Serverless Replication of Object Storage across Multi-Vendor Clouds and Regions

Accepted to: EuroSys '26

Paper Authors: Junyi Shu et al. (Peking University)

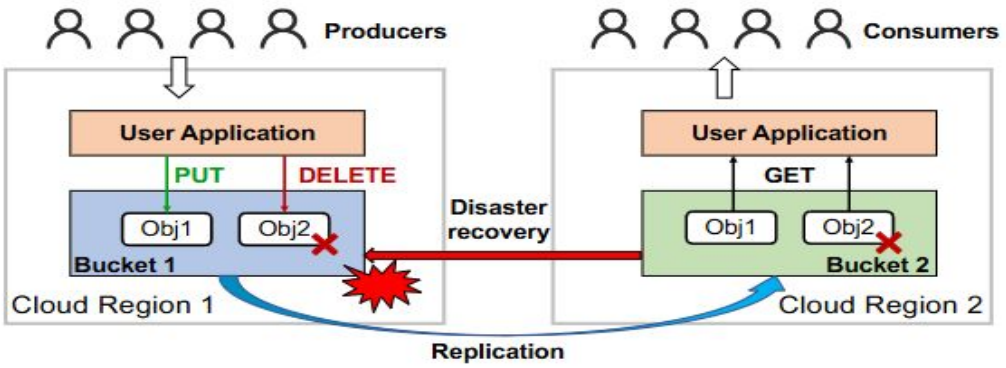
Team 4: Xiaoling Wei, Bohan Xiong, Xu Zhu

Talk Outline

- **Introduction & Problem Statement**
- **Background & Related Work**
  - Limitations of existing VM-based solutions
- **λReplica System Design**
  - Distribution-aware performance modeling
  - Decentralized part-granularity scheduling
- **Key Contributions**
  - Performance gains and cost reductions
- **Experimental Evaluation**
  - Delay, Cost, and Real-world Traces
- **Conclusion & Critique**
  - Strengths, Weaknesses, and Future Work

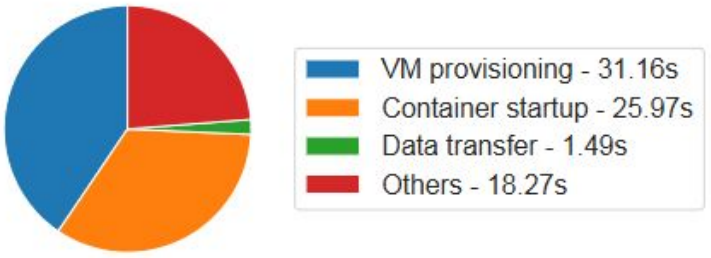
Introduction: Cross-Cloud Data Replication

- Object Storage is ubiquitous:
  - Dominates Data: 80-90% enterprise data (unstructured).
- Why Replicate?
  - Reliability: Survive regional outages.
  - Performance: Low latency for global users.
- Current Limitation:
  - Vendor Lock-in: Native tools lack cross-cloud support.

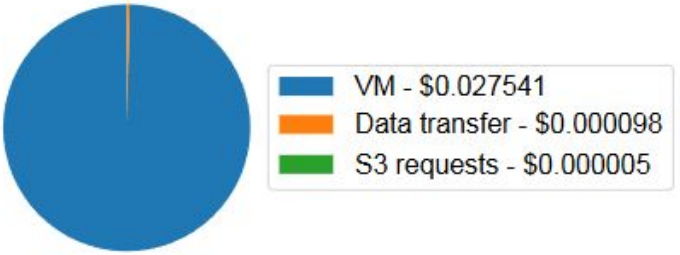


Problem: Limitations of VM-based Solutions

- Current Approach:
  - Relies on Virtual Machines (e.g., Skyplane).
- Pain Point 1: High Latency
  - Slow Provisioning: VM startup takes tens of seconds.
  - Impact: High delays for small objects or bursty traffic.
- Pain Point 2: Cost Inefficiency
  - Billing Granularity: Minimum billable duration (e.g., 1 min).
  - Impact: Wasteful for short-duration tasks.



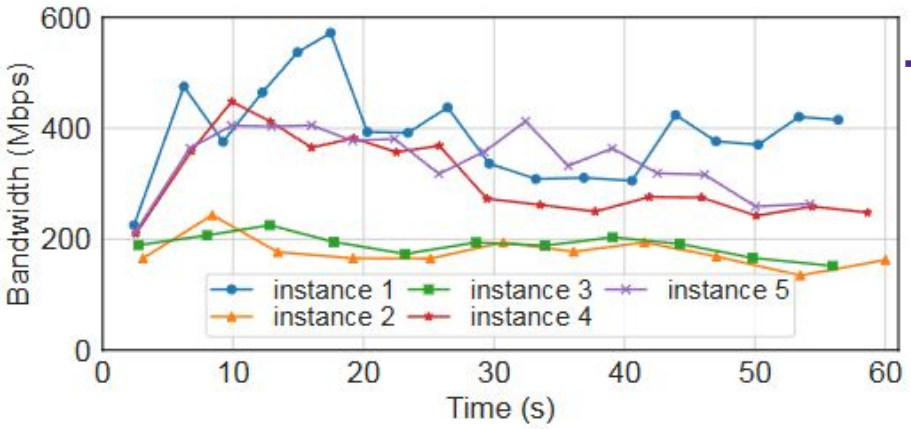
(a) Time



(b) Cost

Serverless & New Challenges

- **Proposed Solution: Cloud Functions**
  - **Fast: Millisecond-level startup (vs. tens of seconds for VMs). Elastic: Scales instantly to thousands of instances. Cost-Effective: Fine-grained billing (pay-per-ms).**



➔ **Challenge 1: Performance Asymmetry (Cloud-level)**

- **Bandwidth varies significantly across different clouds/regions. Key Insight: Not all links are equal.**

➔ **Challenge 2: Performance Variability (Instance-level)**

- **"Straggler" Problem: Performance varies randomly among instances. Result: One slow function can delay the whole transfer.**

Background: Limitations of Existing Approaches

VM-based Intercloud Brokers

- **Approach: Orchestrates VMs to move data and compute across clouds.**
- **Critical Weakness:**
  - **Slow Provisioning: "Typical provisioning time of several minutes" [1]**
  - **Inefficient: Overkill for small objects or bursty transfers.**

Cloud-Native Services (e.g., S3 Replication)

- **Vendor Lock-in: No incentives for cross-cloud support.**
- **High Cost: Requires object versioning enabled (doubles storage cost).**

Reference: [1] *SkyPilot: An Intercloud Broker for Sky Computing (Yang et al., NSDI '23).*

Acknowledgement: Generative AI (Gemini) was used to assist with paper summarization.

Why Serverless? The Unexplored Gap

The potential of Serverless | The Gap | The solution: λReplica

- **Instant Startup: Milliseconds vs. Minutes (VMs).**
- **Cost-Efficiency: Pay-per-use billing eliminates idle costs.**

- **Technical Hurdle: Extreme performance variability makes reliable replication difficult.**
- **Limitation: Functions are stateless and cannot directly address each other.**

First system to harness Serverless for robust, high-performance cross-cloud replication.

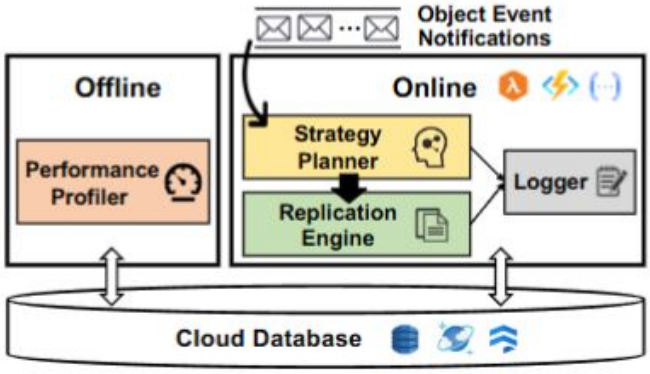
System Overview: λReplica Architecture

Offline Phase:

- **Performance Profiler: Profiles cloud/region pairs to build a distribution-aware performance model.**

Online Phase (Runtime):

- **Strategy Planner: Generates an SLO-compliant plan before replication starts.**
- **Replication Engine: Executes the plan using decentralized part-granularity scheduling to handle variability.**



(Shared State Store, e.g., DynamoDB)

The Brain: Distribution-Aware Performance Model

$$T_{rep} = T_{func} + T_{transfer}$$

Goal: Find the cheapest plan (Region + Concurrency) that meets the SLO.

Modeling Uncertainty:

- **Traditional models use averages (Static).**
- **λReplica models execution time as probability distribution (Normal / Gumbel) to capture variability.**

Prediction Logic:

- **Calculates replication Time(T_{rep}) for different parallelism levels (η).**
- **Selects the plan where the estimated tail latency (e.g., p99) $\leq$ User SLO.**

Decentralized Part-Granularity Scheduling

The Challenge:

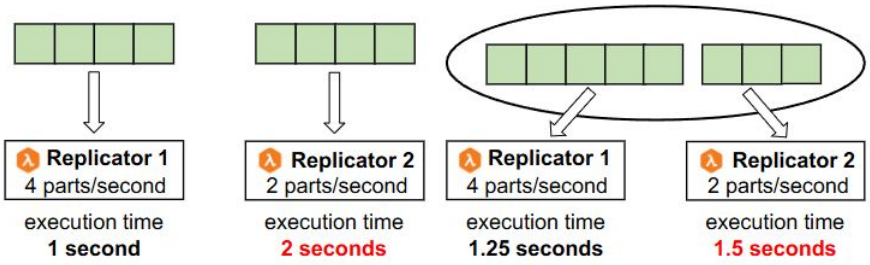
- **Equal distribution fails due to stragglers (slow instances delay everyone).**

The Solution:

- **Dynamic Assignment: Split object into small parts.**
- **Shared Pool: Functions autonomously "pull" parts from a shared pool.**

Benefit:

- **Load Balancing: Fast instances process more parts; slow instances process fewer.**
- **Minimizes the total end-to-end replication time.**



(a) Equal (b) Optimal

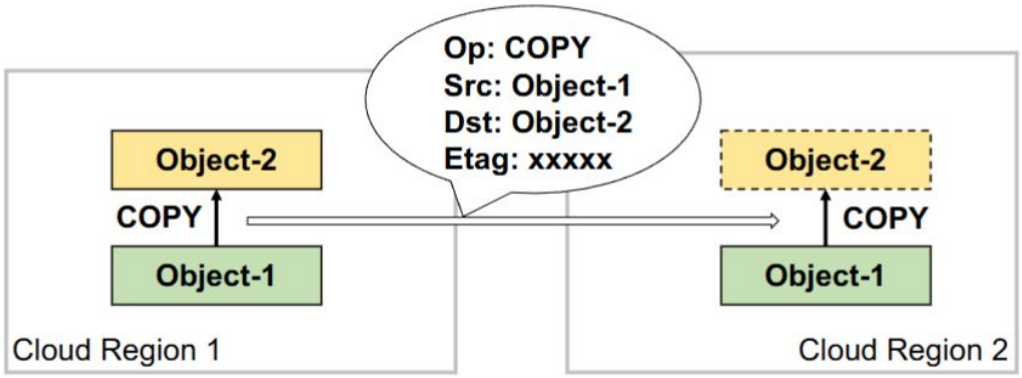
Further Optimization: Cost Reduction

Technique 1: Changelog Propagation

- Idea: Propagate operation logs (e.g., COPY, CONCAT) instead of full object data.
- Benefit: Near-zero cost for common operations (avoids unnecessary replication).

Technique 2: SLO-bounded Batching

- Idea: Aggregate frequent updates into a single transfer if the deadline permits.
- Benefit: Reduces egress cost for "hot" objects without violating SLO.



Key Contributions

01	First Serverless Replication System	• Proposed λReplica, a novel approach replacing VMs with cloud functions for cross-cloud data movement.
02	Handling Uncertainty	• Designed a Distribution-aware Performance Model for proactive planning. • Developed Decentralized Part-granularity Scheduling to mitigate runtime variability.
03	Significant Performance & Cost Gains	• Delay: Reduced by 61%-99% compared to Skyplane and commercial tools. • Cost: Reduced by up to 3 orders of magnitude. • Reliability: Maintained p99.99 delay < 10s on production traces.

Acknowledgement: Generative AI (Gemini) was used to assist with paper summarization.

Experimental Setup and Evaluation Goals

Evaluation goals	Quantify how much λReplica reduces **replication delay** and **cost** vs existing solutions
Platforms & deployment	Three major clouds: **AWS, Azure, GCP**. λReplica deployed as cloud functions (AWS Lambda: 512 MB – 1 GB, Azure Functions: minimum 2048 MB, Google Cloud Run: 1024 MB memory, **1–2 vCPUs**); metadata stored in serverless NoSQL databases.
Workloads	Synthetic objects with sizes **1 MB, 128 MB, 1 GB**, plus a **100 GB** bulk replication scenario **Real-world trace:** 1-hour segment of IBM COS production trace with ~0.99M PUT/DELETE requests
Baselines	**Skyplane** (open-source VM-based cross-cloud/region replication) **AWS S3 Replication Time Control (S3 RTC)** and **Azure object replication (AZ Rep)** as proprietary intra-cloud baselines
Metrics	**Replication delay:** time from completing a PUT to successfully retrieving that version (or a newer one) at the destination. **Cost:** estimated from provider price lists and measured resource usage (functions, storage, data egress, API calls).

Acknowledgement: GPT was used to assist with analysis the experiment.

Main Quantitative Results: Delay & Cost - 1

Table 1. Replication delay and cost from AWS us-east-1

Cloud			AWS			Azure			GCP		
Region			ca-central-1	eu-west-1	ap-northeast-1	eastus	uksouth	southeastasia	us-east1	europe-west6	asia-northeast1
1MB	Delay (second)	λReplica	1.5	1.5	1.6	1.3	2.0	2.5	1.4	2.2	3.3
		Skyplane	76.2	84.7	90.2	134.3	146.5	149.4	109.4	126.5	115.2
		S3 RTC	21.3	24.1	24.5	N/A	N/A	N/A	N/A	N/A	N/A
		Δ	**-92.79%**	**-93.62%**	**-93.60%**	**-99.00%**	**-98.66%**	**-98.32%**	**-98.76%**	**-98.26%**	**-97.09%**
	Cost ($10^{-4}$$)	λReplica	0.3	0.3	0.3	0.9	1.0	1.0	1.0	1.0	1.1
		Skyplane	541.6	541.6	586.9	768.9	1104.9	1152.9	771.9	1238.1	845.2
		S3 RTC	0.4	0.4	0.4	N/A	N/A	N/A	N/A	N/A	N/A
		Δ	**-32.25%**	**-31.43%**	**-28.55%**	**-99.88%**	**-99.91%**	**-99.91%**	**-99.88%**	**-99.92%**	**-99.87%**

Table 1 shows the replication delay and cost of λReplica from AWS us-east-1 to the other nine regions.
λReplica outperforms the best baseline in every experiment and reduces the replication delay by 61%-99%.
λReplica also reduces the cost by 28.5%-99.9%. Using λReplica with underlying Lambda and DynamoDB is more cost-effective than S3 RTC, providing 28.5%-39.9% cost savings.

Main Quantitative Results: Delay & Cost - 2

Bulk replication results - the replication time and cost of λReplica and Skyplane for replicating a 100 GB object

- The notification delay is not included in these experiments
- Skyplane still suffers from the non-negligible VM provisioning time. When multiple VMs are used and one starts slowly, the others must wait, increasing replication time and cost.
- λReplica can replicate the 100GB object in a minute, reducing the replication time by 76%-91%.
- λReplica does not reduce the cost significantly compared to Skyplane because the fixed data egress cost dominates for large objects
- For 100GB objects, λReplica uses 128-512 function instances to replicate between the reported region pairs

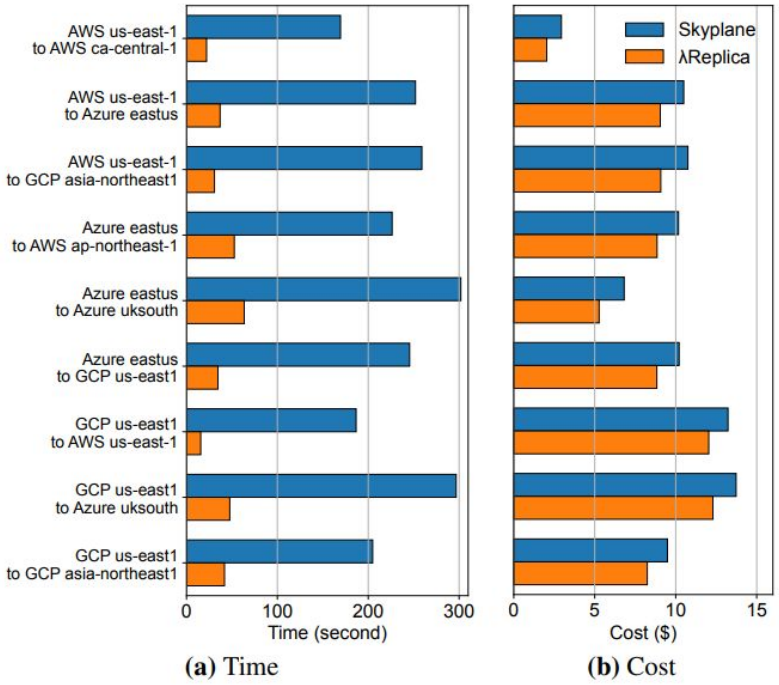
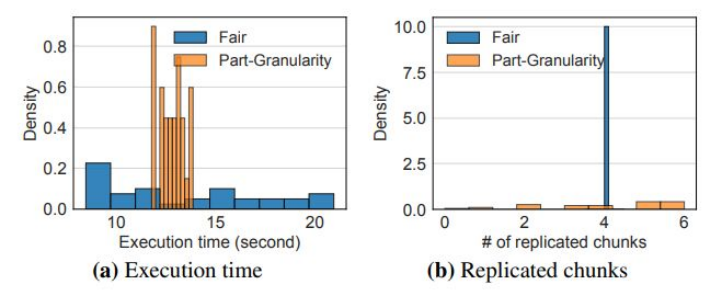


Figure 16. Replication time and cost of a 100GB object.

Ablation Studies and Real-World Evaluation

Impact of decentralized part-granularity scheduling



- Compared to fair, equal-part dispatching, λReplica's decentralized scheduling lets faster instances process more chunks and slower ones process fewer or none.
- This balances completion times across functions and significantly reduces tail latency for large objects.

Effectiveness of dynamic region selection

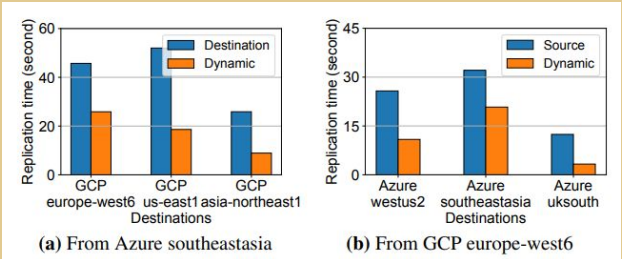


Figure 20. Effectiveness of dynamic replication strategy.

- Neither statically using the source region nor the destination region can provide optimal performance.
- Dynamically selecting where to execute the functions can significantly reduce the replication time.

Ablation Studies and Real-World Evaluation - 2

Effectiveness of SLO-bounded batching

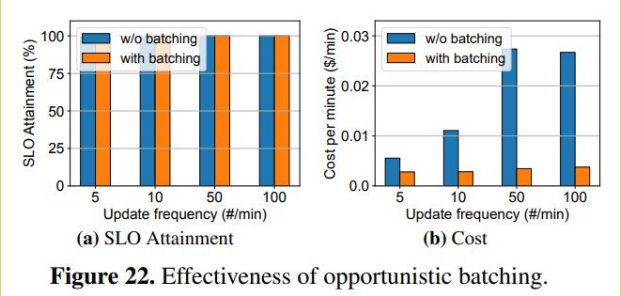


Figure 22. Effectiveness of opportunistic batching.

- Experimental Setup: 1) Object size: 100 MB, 2) Update frequency: 5, 10, 50, 100 updates/min, 3) SLO: 30 second
- SLO attainment is nearly 100% in all cases
- The real benefit of batching is dramatic cost reduction under high update rates

Real-World Object Storage Trace

- S3 RTC's replication time is typically around 20 seconds, but its p99.99 delay exceeds 30 seconds during traffic bursts.
- λReplica 's elasticity and adaptivity keeps the p99.99 replication time under 10 seconds for the entire period, despite dynamic bursts and varying object sizes.

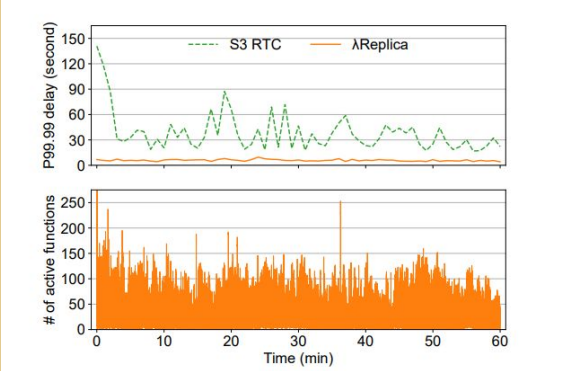


Figure 23. Replication time on the IBM production trace.

Conclusions and Takeaways

λReplica summary	A serverless, platform-independent system for replicating objects across **multiple clouds and regions**.
Key design ideas	Uses a **distribution-aware performance model**, **decentralized part-granularity scheduling**, and **changelog propagation with SLO-bounded batching** to plan replication under user-specified SLOs.
Experimental conclusions	Across AWS, Azure, and GCP, λReplica **reduces replication delay by about 61–99%** and achieves **significant cost savings (up to orders of magnitude)** compared to VM-based and provider-native baselines.
Real-world impact	On a one-hour IBM COS production trace with ~1M updates, λReplica **keeps p99.99 replication delay under ≈10 seconds**, even during bursty traffic.
Take-home message	**Serverless functions can be an effective building block for fast, SLO-aware, and cost-efficient multi-cloud object replication**, enabling use cases such as global model distribution and geo-distributed applications.

Critique: Strengths

Massive Performance Improvement

> **λReplica** outperforms existing solutions and proprietary cloud services, reducing replication delay by **61%–99%**.
> This far exceeds the "10% improvement" benchmark, offering near-synchronous speeds.
> Maintains p99.99 replication delay below **10 seconds** even on production traces with bursty traffic.

Significant Cost Reduction

> Achieves cost savings of up to **three orders of magnitude** compared to VM-based solutions.
> Leverages the millisecond-level billing of serverless functions to eliminate idle resource costs.

Scalability & Elasticity

> Exploits serverless elasticity to handle transient bursts instantly without the provisioning time required for VMs.
> Performance scales nearly linearly with the number of function workers.

Algorithmic Innovation

> **Decentralized Part-Granularity Scheduling:** Effectively mitigates the "straggler" problem inherent in serverless instances by allowing faster instances to steal work from a shared pool.

Acknowledgement: Generative AI (Gemini) was used to assist with paper summarization.

Critique: Weaknesses

Consistency Model Limitations

> The system guarantees **Eventual Consistency**, aligning with the standard adopted by major cloud providers for cross-region replication.
> Implementing strong consistency across multi-vendor clouds is largely **practically infeasible** due to high WAN latency; attempting to do so would negate the performance and cost benefits of the serverless architecture.

Implementation Complexity & Dependencies

> **Requires managing external state in cloud databases to handle locking and coordination, which adds architectural complexity compared to simple point-to-point transfers.**
> Relies on **offline profiling** to train performance models when onboarding new regions, rather than being fully self-adapting online immediately.

Platform-Specific Constraints

> **Cost Effectiveness Variability:** While highly efficient on AWS, the cost benefits are less pronounced on GCP due to higher pricing for Cloud Run and Firestore.
> **Resource Quotas:** Users are still bound by cloud provider concurrency limits, potentially requiring manual quota increases for massive workloads.

Critique: Evaluation

Comprehensive & Reproducible

- **Baselines:** Compared against the state-of-the-art open-source solution (**Skyplane**) and commercial proprietary tools (**AWS S3 RTC**, **Azure Object Replication**).
- **Environments:** Evaluated across three major cloud providers (AWS, Azure, GCP) and multiple geographic regions.
- **Reproducibility:** The prototype code is open-sourced, allowing verification of results.

Identify GAPS

Strong Consistency in Serverless

Gap: The current approach relies on eventual consistency. Solving strong consistency efficiently in a stateless, serverless environment without high latency penalties remains an open challenge.

Handling Extreme Contention

Gap: The concurrency control relies on an abort-and-retry mechanism.

Fully Online Adaptation

Future Work: Eliminating the need for **offline profiling** to onboard new clouds. A fully zero-configuration, online learning model would improve usability.

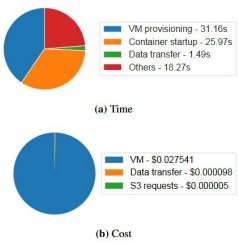
Cross-Cloud Cost Parity

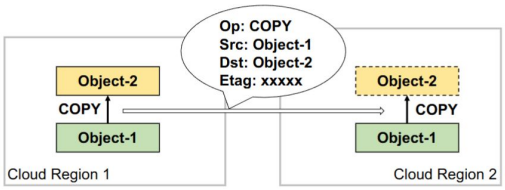
Limitation: The cost savings are uneven across vendors. Future research could explore "Cloud-Arbitrage" scheduling to route data through cheaper intermediate hops to normalize costs.

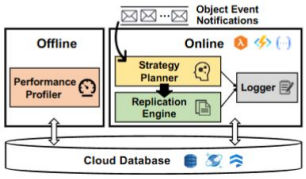
Questions

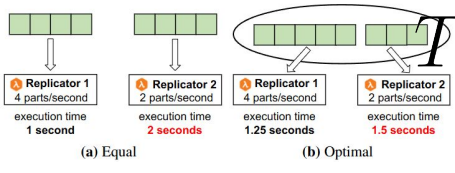
Q&A Session

(Skipped slides) Raw materials

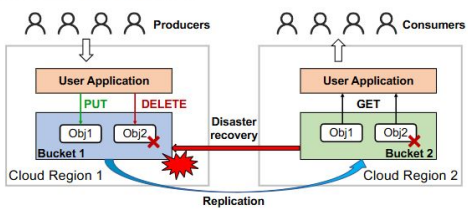








$$T_{rep} = T_{func} + T_{transfer}$$



Microsoft Azure
Blob Storage

Why Serverless? The Unexplored Gap

The potential of Serverless

- **Instant Startup: Milliseconds vs. Minutes (VMs).**
- **Cost-Efficiency: Pay-per-use billing eliminates idle costs.**

Why hasn't it been done? (The Gap)

- **Technical Hurdle: Extreme performance variability makes reliable replication difficult.**
- **Limitation: Functions are stateless and cannot directly address each other.**

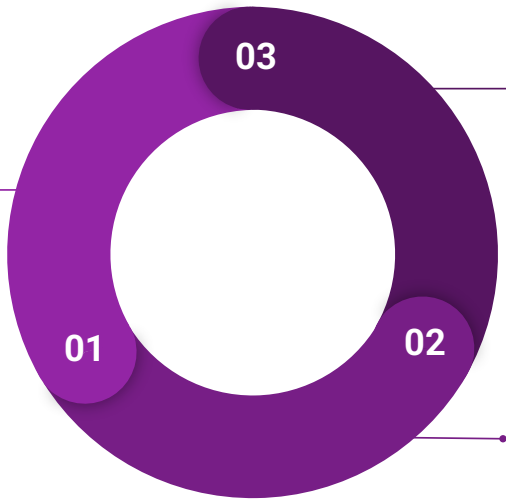
Our Approach: λReplica

- **First system to harness Serverless for robust, high-performance cross-cloud replication.**

Key Contributions

First Serverless Replication System

Proposed λReplica, a novel approach replacing VMs with cloud functions for cross-cloud data movement.



Handling Uncertainty

- Designed a Distribution-aware Performance Model for proactive planning.
- Developed Decentralized Part-granularity Scheduling to mitigate runtime variability.

Significant Performance & Cost Gains

- Delay: Reduced by 61%-99% compared to Skyplane and commercial tools.
- Cost: Reduced by up to 3 orders of magnitude.
- Reliability: Maintained p99.99 delay < 10s on production traces.

Key Contributions

First Serverless Replication System

- Proposed λReplica, a novel approach replacing VMs with cloud functions for cross-cloud data movement.

Handling Uncertainty

- Designed a Distribution-aware Performance Model for proactive planning.
- Developed Decentralized Part-granularity Scheduling to mitigate runtime variability.

Significant Performance & Cost Gains

- Delay: Reduced by 61%-99% compared to Skyplane and commercial tools.
- Cost: Reduced by up to 3 orders of magnitude.
- Reliability: Maintained p99.99 delay < 10s on production traces.