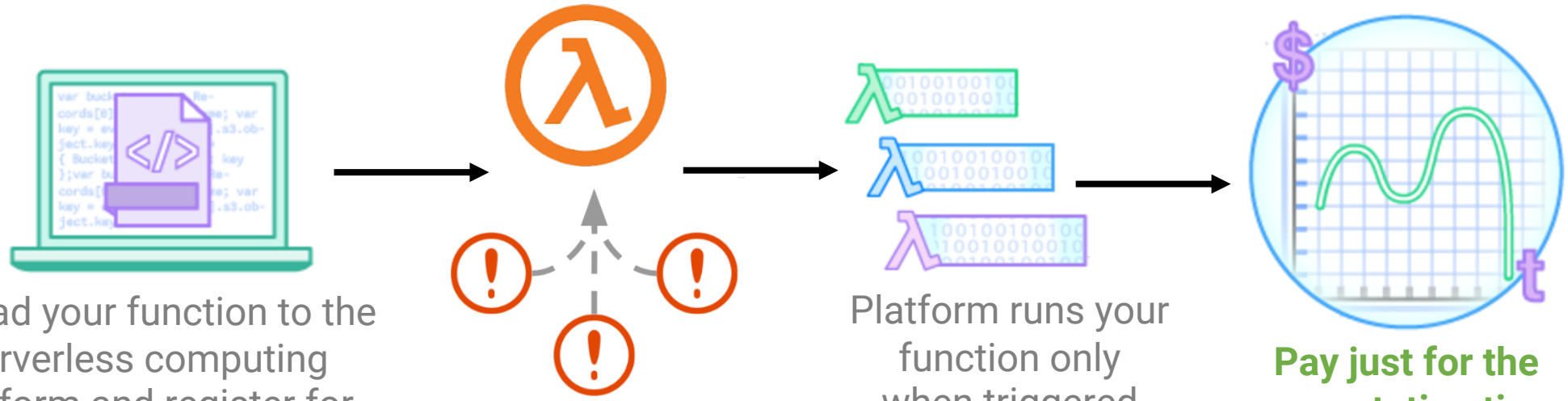
# Atoll: A Scalable Low-Latency Serverless Platform

**Arjun Singhvi**, Arjun Balasubramanian, Kevin Houck, Mohammed Danish Shaikh, Shivaram Venkataraman, Aditya Akella


# Serverless Computing 101



Upload your function to the serverless computing platform and register for event-based triggers

Execution is triggered when an event occurs

Platform runs your function only when triggered

**Pay just for the computation time**

**Automatic Scaling Support**

**Ideal Goal** → Ensure that function end-to-end latency is **close** to native execution time

# Characterizing Real World Serverless Apps

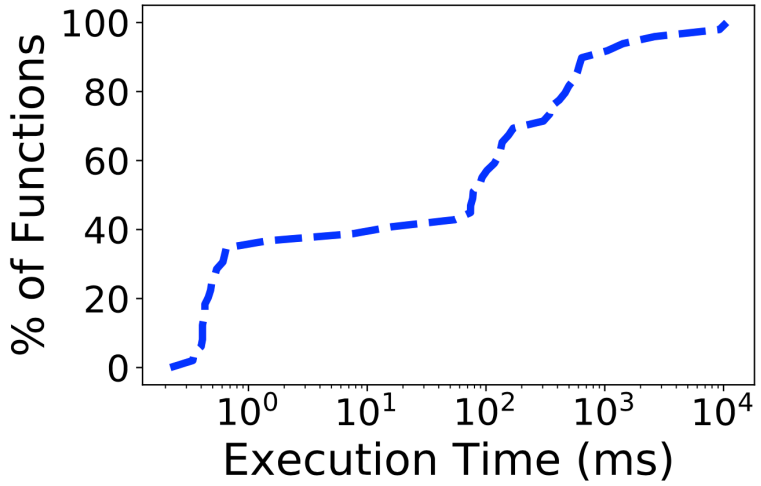
Looked at the **top 50 most deployed real world** apps in the AWS Serverless Application Repository

Benchmarked the apps by triggering their execution from a VM in the same region

Recorded statistics such as **provisioned memory, execution time, sandbox setup overhead** etc.

# Characterizing Real World Serverless Apps

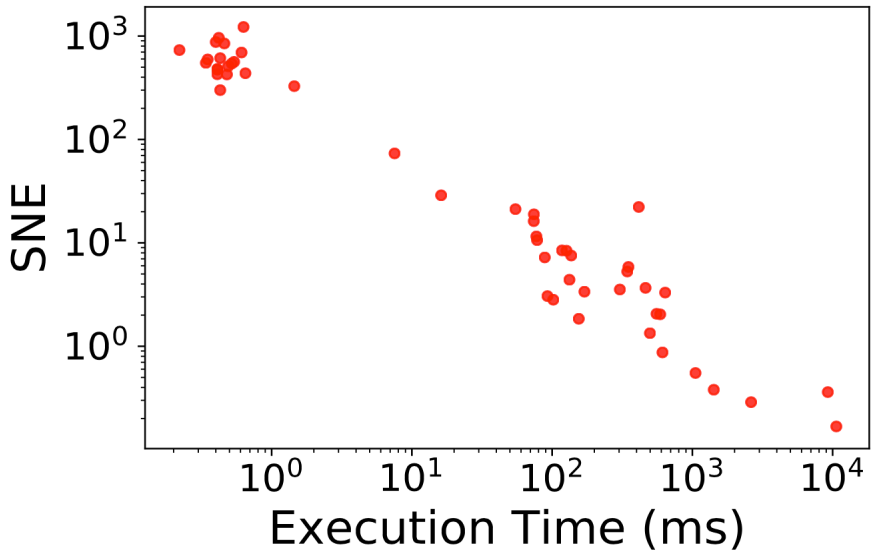
## [T1] **Functions** have a **range of execution times**



# Characterizing Real World Serverless Apps

**[T1] Functions** have a **range of execution times**

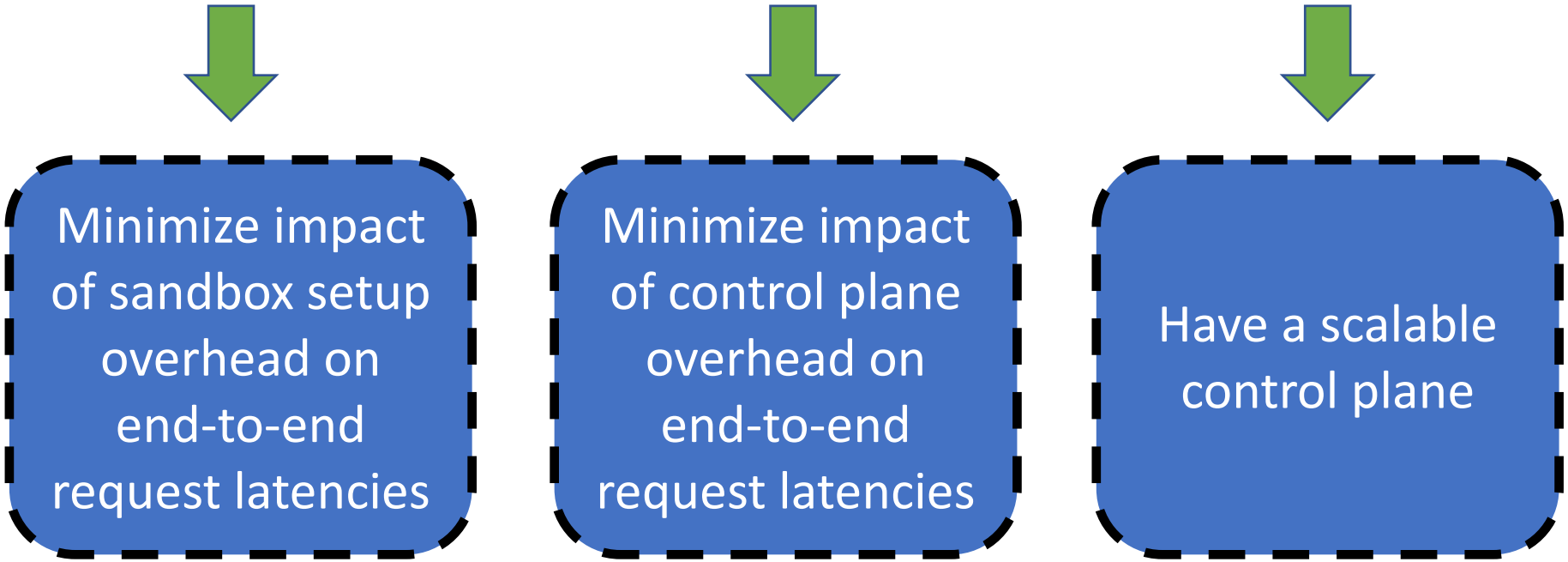
**[T2]** Sandbox **setup dominates execution times**



# Serverless Platform Requirements

Maximize number of requests whose end-to-end latency is close to native execution times

(deadline specified by end-user)



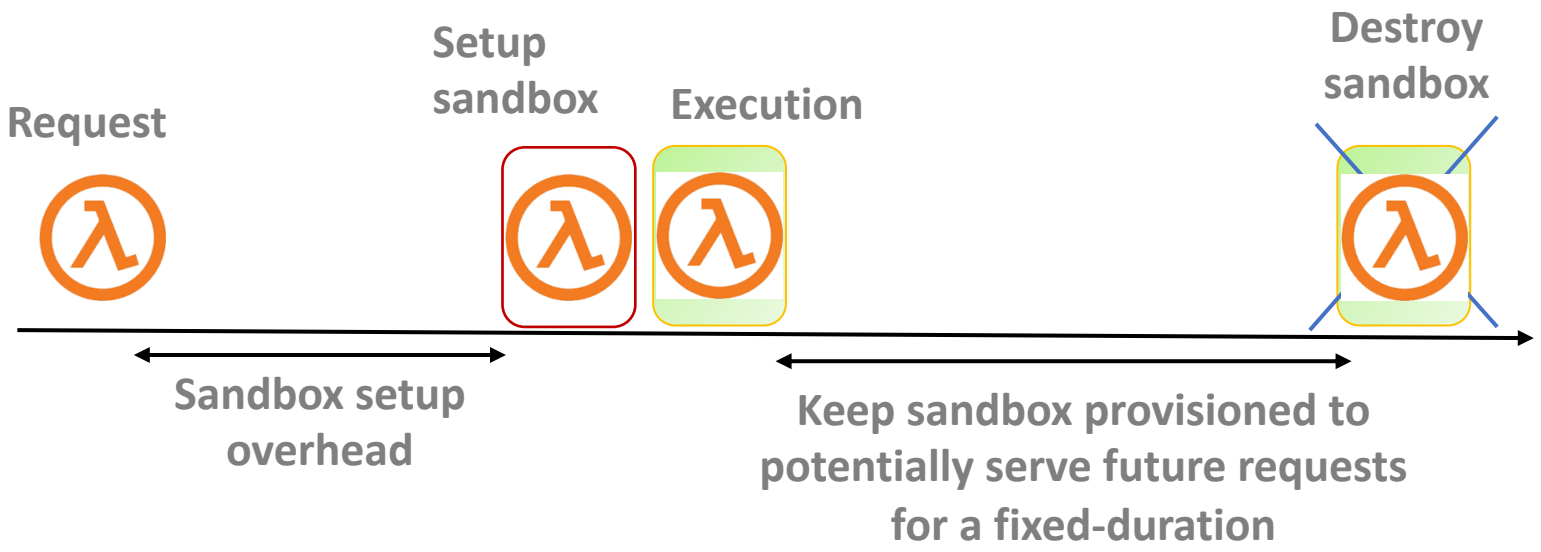
# Current Serverless Platforms - Issues

## Sandbox Management Policy

**Reactive** : setup sandboxes only when requests arrive

**Fixed and Workload Unaware** : keep sandbox around for fixed time

Leads to additional latency overheads or wasteful memory consumption



# Current Serverless Platforms - Issues

**Sandbox Management Policy**

**Reactive** : setup sandboxes only when requests arrive

**Fixed and Workload Unaware** : keep sandbox around for fixed time

Leads to additional latency overheads or wasteful memory consumption

**Sub-Optimal Scheduler Architecture**

**Centralized** approaches **do not scale**

**Decentralized** approaches **trade-off scheduling quality/predictability for scale**

**Homogeneous Request Handling**

Treat all requests in the **same manner**

**But** not all requests have strict latency requirements (have varying slack)

# Current Serverless Platforms - Issues

**Sandbox Management Poli**

**Reactive** : setup sandboxes o ve

**Fixed and Workload Unaware** d for fixed time

Leads to additional latency ov memory consumption

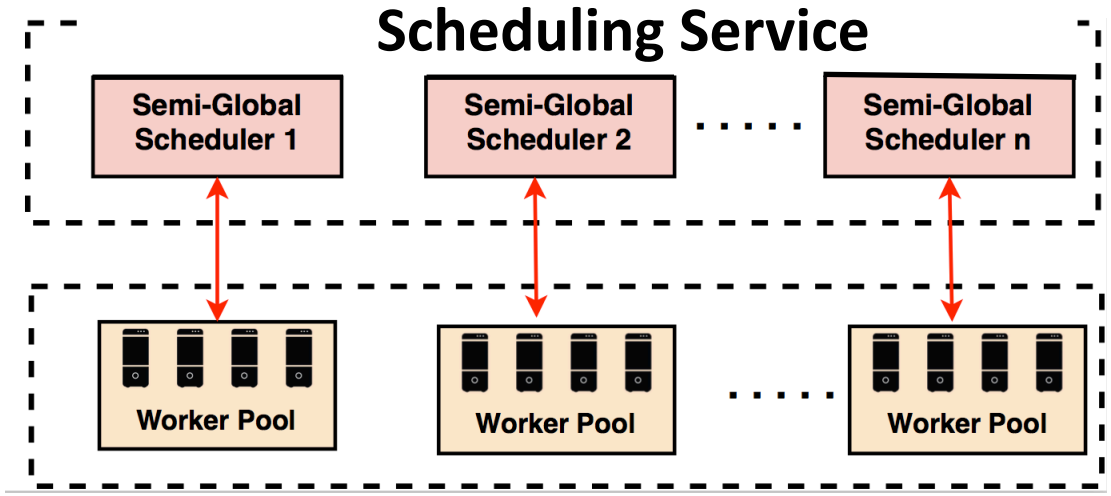
**Atoll** is a **scalable serverless platform** that enables **low latency request executions**

**Homogeneous Request Handling**

Treat all requests in the **same manner**

**But** not all requests have strict latency requirements (have varying slack)

# Atoll Design Overview : Key Idea #1

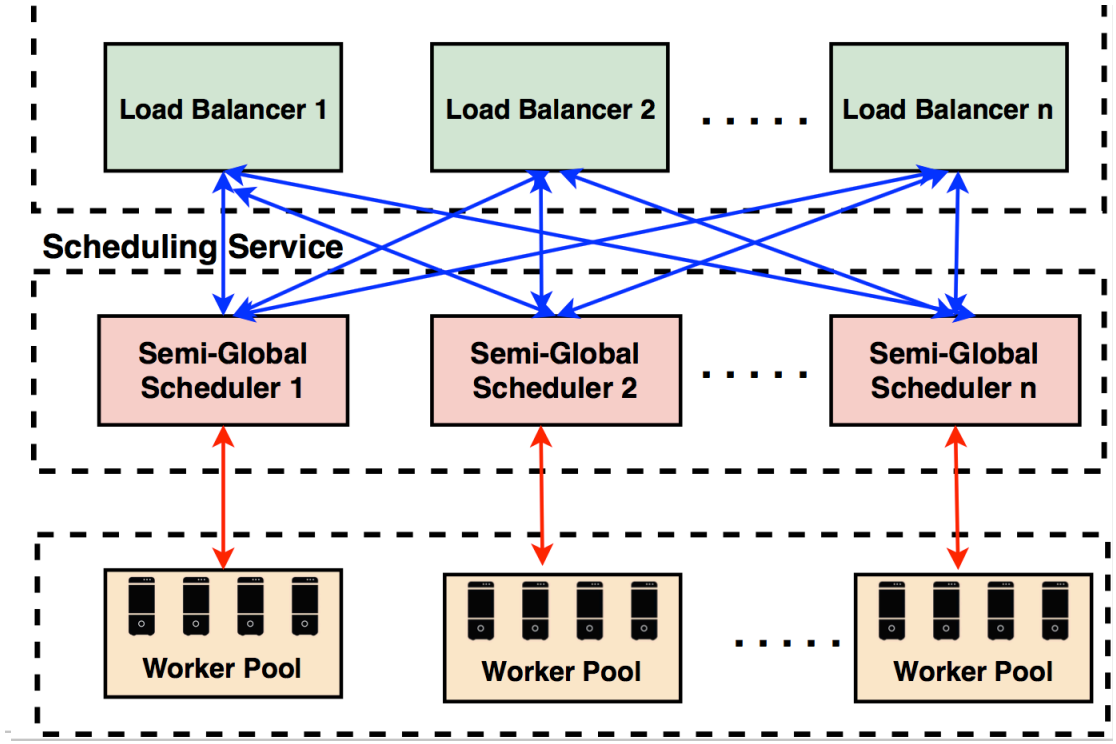



Cluster managed by **autonomous semi-global schedulers** (SGS)

Each SGS **exclusively** manages a partition of cluster machines – worker pool

**Ensures that schedulers don't become a bottleneck and yet make optimal decisions within their worker pool**

# Atoll Design Overview : Key Idea #2

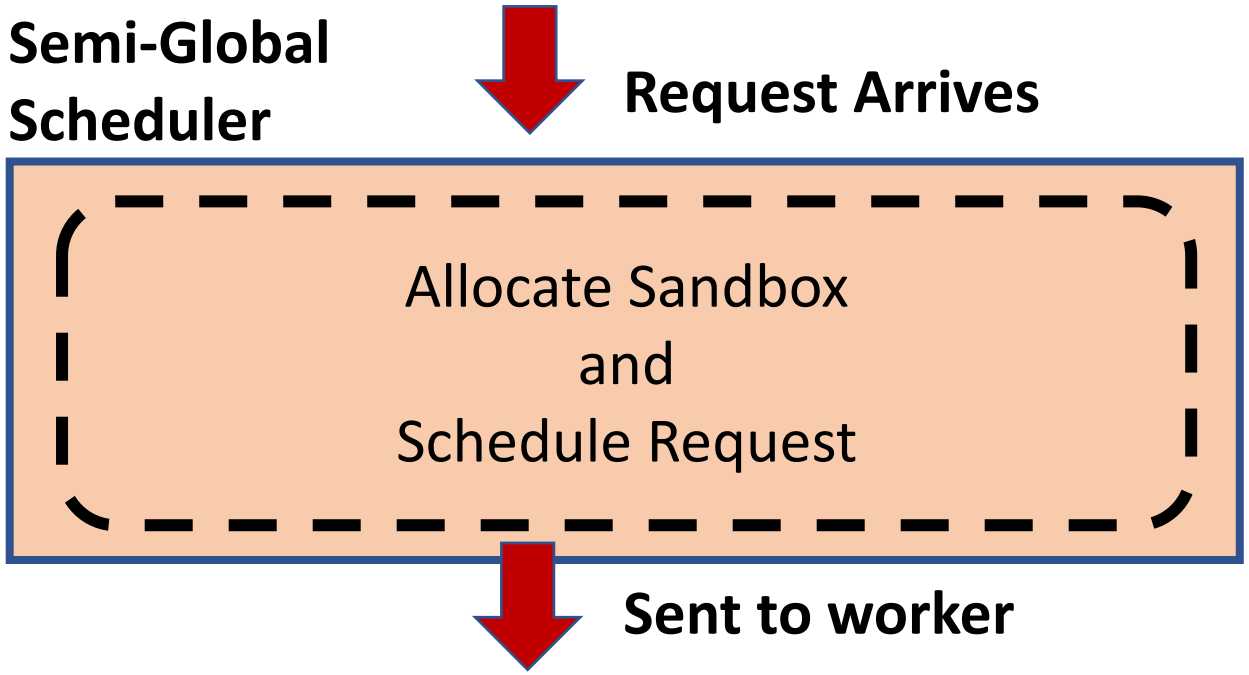



**Co-design** the load balancer and scheduling layers

**Provides the required visibility to ensure individual schedulers do not become hotspots**

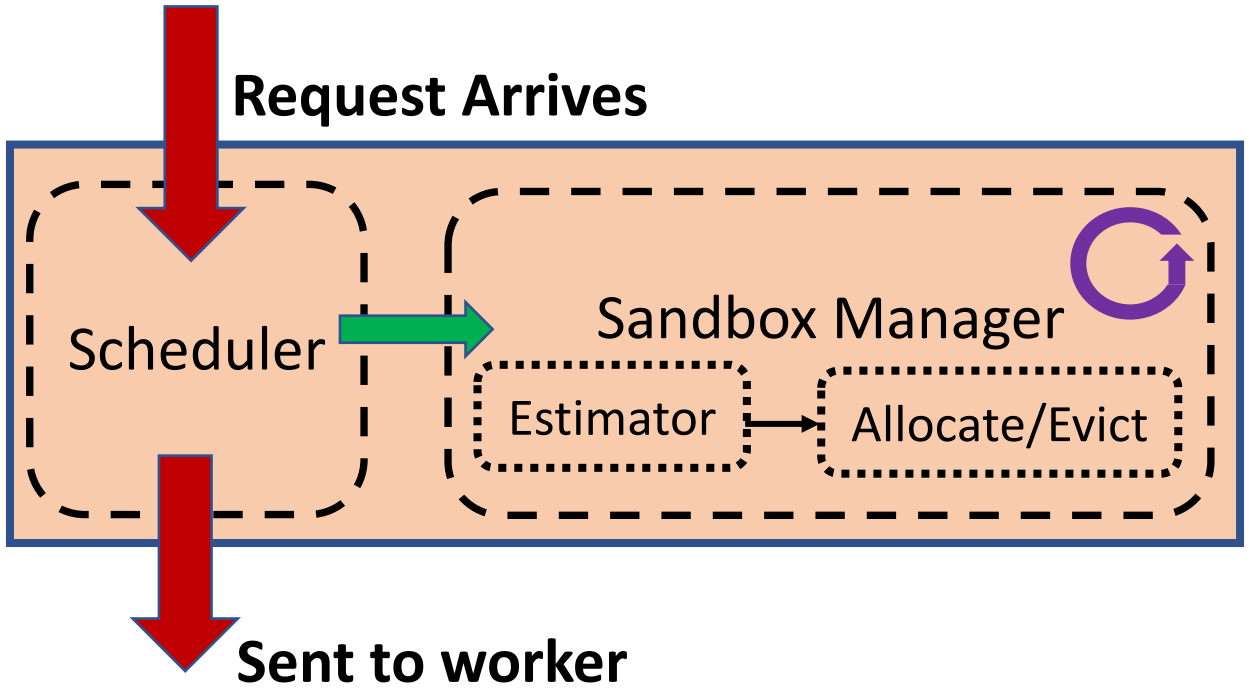
**Enables maximizing sandbox reuse leading to better latency performance**

# Atoll Design Overview : Key Idea #3




**Decouple** sandbox allocation from request scheduling

# Atoll Design Overview : Key Idea #3

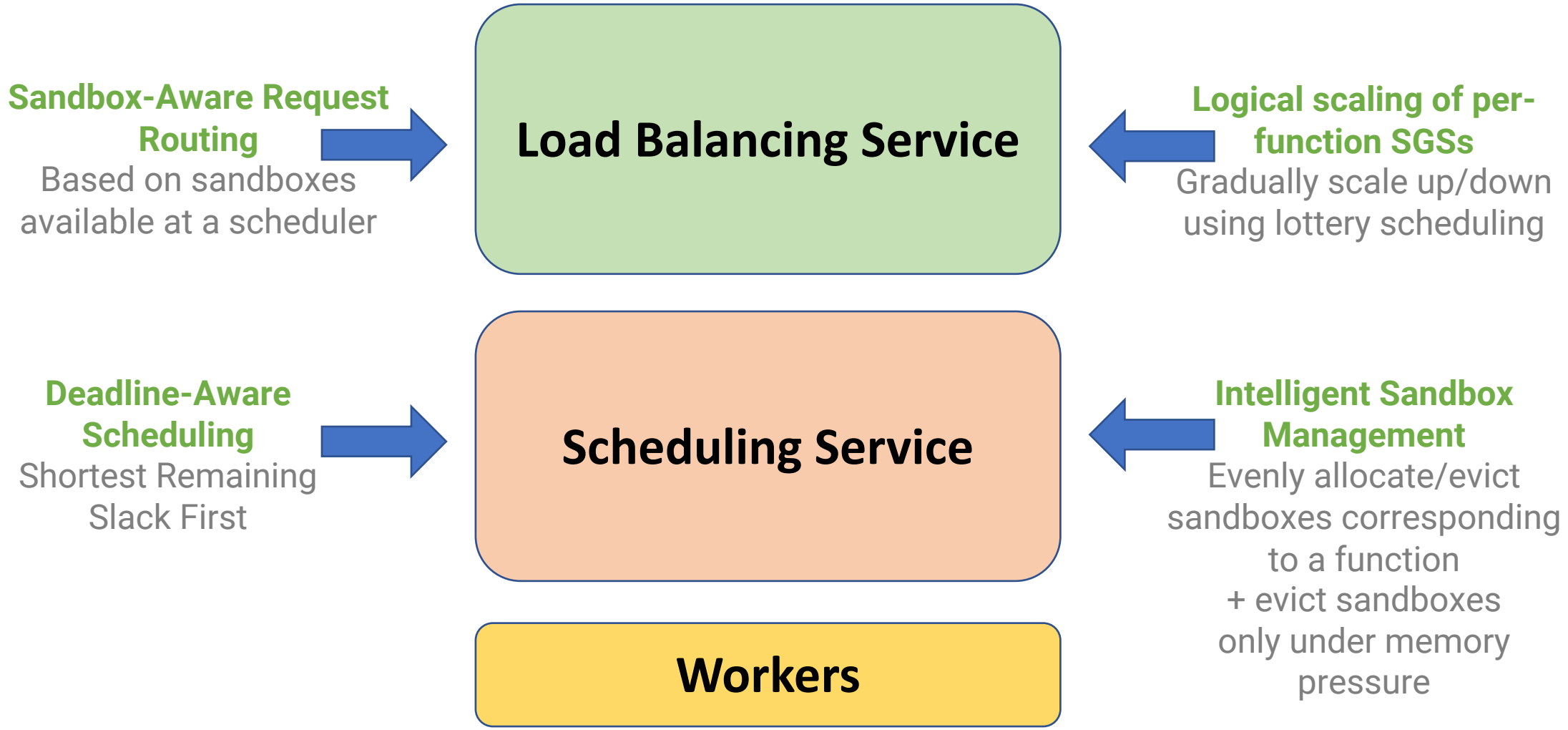



**Decouple** sandbox allocation from request scheduling

**Removes sandbox allocation from critical path**

**Enables proactive workload-aware sandbox allocation and eviction**

# Atoll Design Overview : Additional Details



# Atoll Evaluation : Implementation and Setup

**Prototype:** Built from scratch in Go

**Setup:** 74-machine cluster on CloudLab

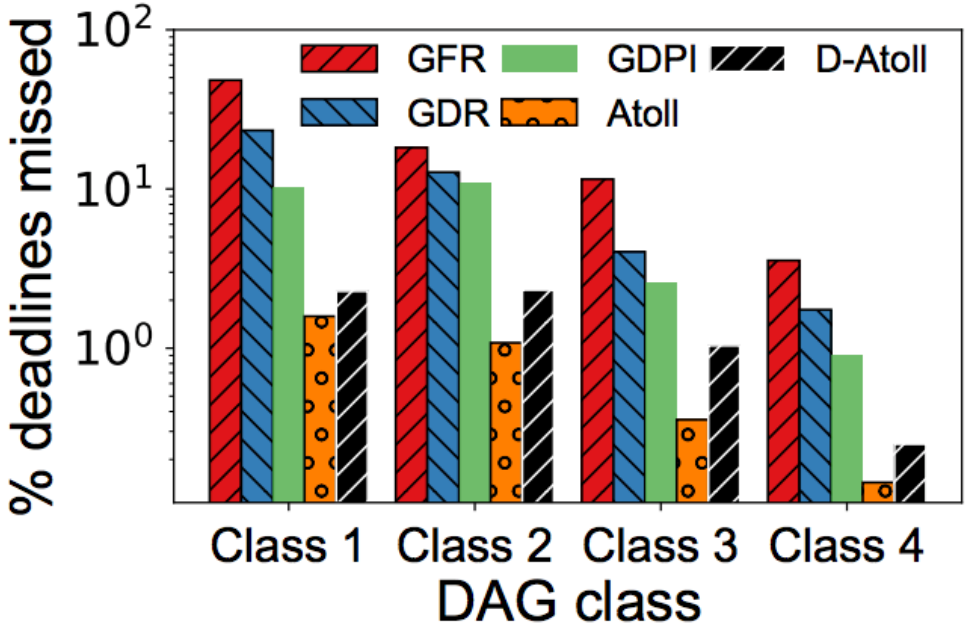
- 1 load balancer, 8 semi-global schedulers with each managing 8 machines

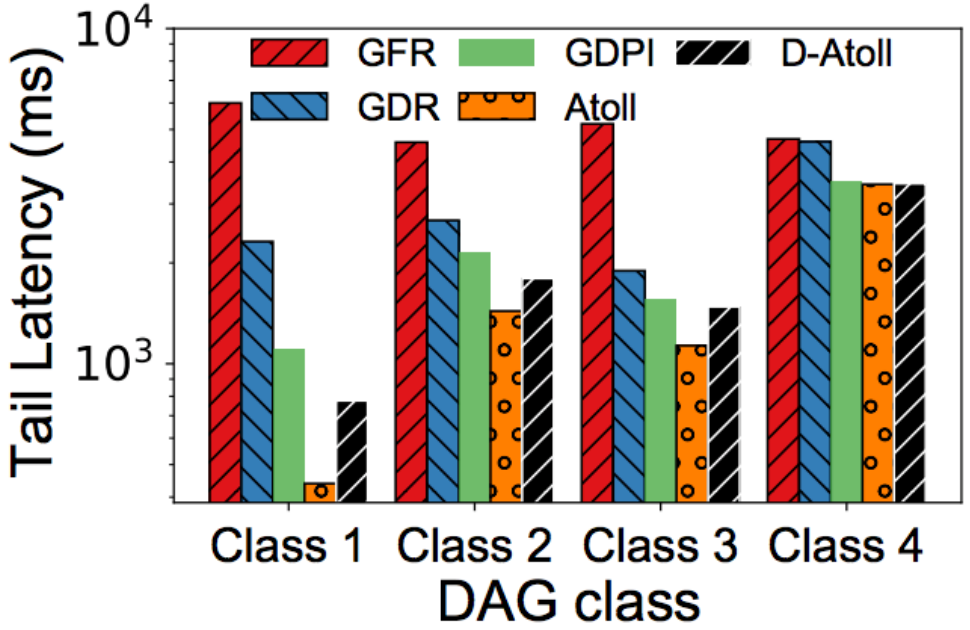
**Workload:** Mixture of DAGs that have varying execution times and deadlines and follow Poisson/sinusoidal/on-off arrival patterns

**Incremental Baselines:**

- **GFR** – **G**lobal View, **F**IFO Scheduler and **R**eactive Sandbox Allocation
- **GDR** – Replace FIFO Scheduler with **D**eadline-Aware Scheduler in GFR
- **GDPI** – Replace Reactive Sandbox Allocation with **P**roactive Sandbox Allocation and **I**nstant Eviction in GDR
- **Atoll** – Replace Instant Eviction with Soft Eviction in GDPI
- **D-Atoll** – Decentralized version of Atoll – SGS schedules requests from two randomly picked workers

# Atoll Evaluation : Atoll Vs Baselines




Atoll leads to **4.18x** better tail latencies and **26x** fewer deadlines missed over GFR

Deadline aware scheduling

Lesser cold starts

Semi-Global View

# Atoll Evaluation : Additional Highlights

Similar trends hold true across a spectrum of sandbox allocation overheads

Atoll continues to provide benefits even under memory pressure

Evenly spreading sandboxes improves performance due to better multiplexing

Gradual scaling using sandbox-aware routing leads to lower latencies

# Atoll Summary

Atoll enables **low latency function execution**

**Partitions** cluster into **small number of worker pools**, with each being **managed by an SGS**

Uses **proactive sandbox allocation** and **deadline-aware scheduling** within an SGS

Uses **sandbox-aware routing** and automatically **scales SGSs per serverless app**

# Thank you!

asinghvi@cs.wisc.edu

# Atoll: A Scalable Low-Latency Serverless Platform

**Arjun Singhvi**, Arjun Balasubramanian, Kevin Houck, Mohammed Danish Shaikh, Shivaram Venkataraman, Aditya Akella