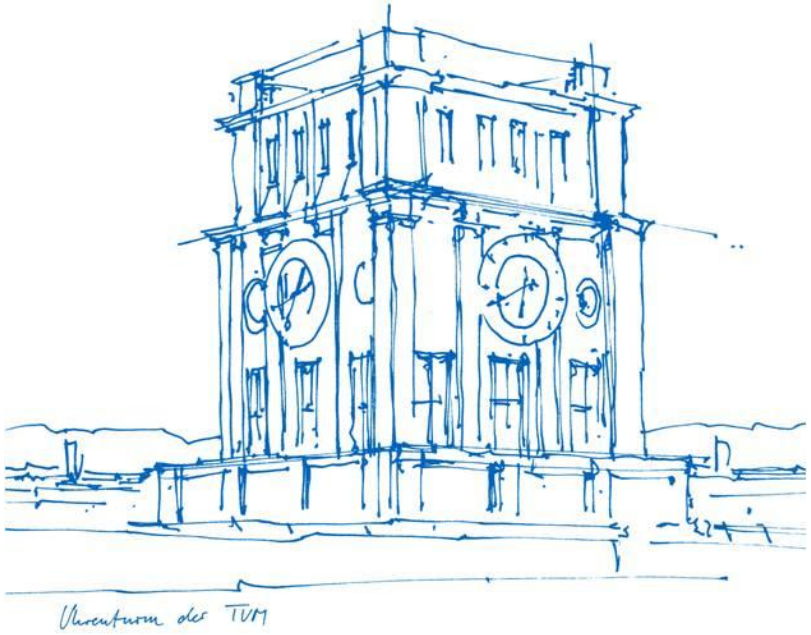
Chair of Computer Architecture and Parallel Systems
TUM Department of Informatics
Technical University of Munich

Data-Aware Function Scheduling on a Multi-Serverless Platform

Master Thesis Defense



Author: Christopher Peter Smith

Supervisor: Prof. Dr. Michael Gerndt

Advisor: M.Sc. Anshul Jindal

Munich, 10. February 2022

Outline

1. Motivation
2. Introducing FaDO
3. Implementation
4. Challenges
5. Results
6. Future Work

Outline

1. Motivation
2. Introducing FaDO
3. Implementation
4. Challenges
5. Results
6. Future Work

Motivation

Function-as-a-Service

Executing modular code in serverless environments.

Advantages

- Focus on the code
- Simplified Deployment
- Automatic Scaling
- Pay as you go

Disadvantages

- Less control
- Data locality
- Heterogeneous / multi-cloud setups

Motivation

Previous Works

The Function Delivery Network [1]

- Function-Delivery-as-a-Service

Master Thesis by Lucas Possani [2]

- Performance impact of data-locality

[1] Jindal et al., Function delivery network: Extending serverless computing for heterogeneous platforms. *Softw Pract Exper*. 2021; 51: 1936 – 1963.

[2] L. R. Possani, Self-Adaptive Data Management for Heterogeneous FaaS Platform. Master Thesis. Technische Universität München, 2020.

Outline

1. Motivation
2. Introducing FaDO
3. Implementation
4. Challenges
5. Results
6. Future Work

Introducing FaDO

The Problem

We want to schedule functions AND distribute data across a heterogeneous platform.

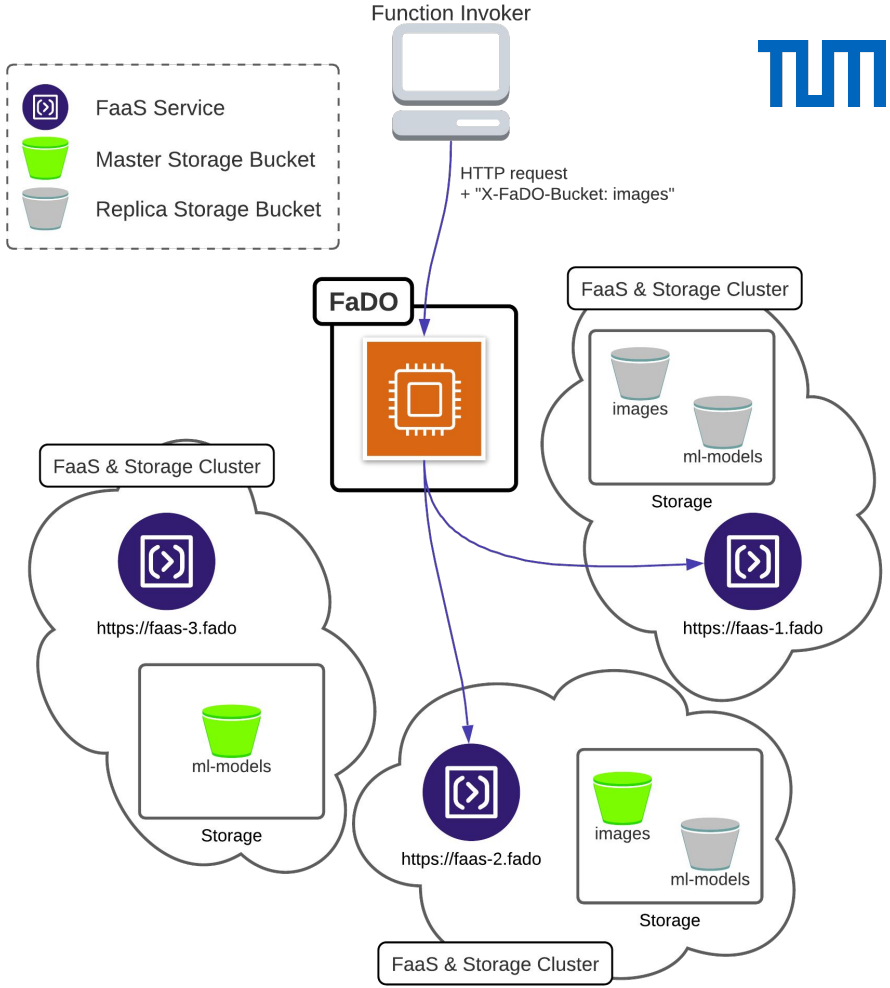
- Track colocated FaaS and storage services → Define “Clusters”
- Distribute data granularly → MinIO Bucket Replication
- Respect data constraints → Policy mechanisms
- Minimize function scheduling overhead → Request load balancing

Introducing FaDO

The Big Picture

The Function and Data Orchestrator

- Abstracts data access across the platform
- Distributes data through bucket replication
- Schedules functions close to their data
- Load-balances function invocations across the platform
- Configurable through user-defined policies

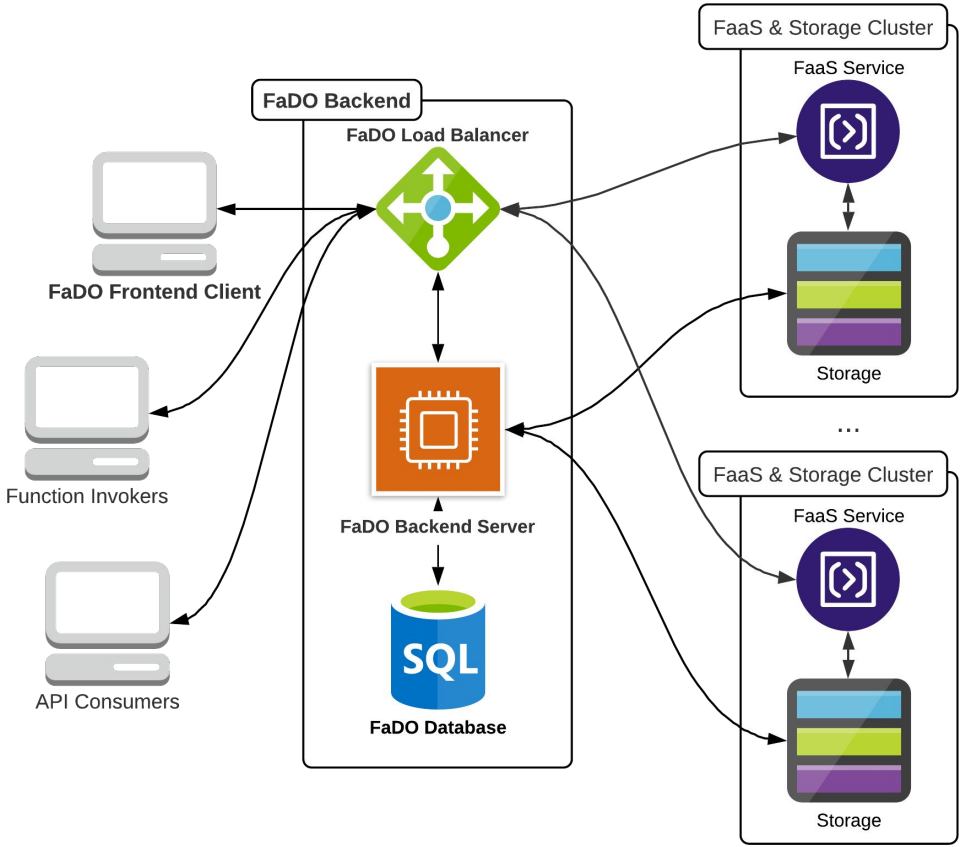


Introducing FaDO

System Architecture

FaDO has 4 distinct sub-components:

- A backend API server
- A PostgreSQL database
- A Caddy load balancer
- A frontend browser client

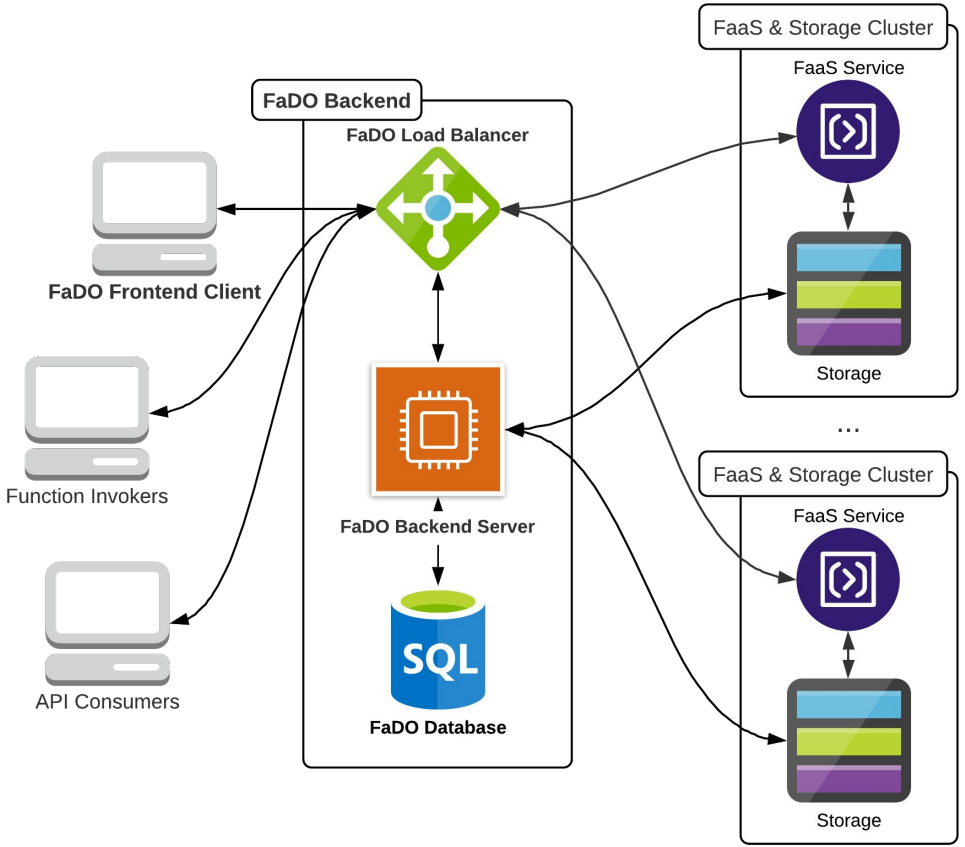


Introducing FaDO

System Architecture

FaDO has 3 major external interactions:

- API consumers
- MinIO storage services
- HTTP function invokers



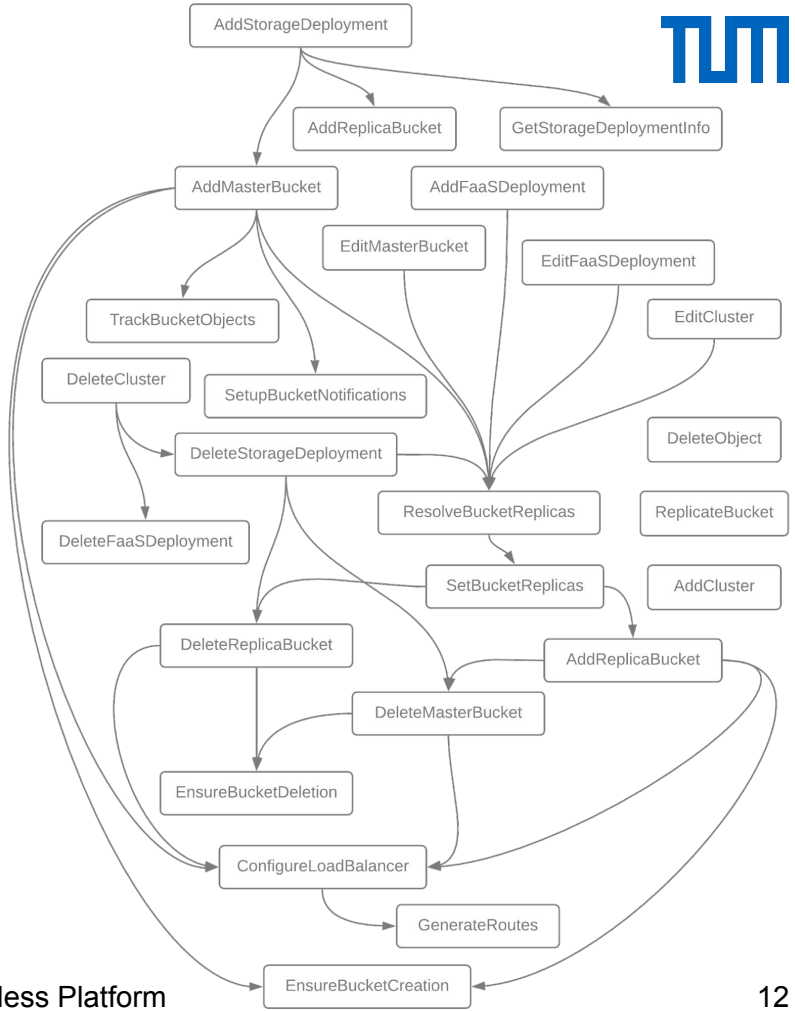
Outline

1. Motivation
2. Introducing FaDO
3. Implementation
4. Challenges
5. Results
6. Future Work

Implementation

Developing the Backend Server

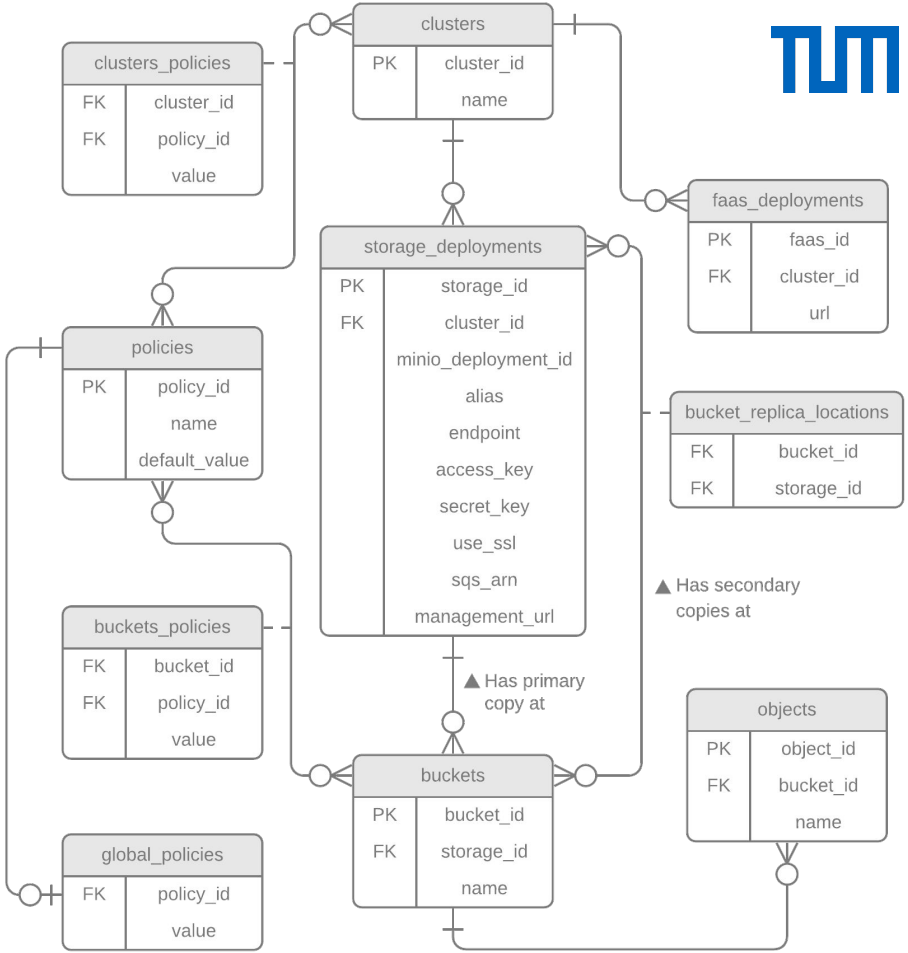
- Written in Go
- Communicates with all platform entities
- Orchestrates and reacts to changes
- Configures the load balancer
- Code organization is crucial



Implementation

Database Design

- Contains all application state
- PostgreSQL for reliability and performant queries on related data
- Tracks and organizes platform resources



Implementation

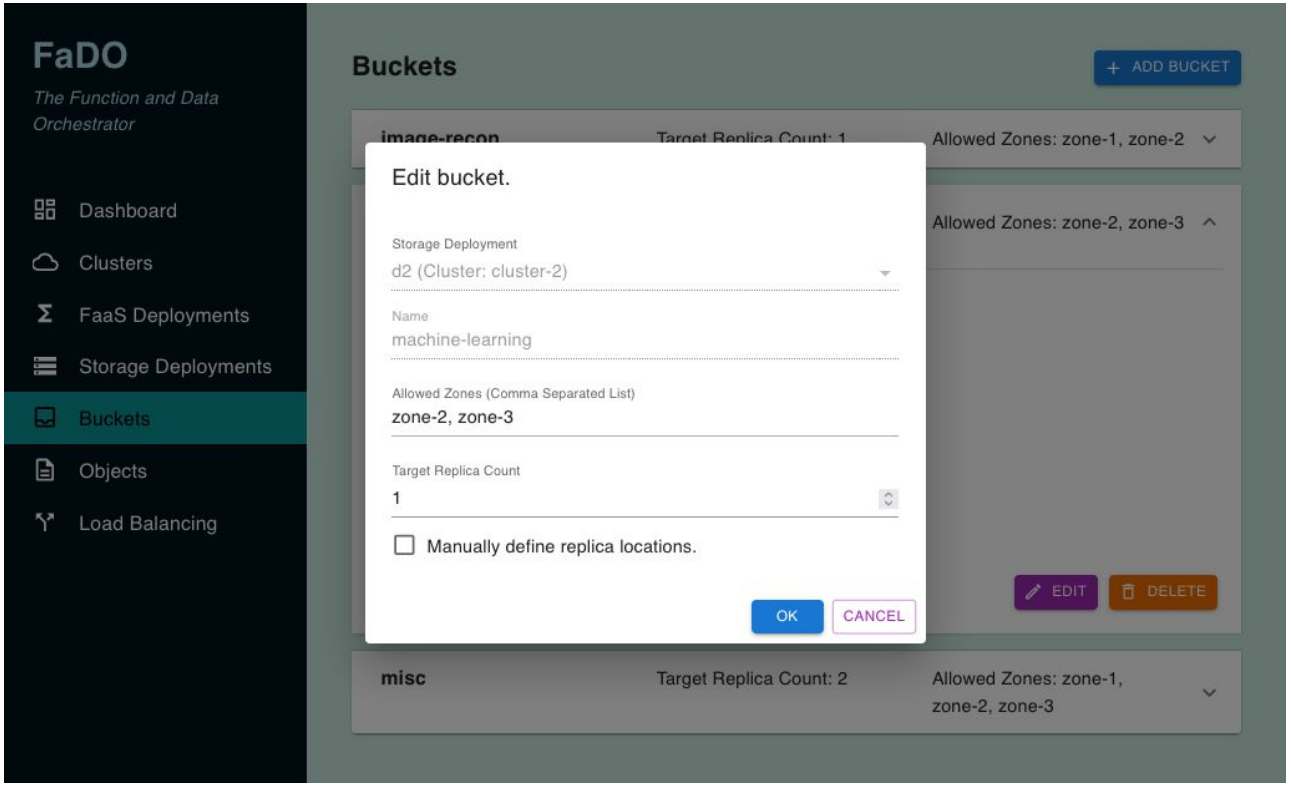
Integrating the Caddy Load Balancer

- Maps function requests to FaaS services
  - HTTP header matching
  - Load balancing policies
- Configurable using JSON/HTTP
- Provides TLS certificates automatically
- Written and extensible in Go

Implementation

The Frontend Client

Demonstration



Outline

1. Motivation
2. Introducing FaDO
3. Implementation
4. Challenges
5. Results
6. Future Work

Challenges

Challenges with the technology:

- MinIO
  - Replication features
  - Documentation
  - Libraries
- Caddy
  - Reconfiguration latencies
  - Administration endpoint failures

Challenges with the design:

- Backend server as a single point of failure
  - Replication Bottleneck
  - Vulnerable to external failures

Outline

1. Motivation
2. Introducing FaDO
3. Implementation
4. Challenges
5. Results
6. Future Work

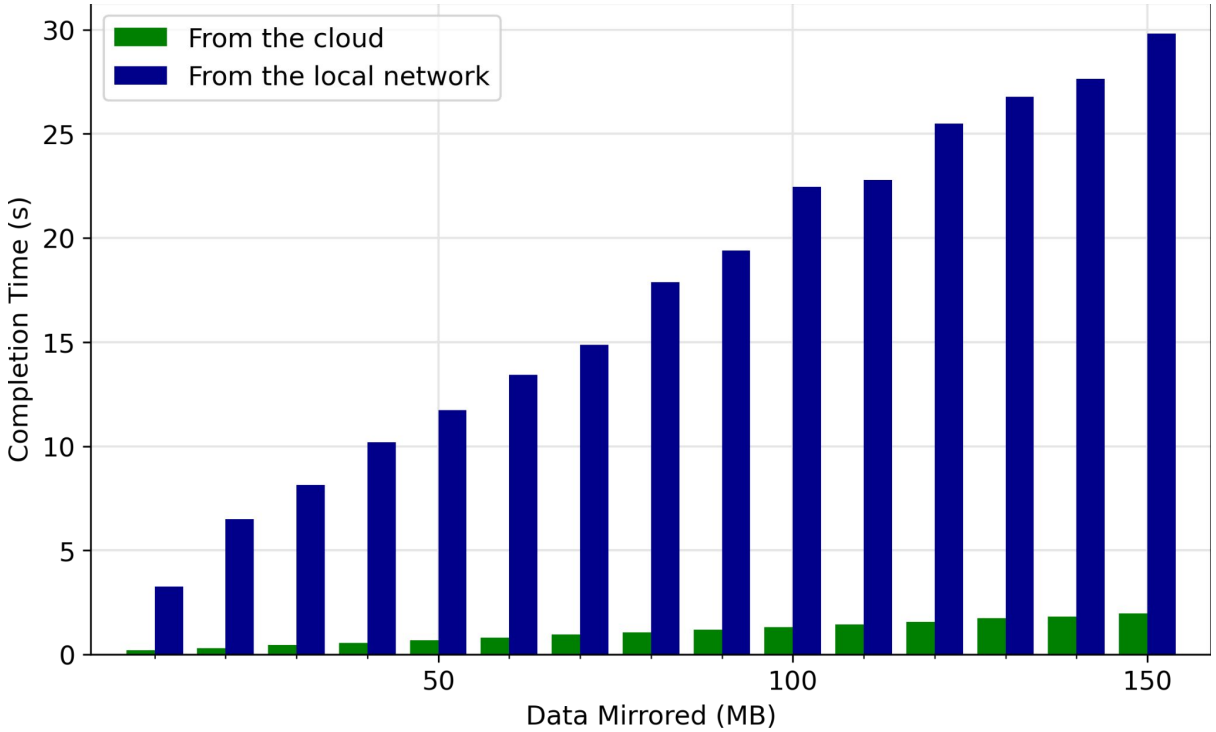
Results

Replication

Replicating data between two MinIO instances in the cloud.

The backend server becomes:

- A single point of failure
- A transfer bottleneck

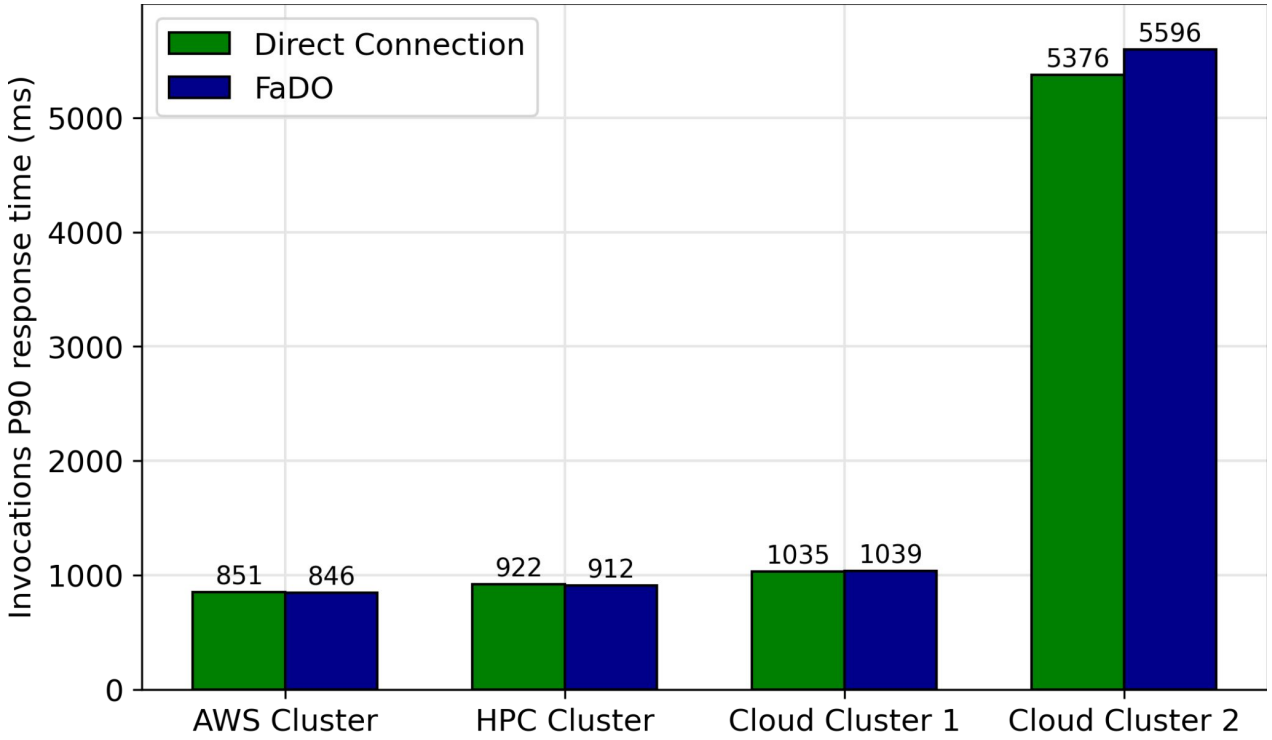


Results

Scheduling Overhead

Load testing FaaS servers directly and through FaDO.

FaDO's performance impact is negligible.

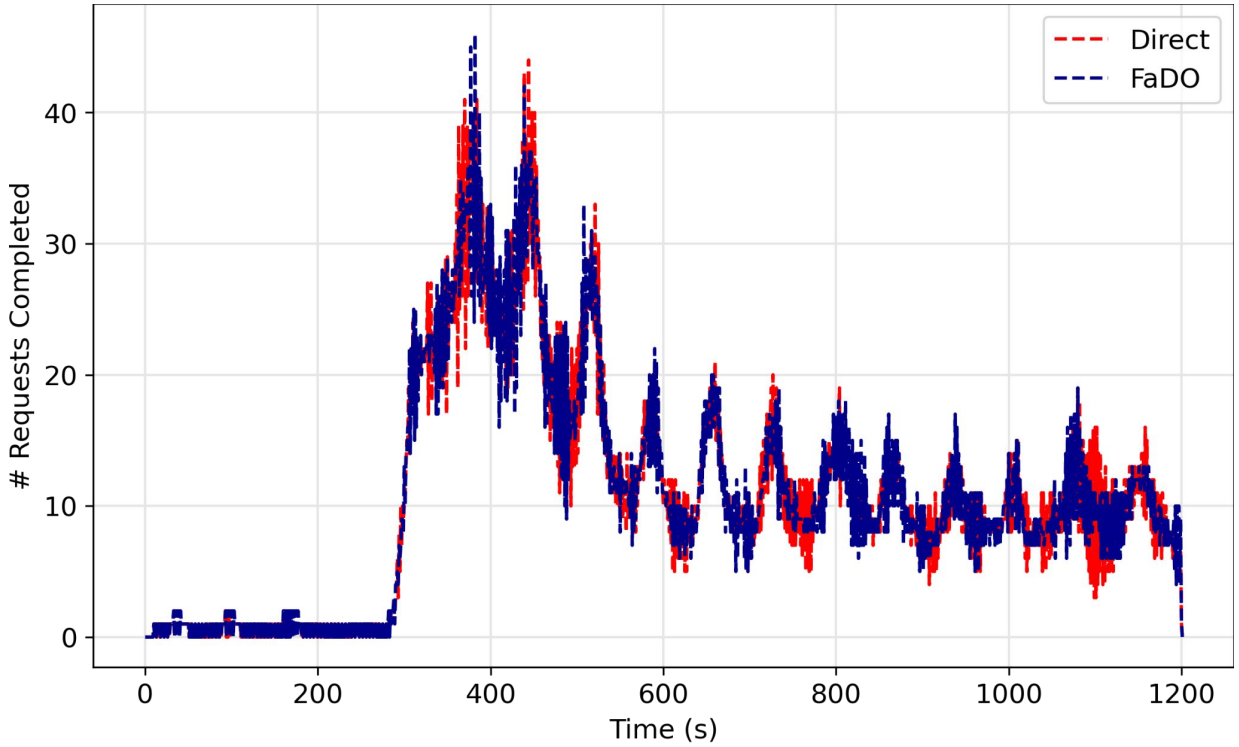


Performance

Scheduling Overhead

Load testing FaaS servers directly and through FaDO.

FaDO's performance impact is negligible.

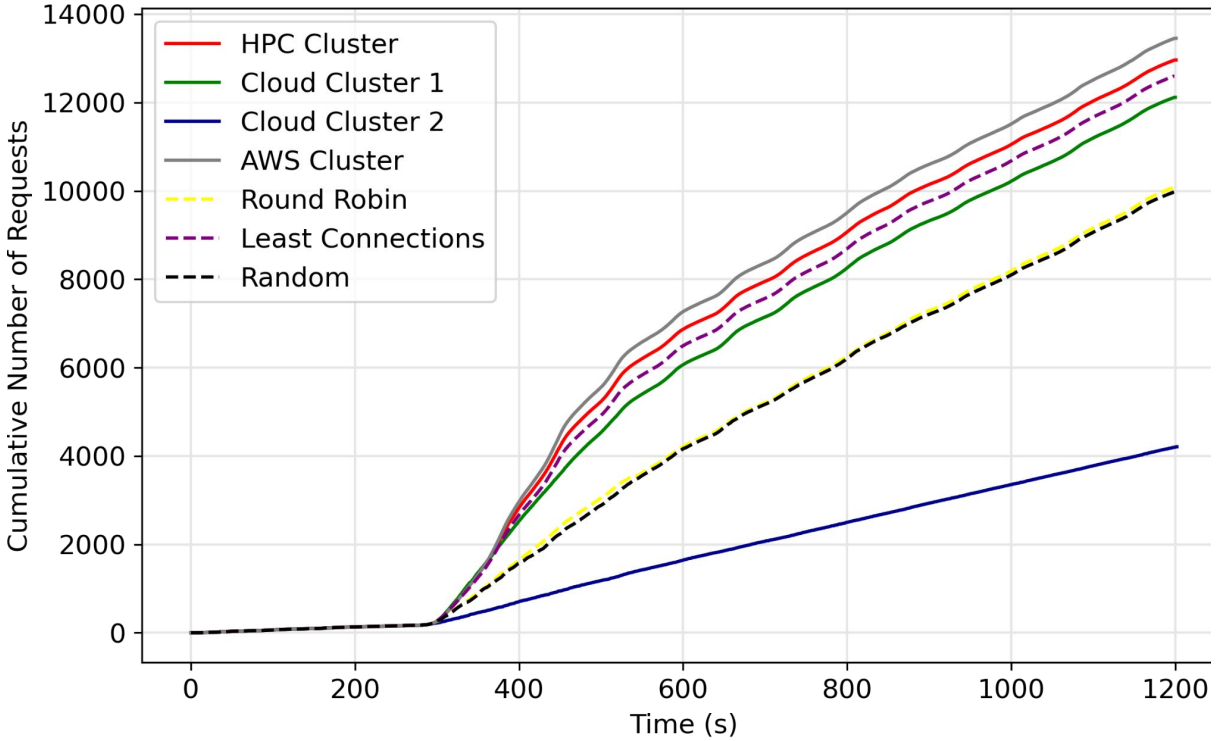


Results

Load Balancing Policies

Load balancing policy performances compared to direct routing.

“Least Connections” is a good mechanism to maximize performance while distributing load to slower nodes.



Outline

1. Motivation
2. Introducing FaDO
3. Implementation
4. Challenges
5. Results
6. Future Work

Future Work

- Extend and diversify application behavior
- Integrate authentication and access control
- Abstract storage access for different technologies
- Extend the load balancer with custom logic

Thank you for your attention!