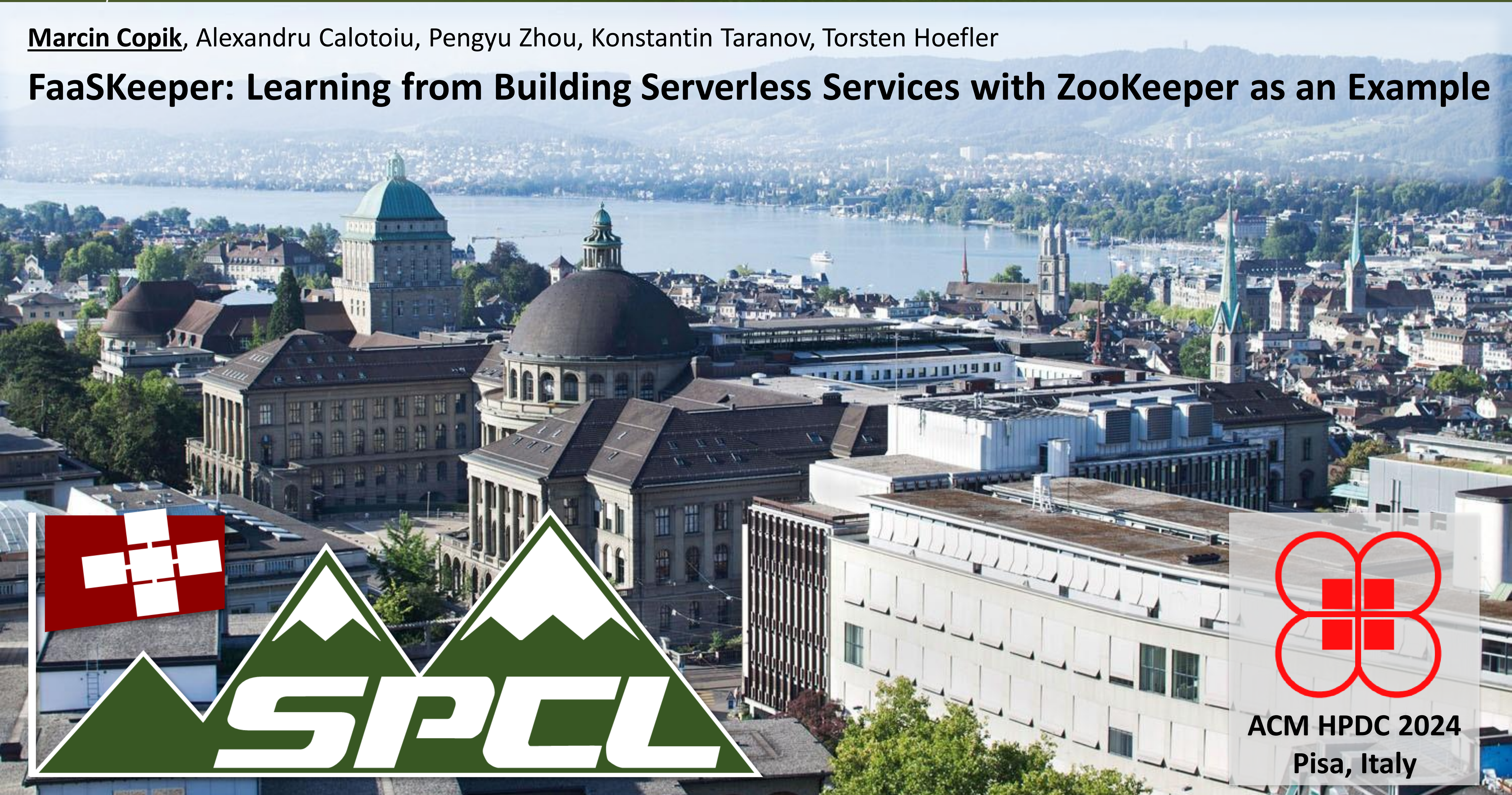
Marcin Copik, Alexandru Calotoiu, Pengyu Zhou, Konstantin Taranov, Torsten Hoefler

FaaSKeeper: Learning from Building Serverless Services with ZooKeeper as an Example

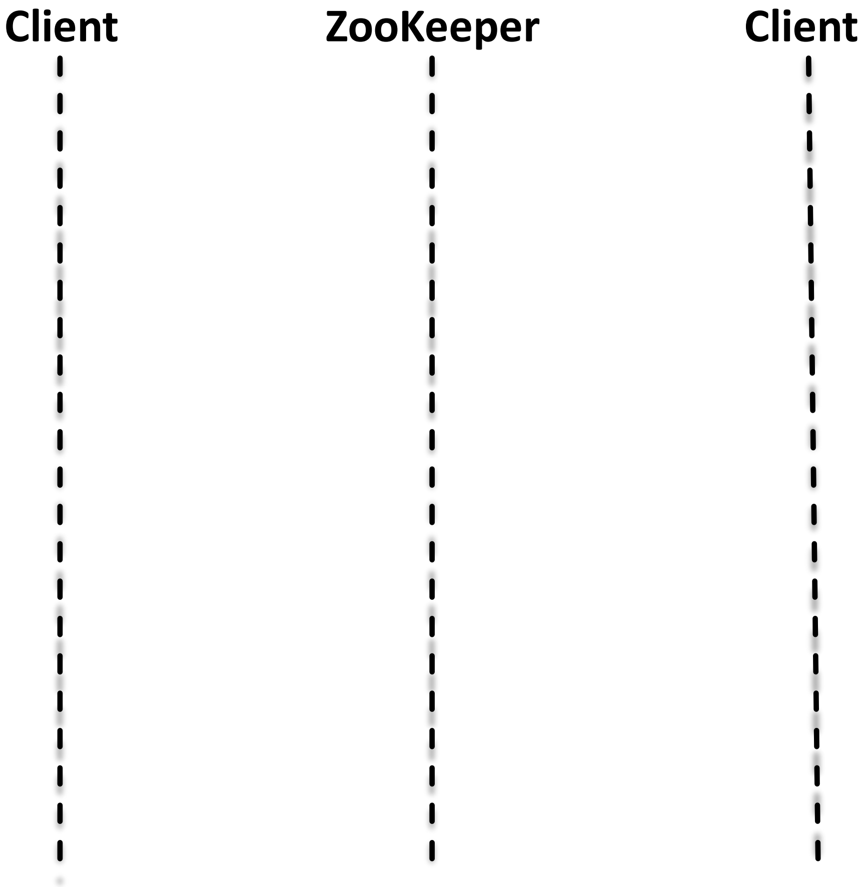


ACM HPDC 2024
Pisa, Italy

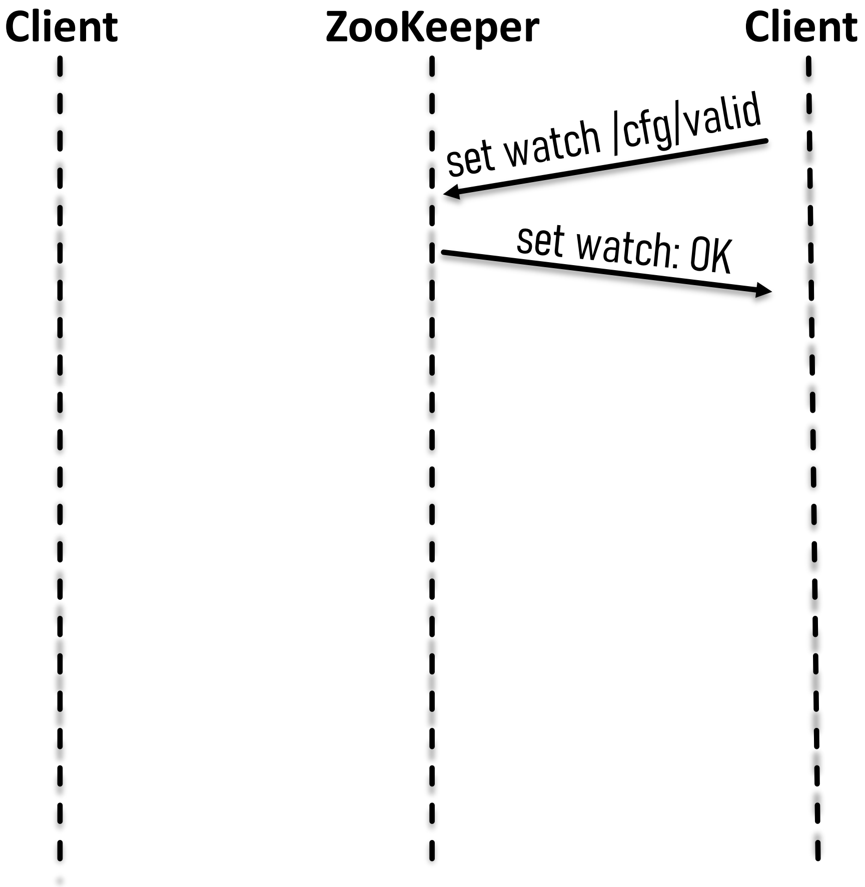
What is ZooKeeper?

What is ZooKeeper?

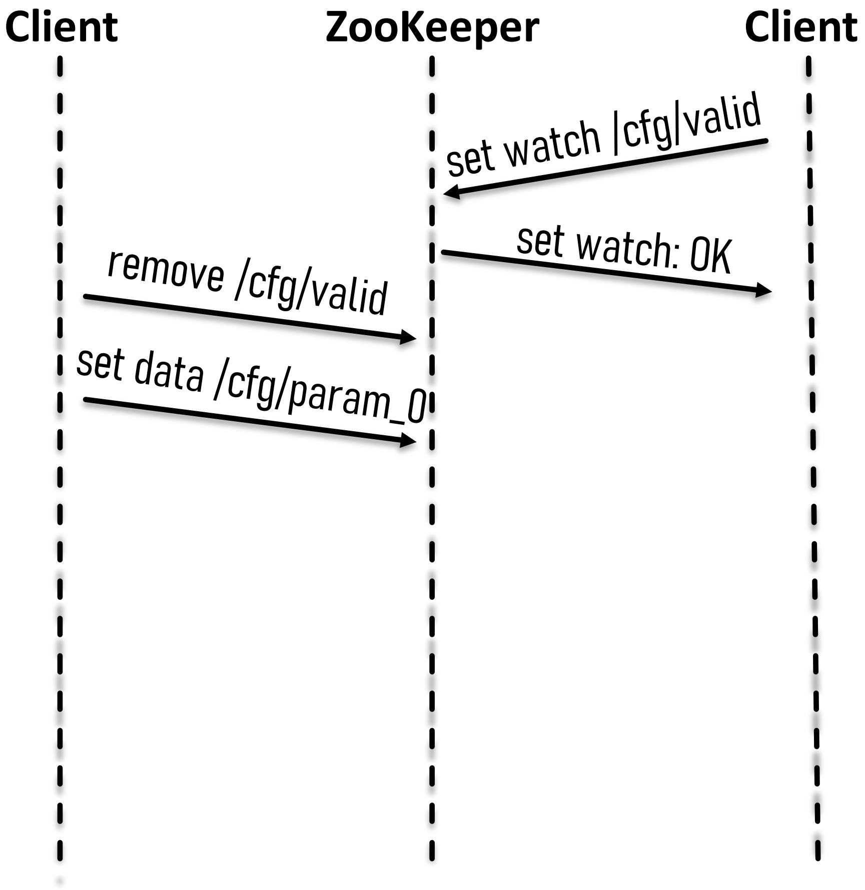
What is ZooKeeper? How it’s used in practice?



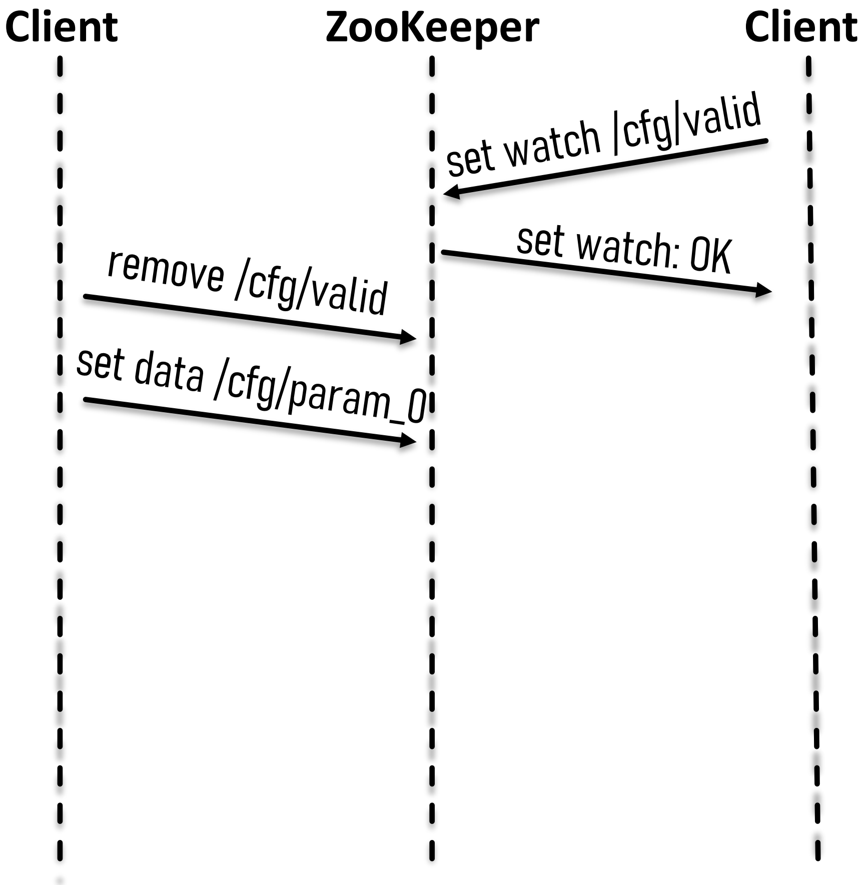
What is ZooKeeper? How it’s used in practice?



What is ZooKeeper? How it’s used in practice?

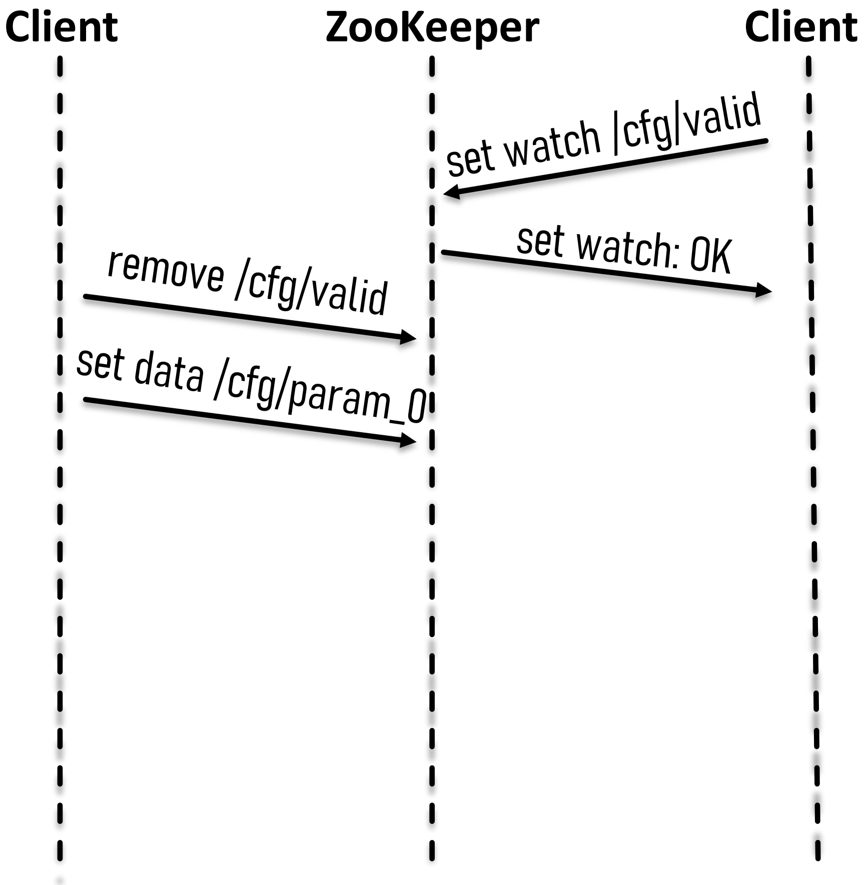


What is ZooKeeper? How it's used in practice?



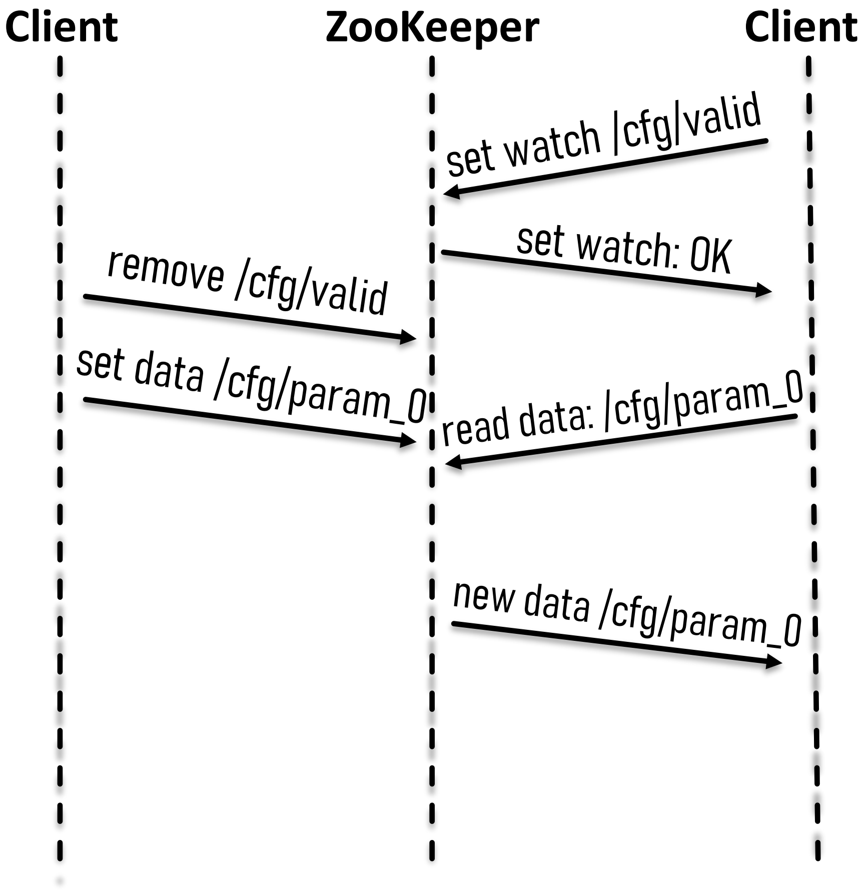
1 **Atomicity**

What is ZooKeeper? How it's used in practice?



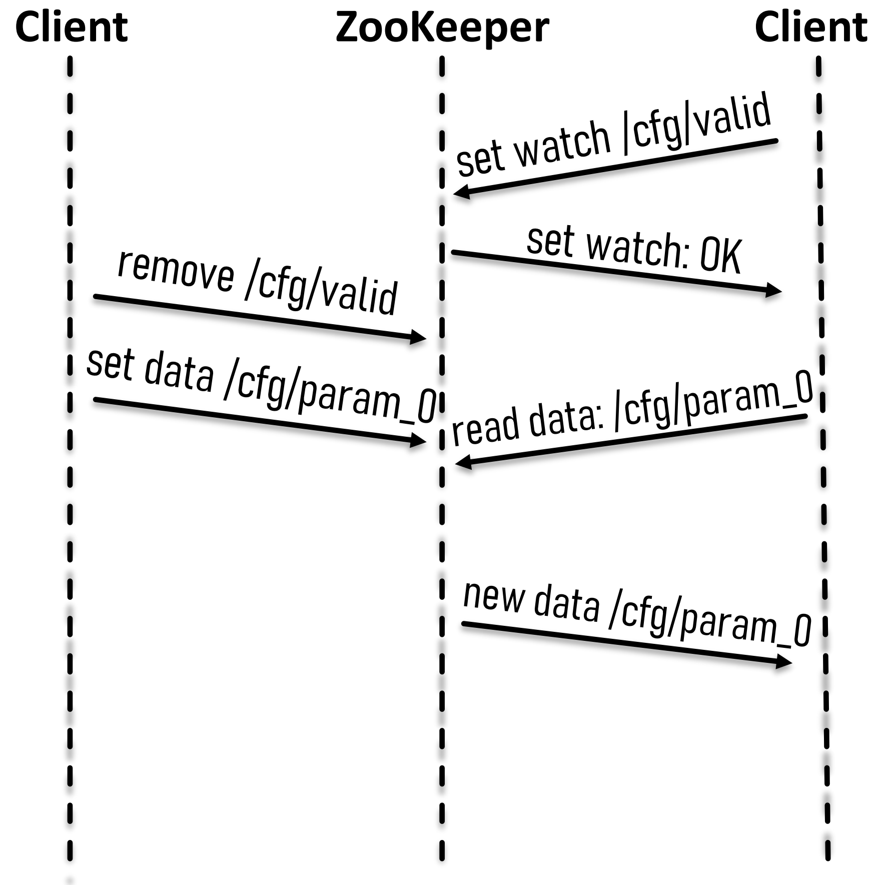
1. **Atomicity**
2. **Linearized Writes**

What is ZooKeeper? How it's used in practice?



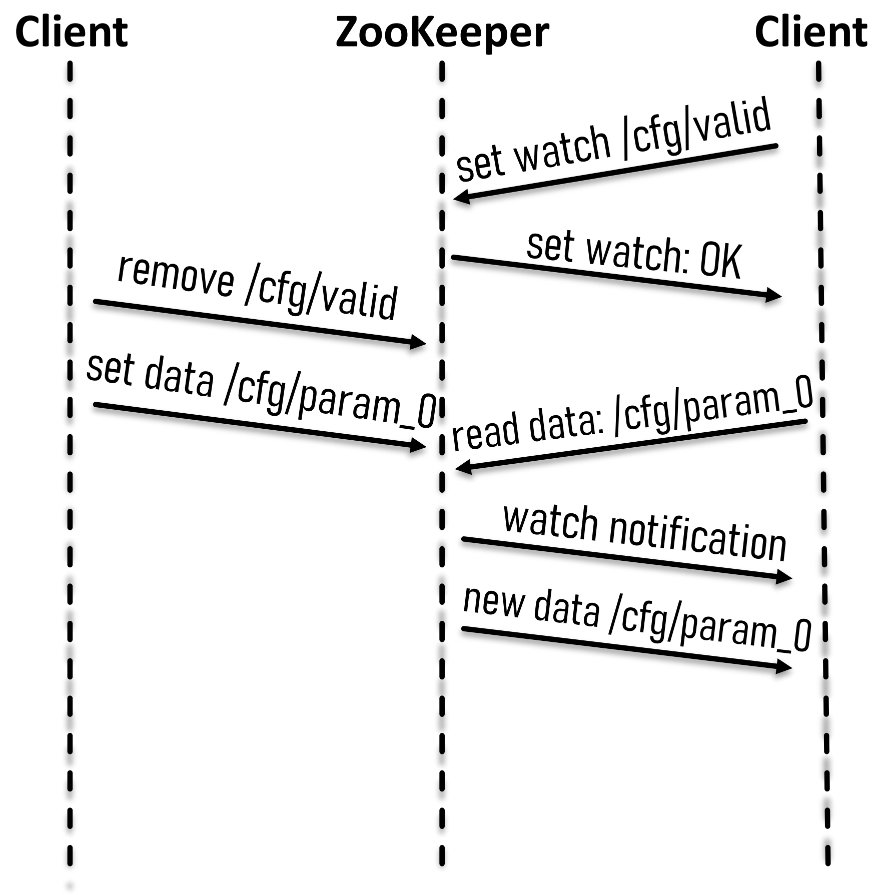
1. **Atomicity**
2. **Linearized Writes**

What is ZooKeeper? How it's used in practice?



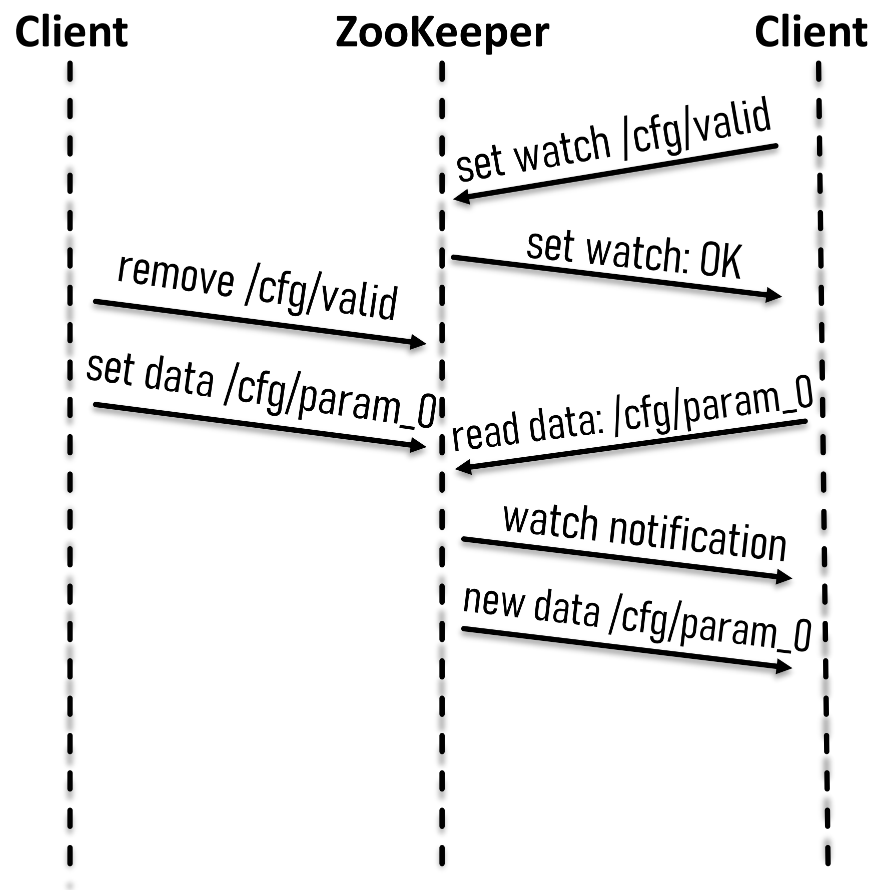
1. **Atomicity**
2. **Linearized Writes**
3. **Single System Image**

What is ZooKeeper? How it's used in practice?



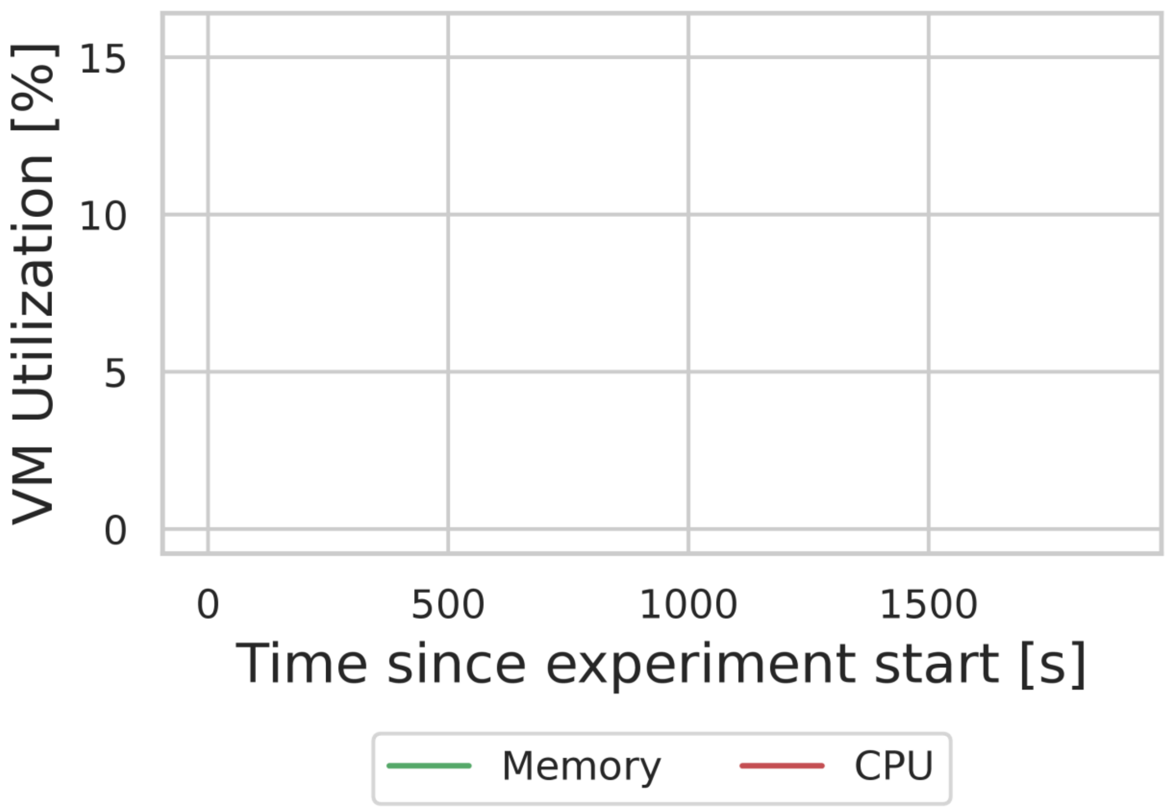
1. Atomicity
2. Linearized Writes
3. Single System Image

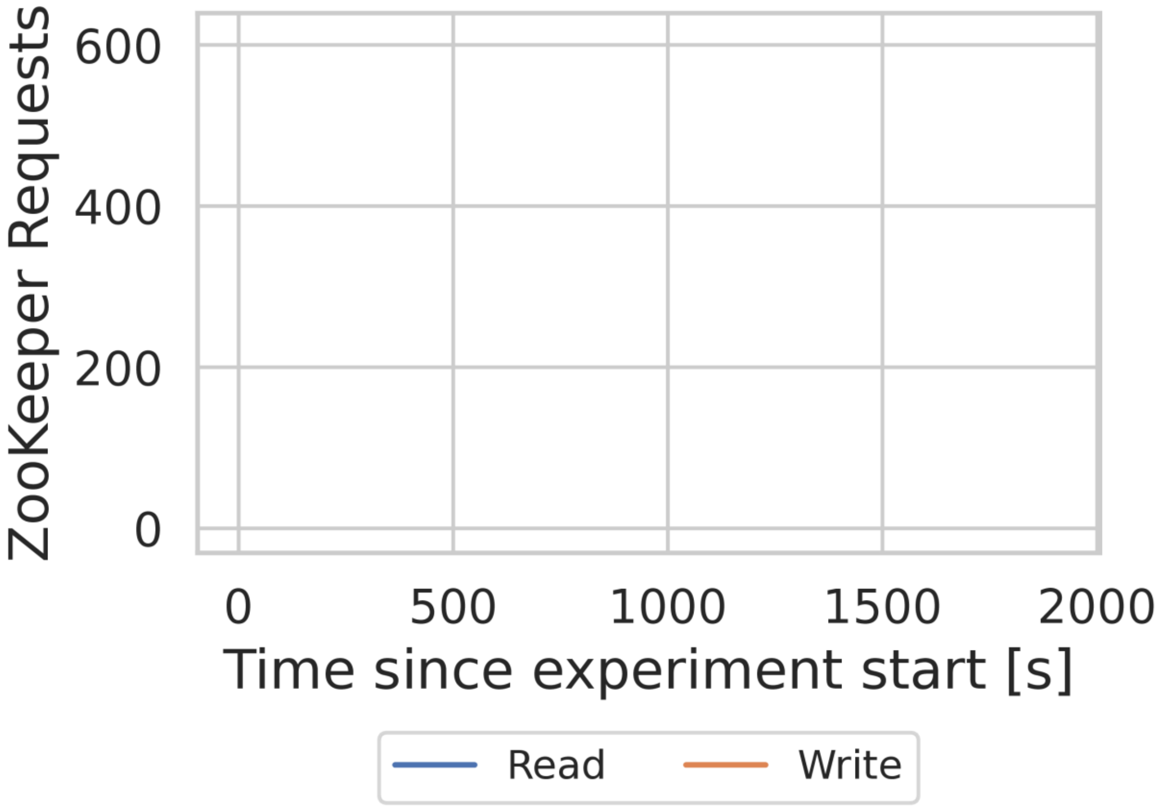
What is ZooKeeper? How it's used in practice?



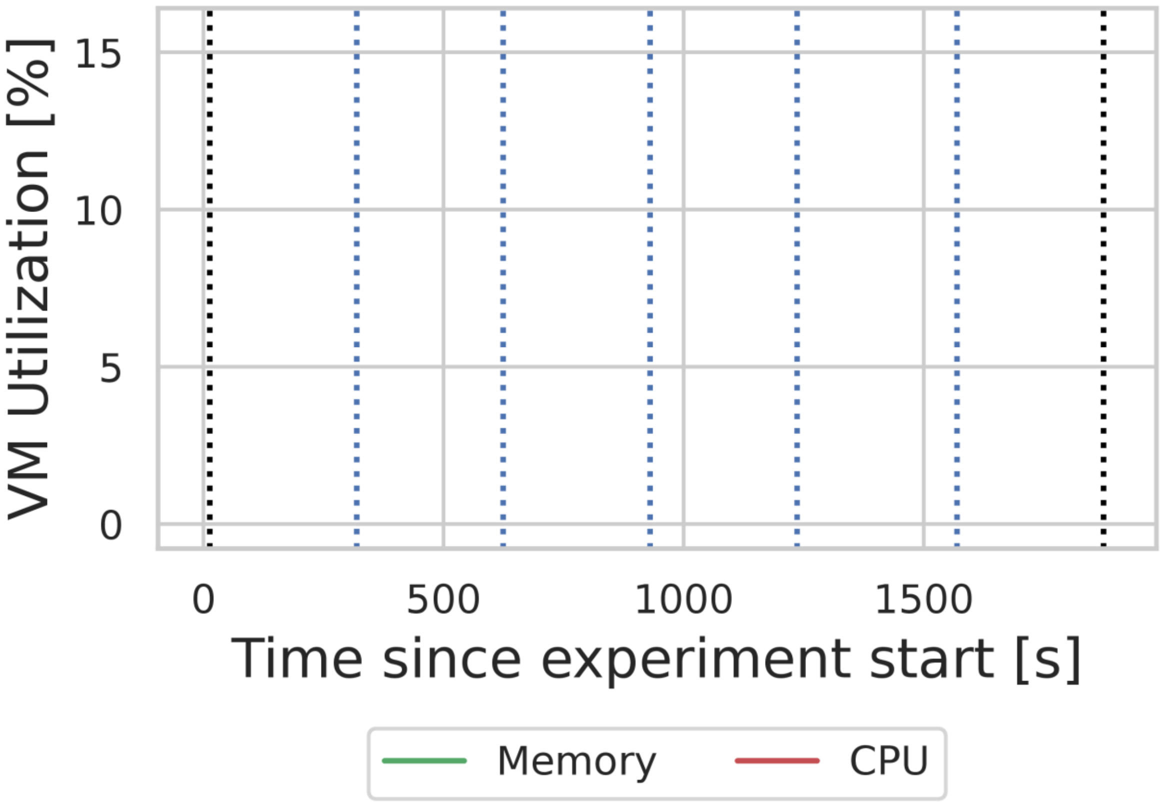
1. Atomicity
2. Linearized Writes
3. Single System Image
4. Ordered Notifications

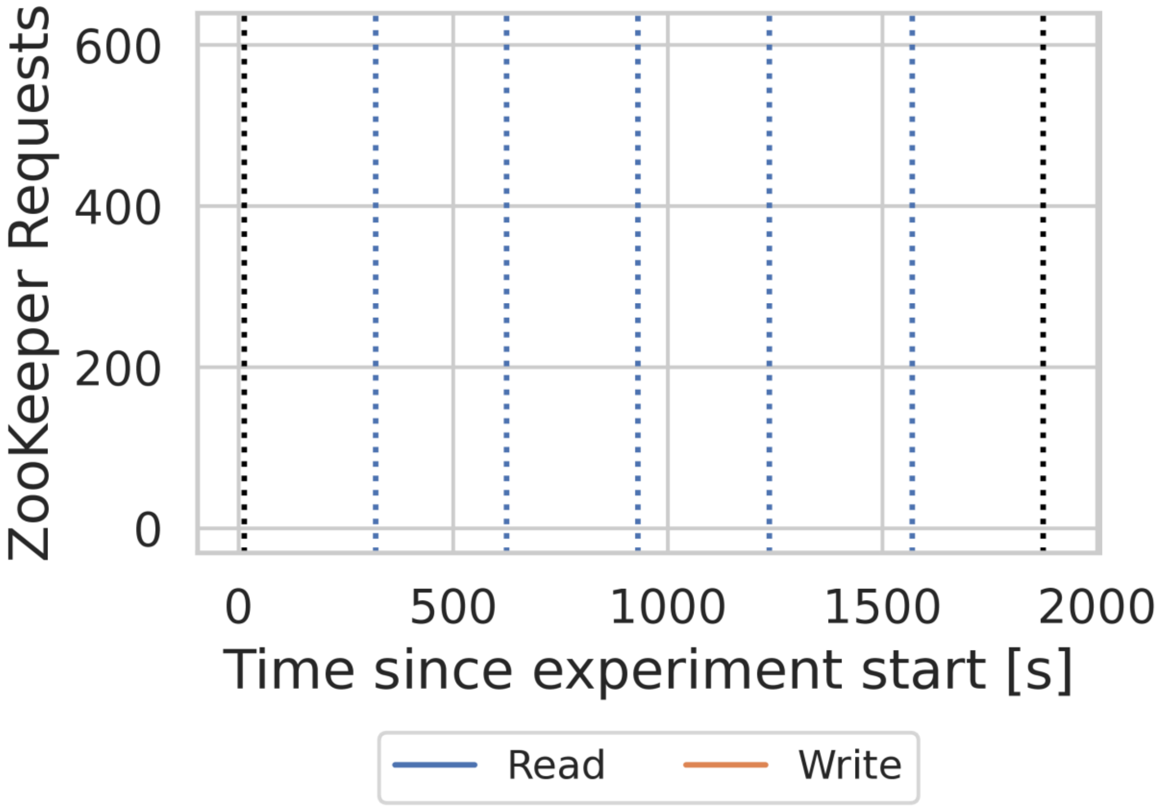
ZooKeeper Utilization in Practice: APACHE HBASE



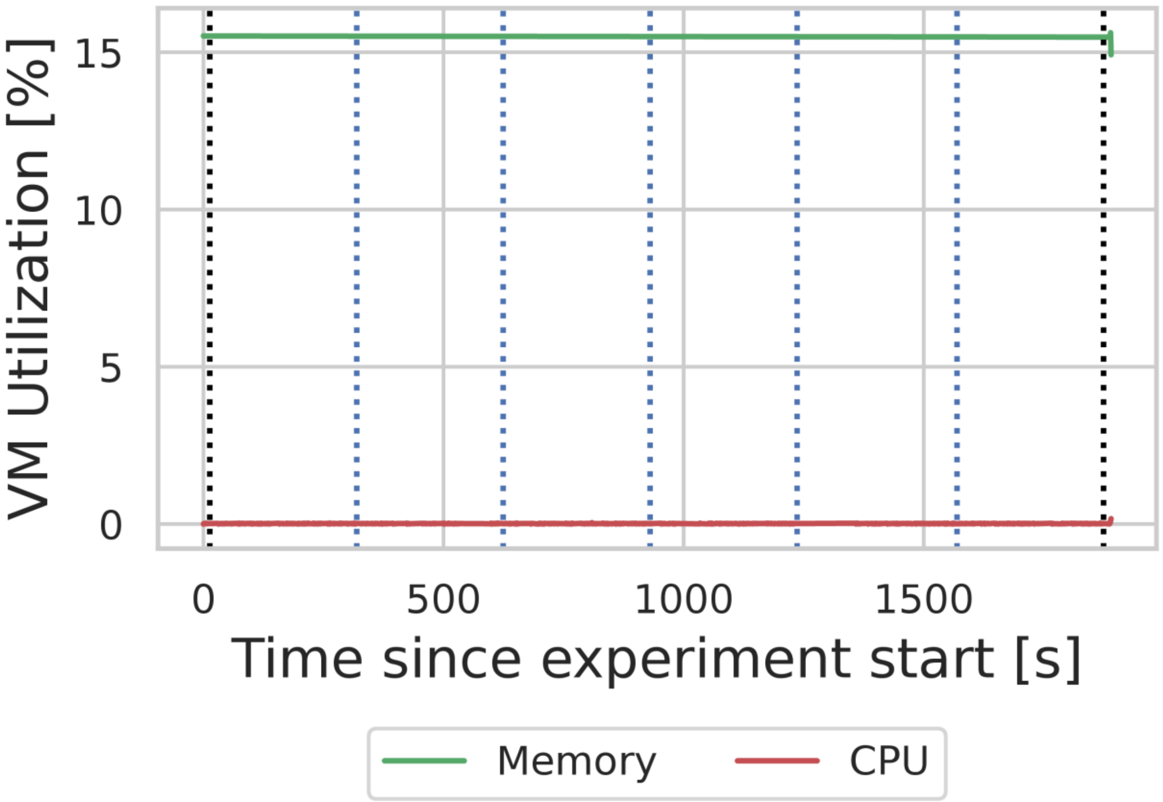


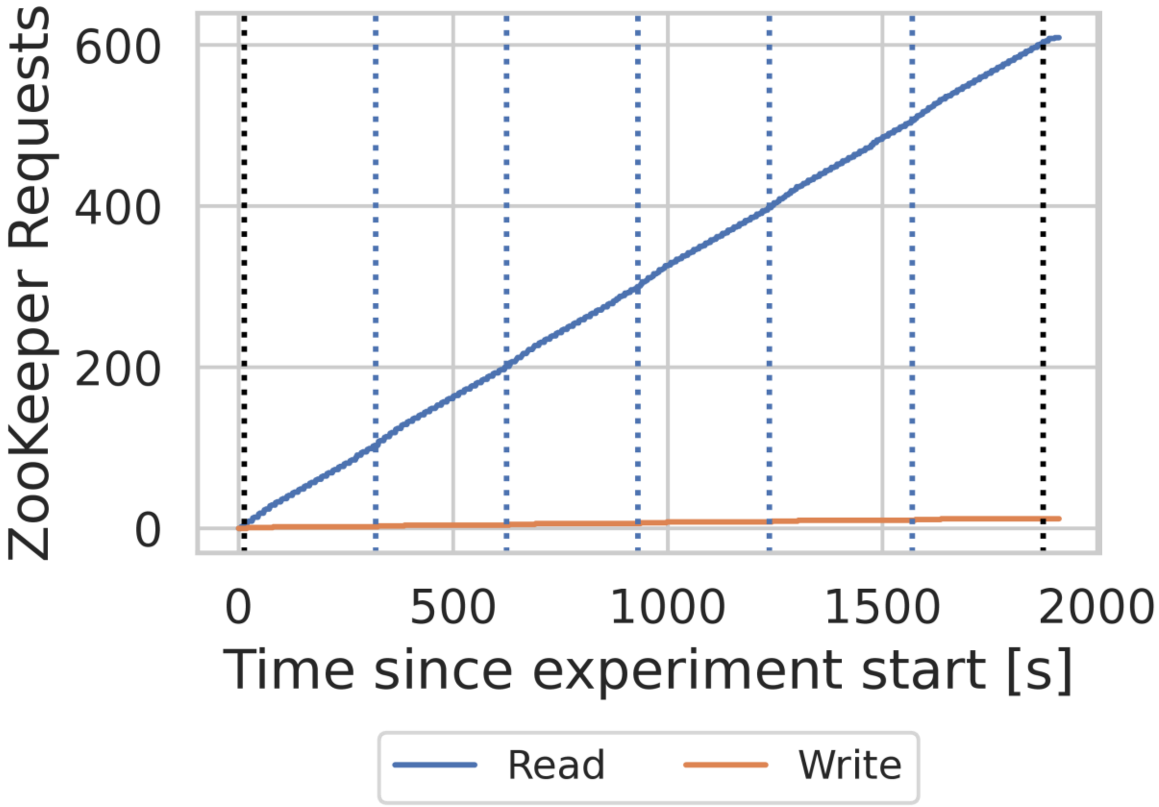
ZooKeeper Utilization in Practice: APACHE HBASE



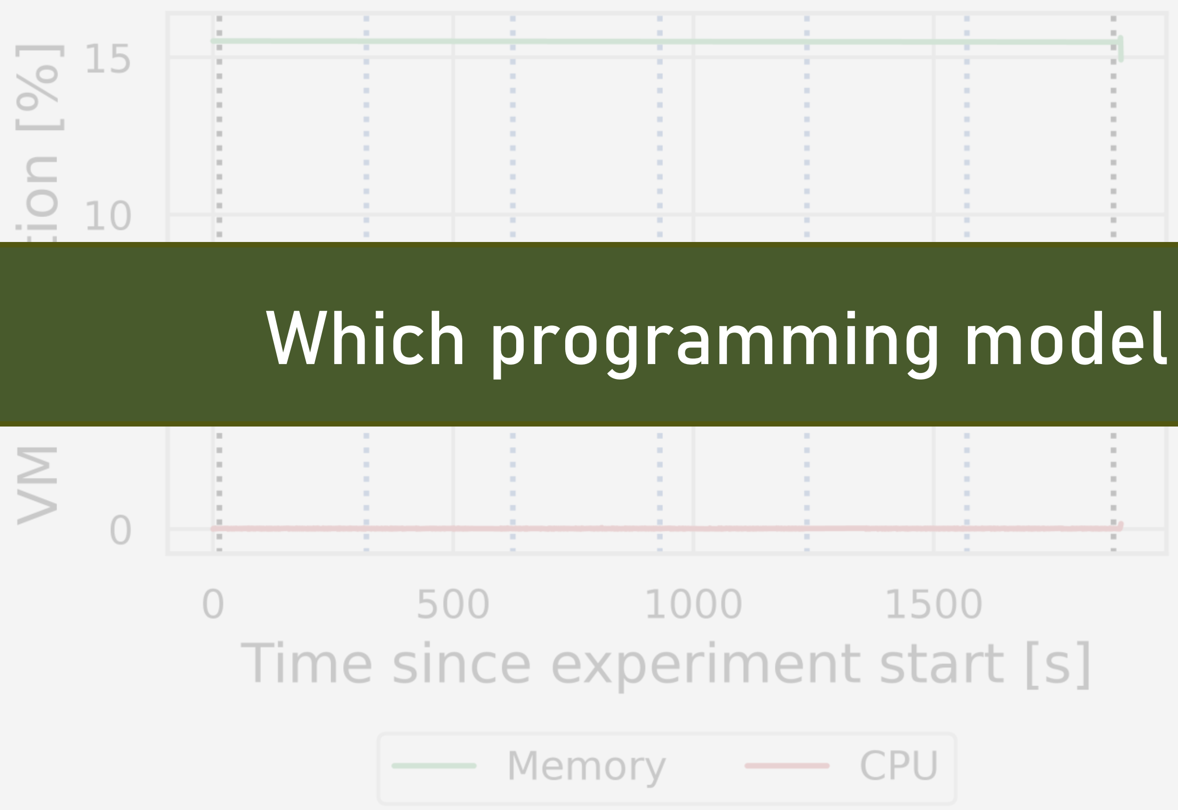


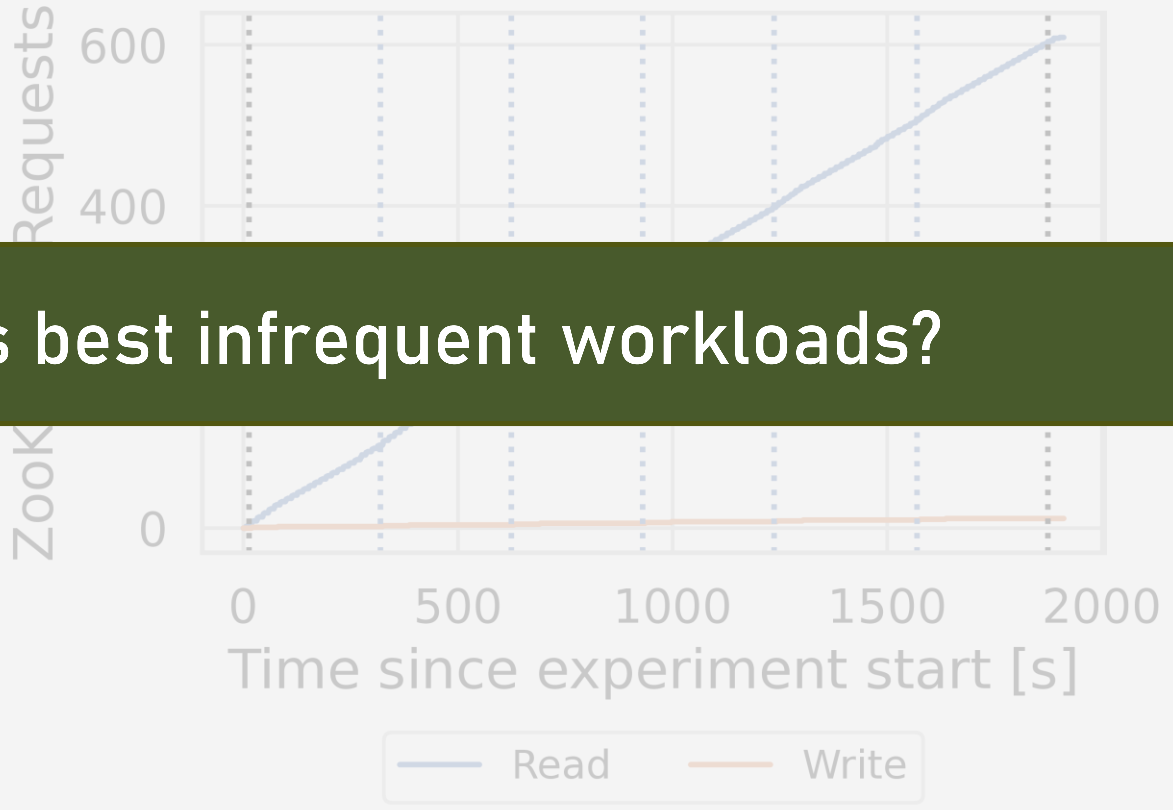
ZooKeeper Utilization in Practice: APACHE HBASE





ZooKeeper Utilization in Practice: APACHE HBASE





Which programming model fits best infrequent workloads?

How does Function-as-a-Service (FaaS) work?

How does Function-as-a-Service (FaaS) work?

```
def handler_function(request: dict, context: dict):

    data = cloud_storage.read(request['id'])

    new_data = process_logic(request['op'], data)

    stamp = cloud_storage.write(request['id'], new_data)

    return stamp
```

How does Function-as-a-Service (FaaS) work?

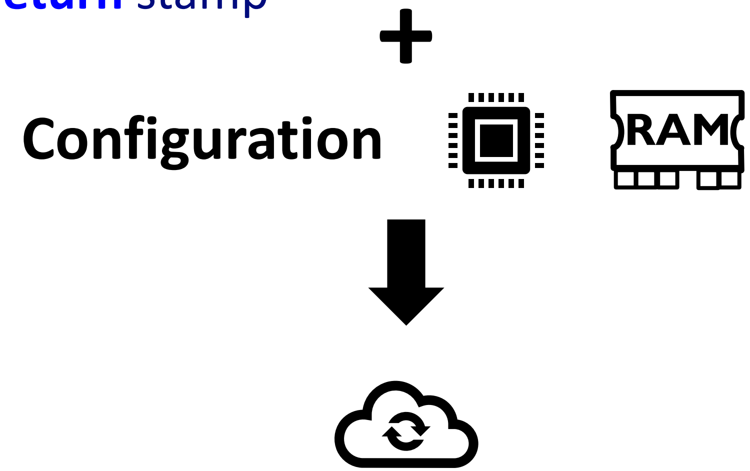
```
def handler_function(request: dict, context: dict):

    data = cloud_storage.read(request['id'])

    new_data = process_logic(request['op'], data)

    stamp = cloud_storage.write(request['id'], new_data)

    return stamp
```

+

Configuration RAM

How does Function-as-a-Service (FaaS) work?

```
def handler_function(request: dict, context: dict):

    data = cloud_storage.read(request['id'])

    new_data = process_logic(request['op'], data)

    stamp = cloud_storage.write(request['id'], new_data)

    return stamp
```



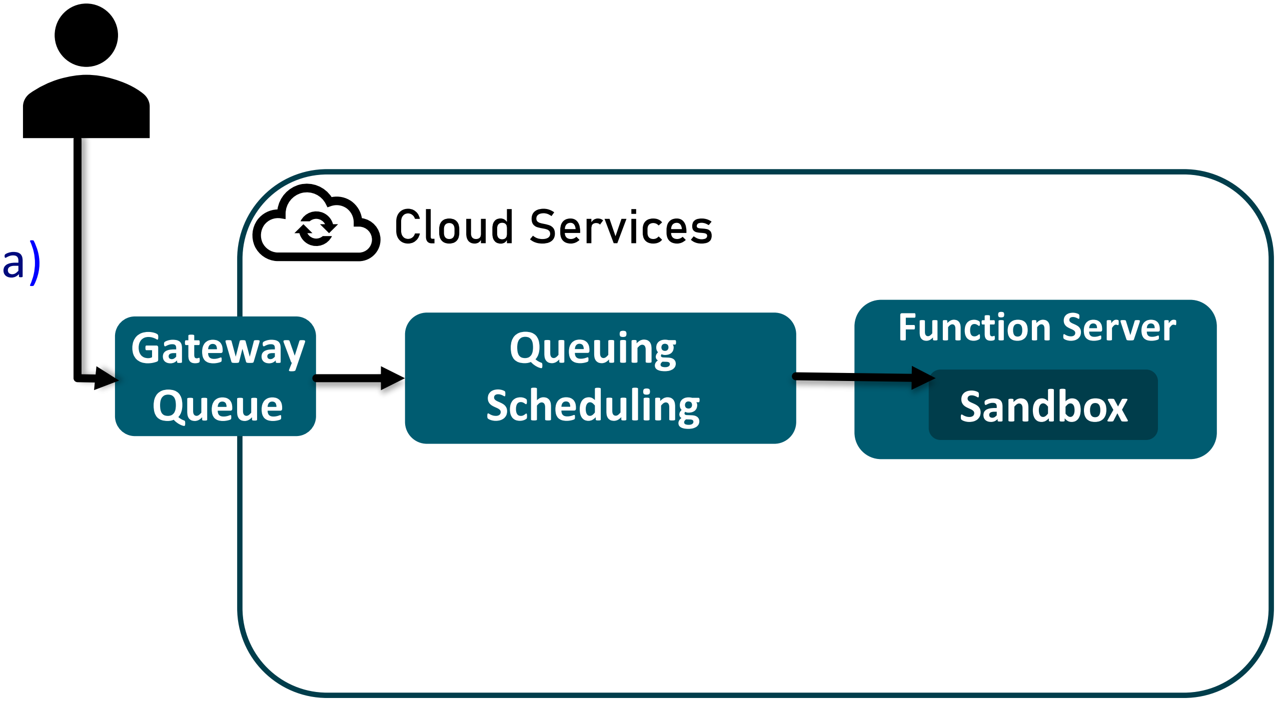
Configuration

How does Function-as-a-Service (FaaS) work?

```
def handler_function(request: dict, context: dict):

    data = cloud_storage.read(request['id'])

    new_data = process_logic(request['op'], data)

    stamp = cloud_storage.write(request['id'], new_data)

    return stamp
```

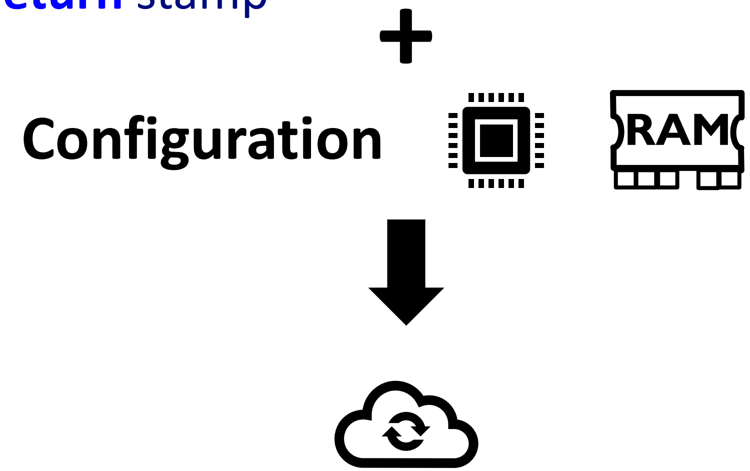
+

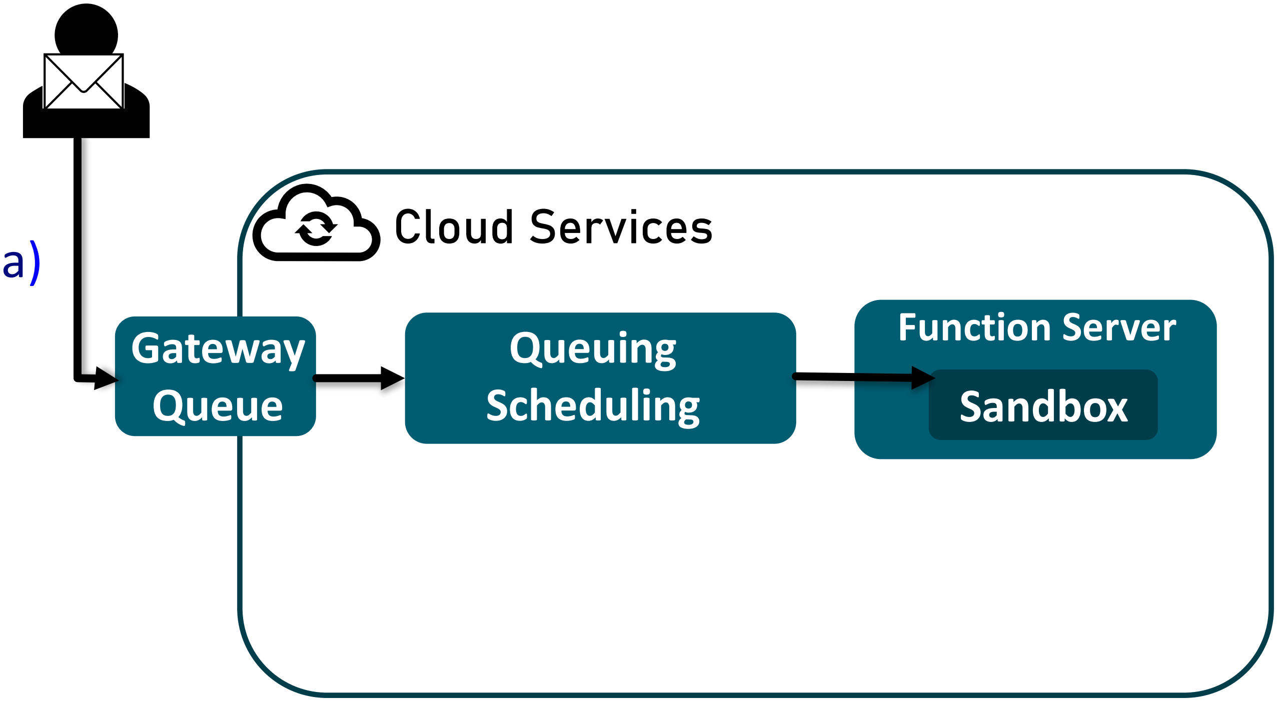
Configuration RAM



How does Function-as-a-Service (FaaS) work?

```
def handler_function(request: dict, context: dict):

    data = cloud_storage.read(request['id'])

    new_data = process_logic(request['op'], data)

    stamp = cloud_storage.write(request['id'], new_data)

    return stamp
```

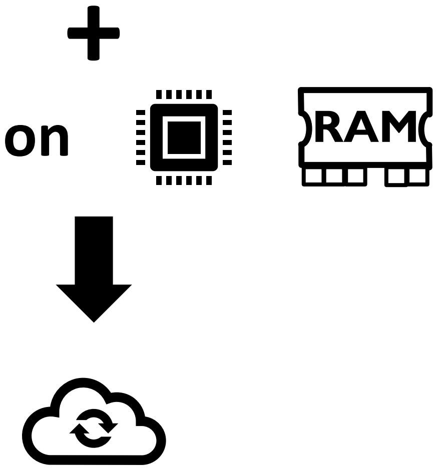


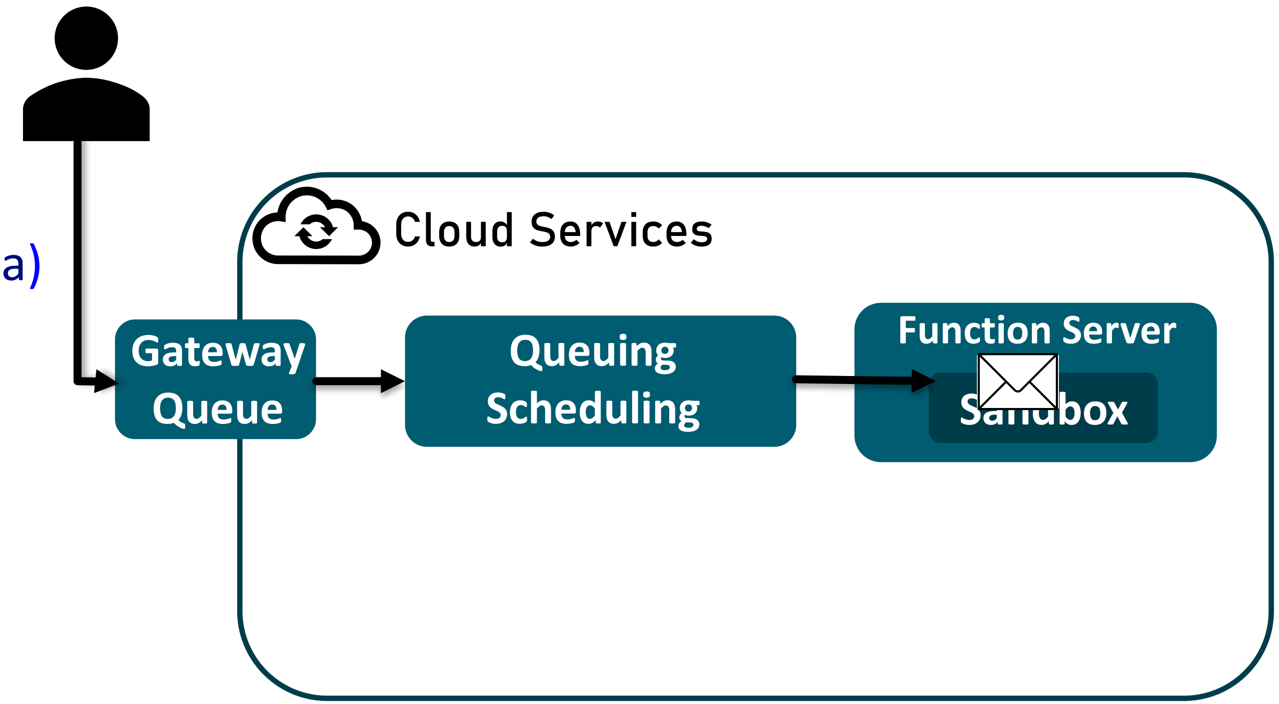


How does Function-as-a-Service (FaaS) work?

```
def handler_function(request: dict, context: dict):

    data = cloud_storage.read(request['id'])

    new_data = process_logic(request['op'], data)

    stamp = cloud_storage.write(request['id'], new_data)

    return stamp
```

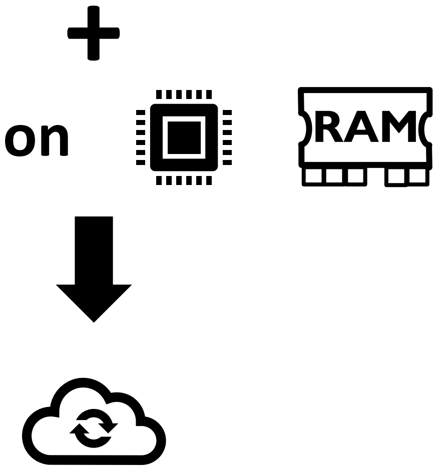
Configuration

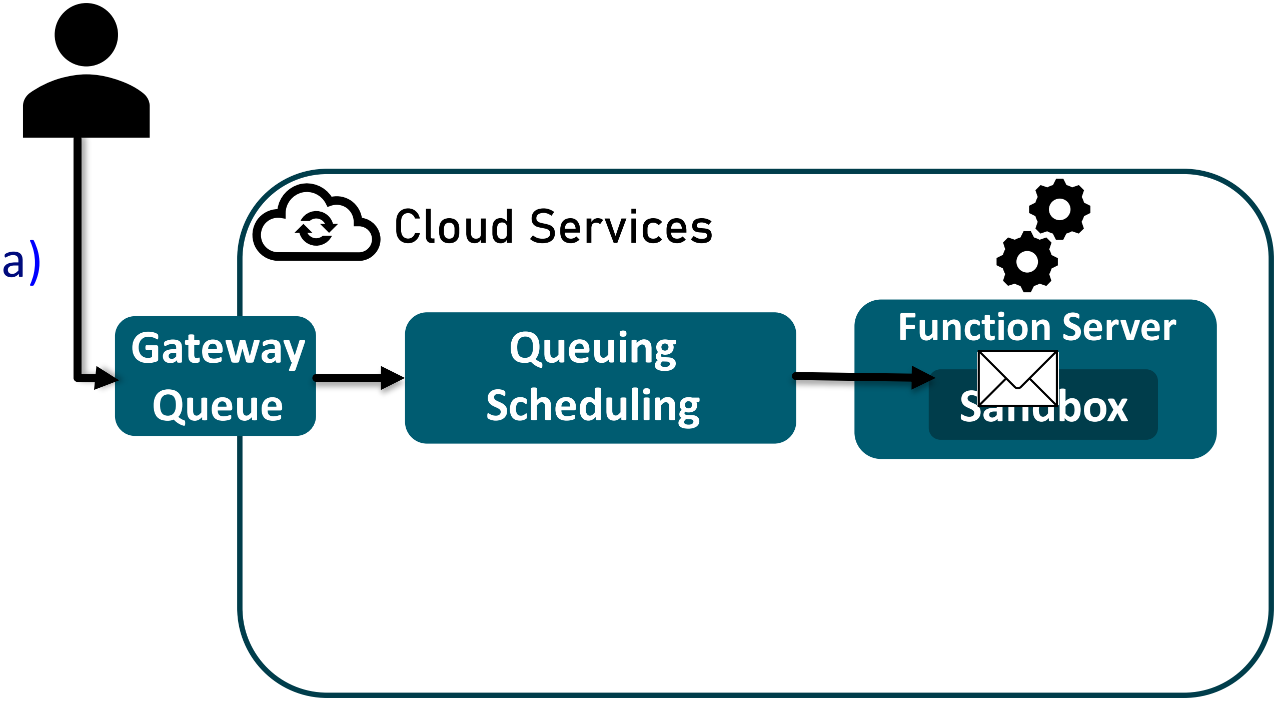




How does Function-as-a-Service (FaaS) work?

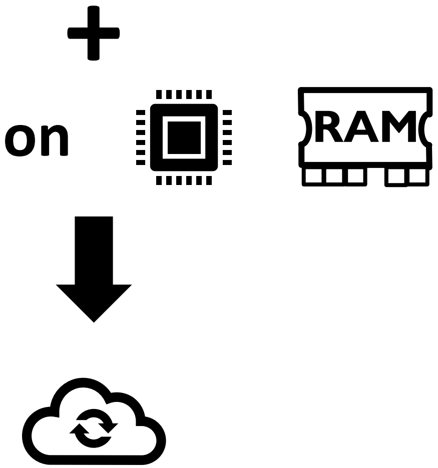
```
def handler_function(request: dict, context: dict):

    data = cloud_storage.read(request['id'])

    new_data = process_logic(request['op'], data)

    stamp = cloud_storage.write(request['id'], new_data)

    return stamp
```

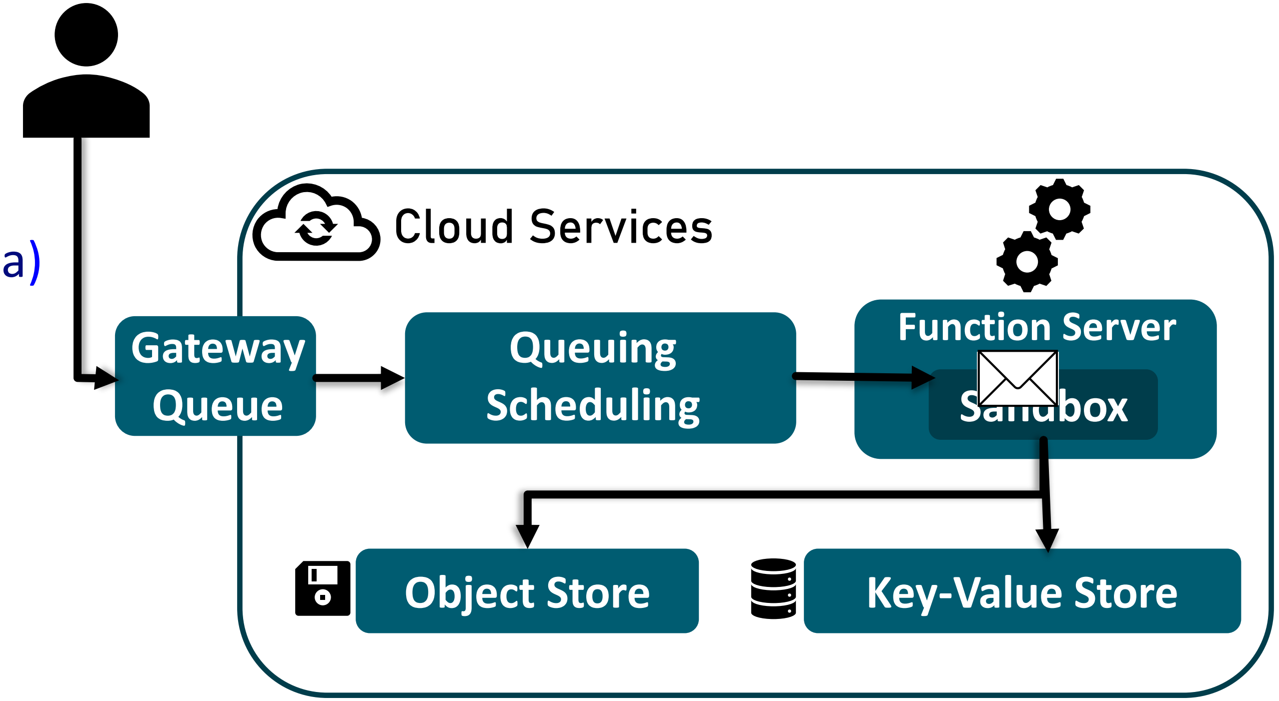




How does Function-as-a-Service (FaaS) work?

```
def handler_function(request: dict, context: dict):

    data = cloud_storage.read(request['id'])

    new_data = process_logic(request['op'], data)

    stamp = cloud_storage.write(request['id'], new_data)

    return stamp
```



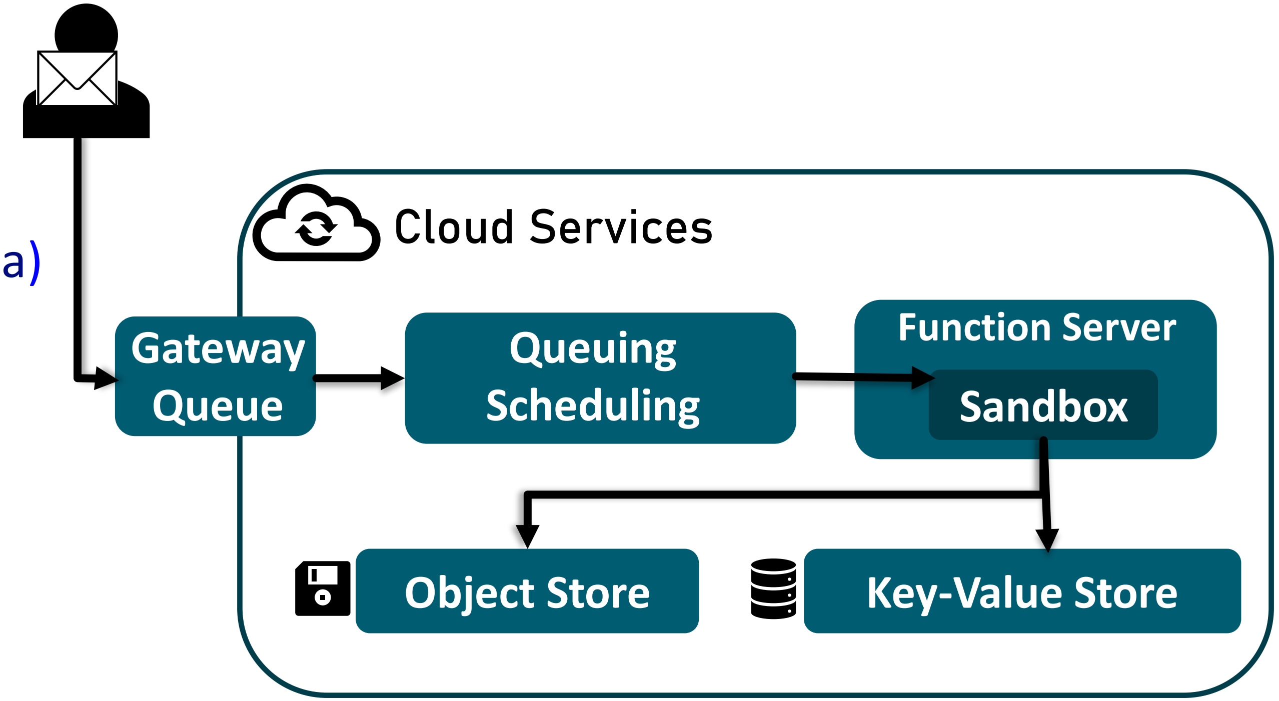


How does Function-as-a-Service (FaaS) work?

```
def handler_function(request: dict, context: dict):

    data = cloud_storage.read(request['id'])

    new_data = process_logic(request['op'], data)

    stamp = cloud_storage.write(request['id'], new_data)

    return stamp
```

+

Configuration RAM



Serverless ZooKeeper: Why and Why Not?

Serverless ZooKeeper: Why and Why Not?

Infrequent Use

Benefit from pay-as-you-go billing.

Serverless ZooKeeper: Why and Why Not?

Infrequent Use

Benefit from pay-as-you-go billing.

High Read-to-write Ratio

Allocate resources accordingly.

Serverless ZooKeeper: Why and Why Not?

Infrequent Use
Benefit from pay-as-you-go billing.

High Read-to-write Ratio
Allocate resources accordingly.

Server-centric Design
ZooKeeper relies on warm TCP connections.

Serverless ZooKeeper: Why and Why Not?

Infrequent Use

Benefit from pay-as-you-go billing.

High Read-to-write Ratio

Allocate resources accordingly.

Server-centric Design

ZooKeeper relies on warm TCP connections.

Complex Data Model

Linearized writes with ordered notifications.

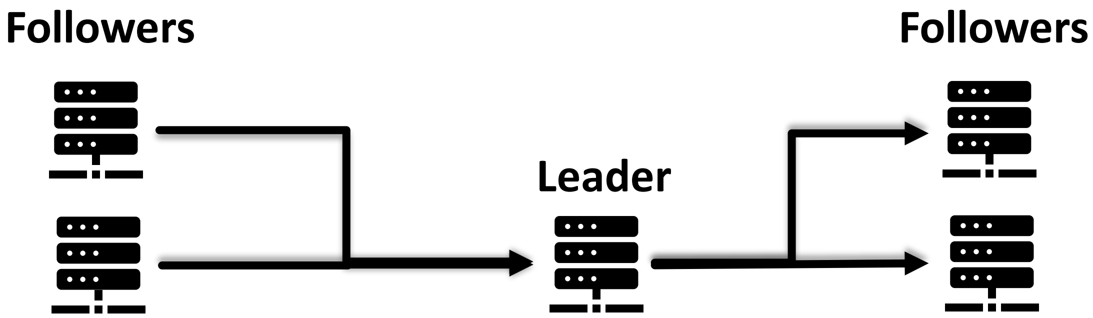
From ZooKeeper to FaaSKeeper

From ZooKeeper to FaaSKeeper

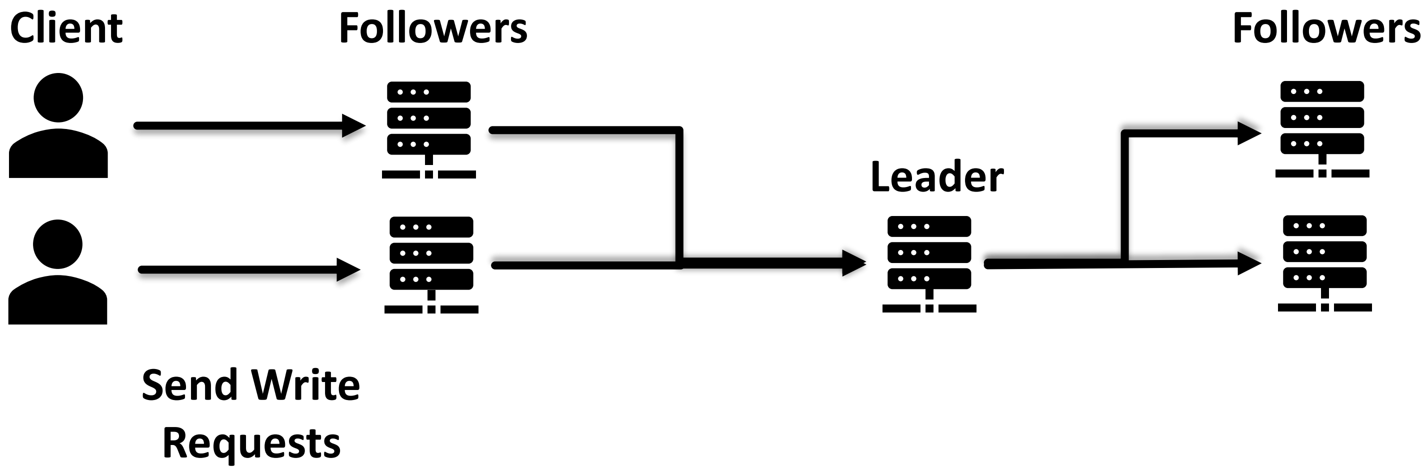
APACHE ZooKeeper™

Leader

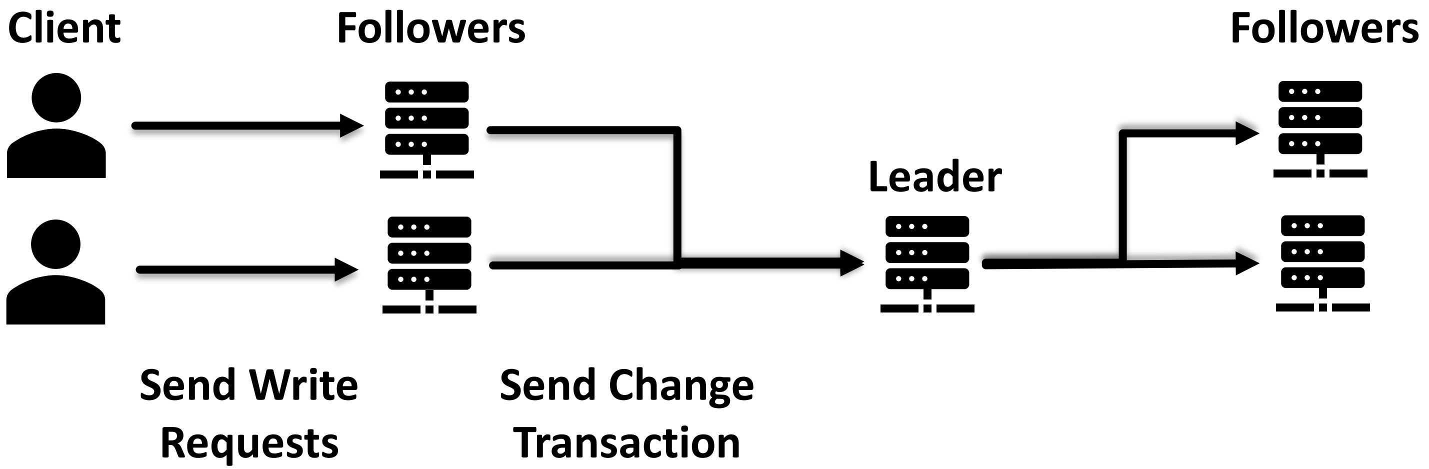
From ZooKeeper to FaaSKeeper



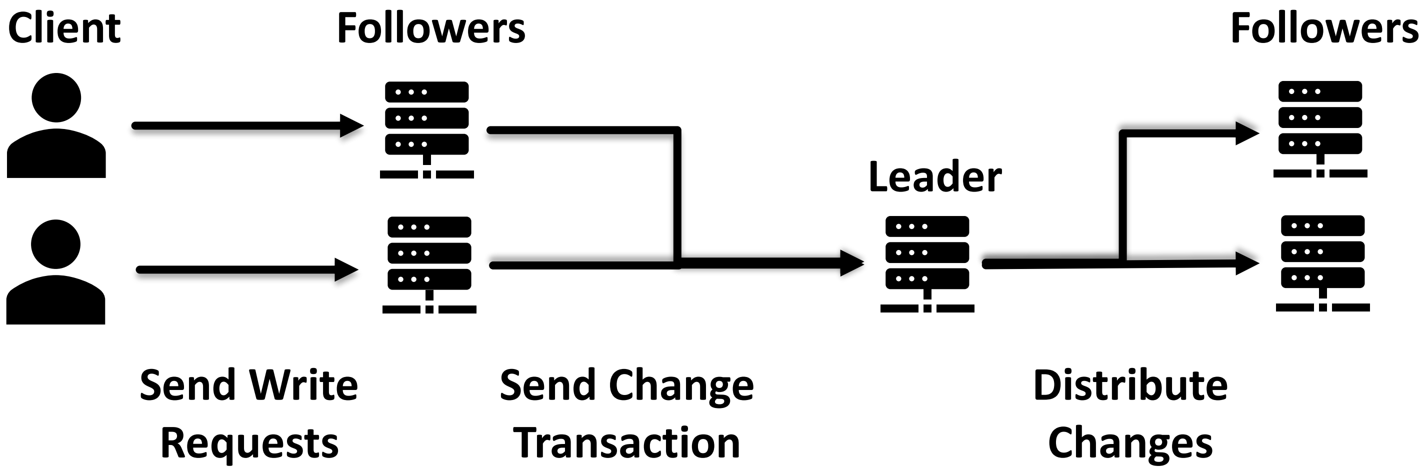
From ZooKeeper to FaaSKeeper



From ZooKeeper to FaaSKeeper



From ZooKeeper to FaaSKeeper



Client

Followers

Followers

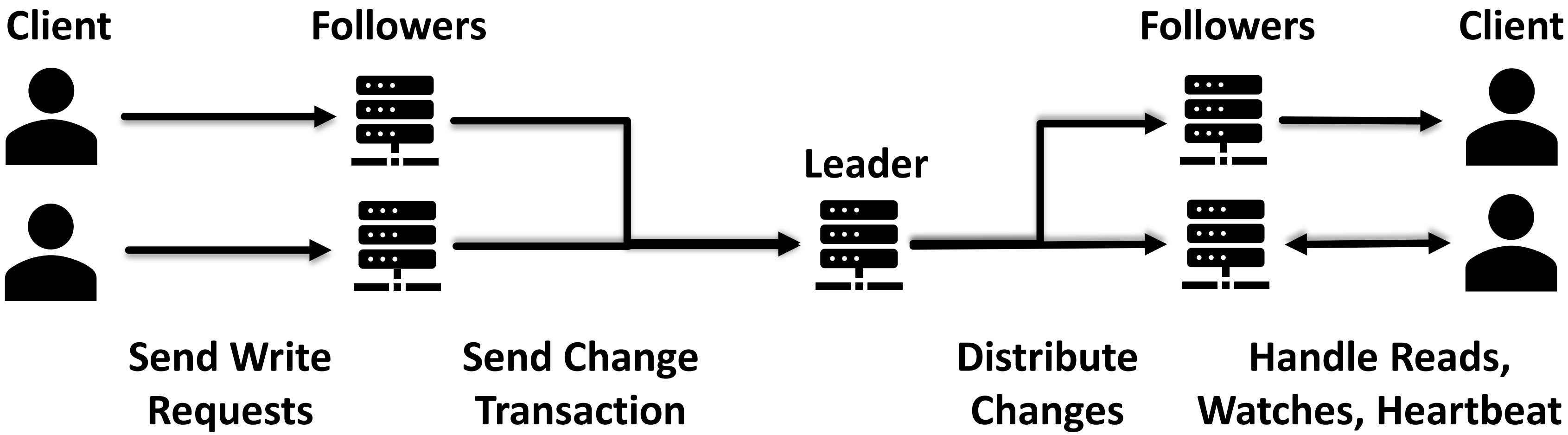
Leader

Send Write Requests

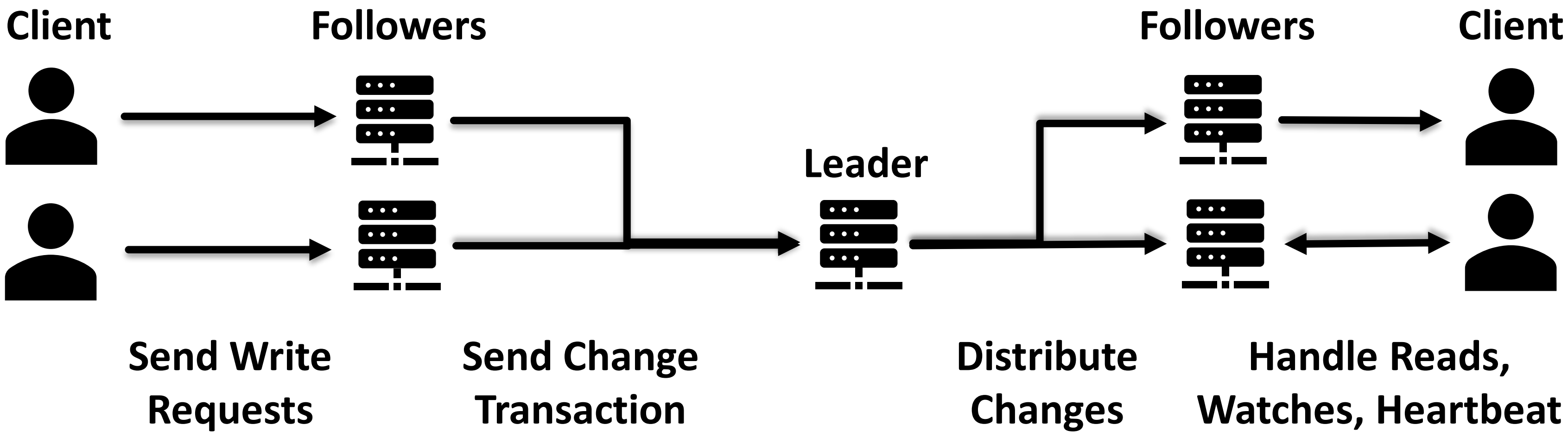
Send Change Transaction

Distribute Changes

From ZooKeeper to FaaSKeeper



From ZooKeeper to FaaSKeeper



Client

Followers

Leader

Followers

Client

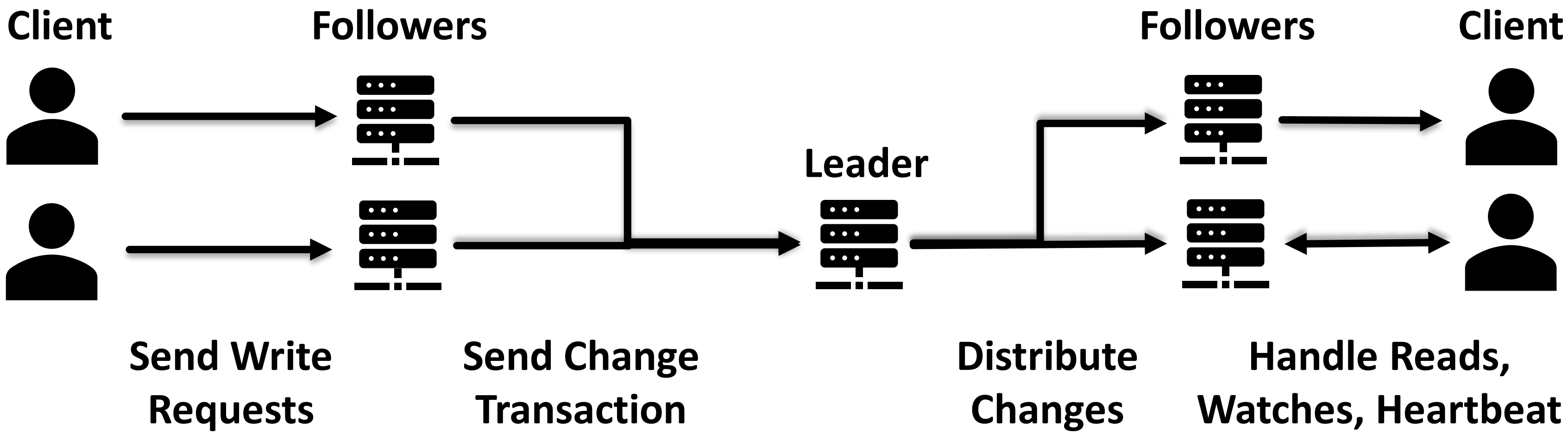
Send Write Requests

Send Change Transaction

Distribute Changes

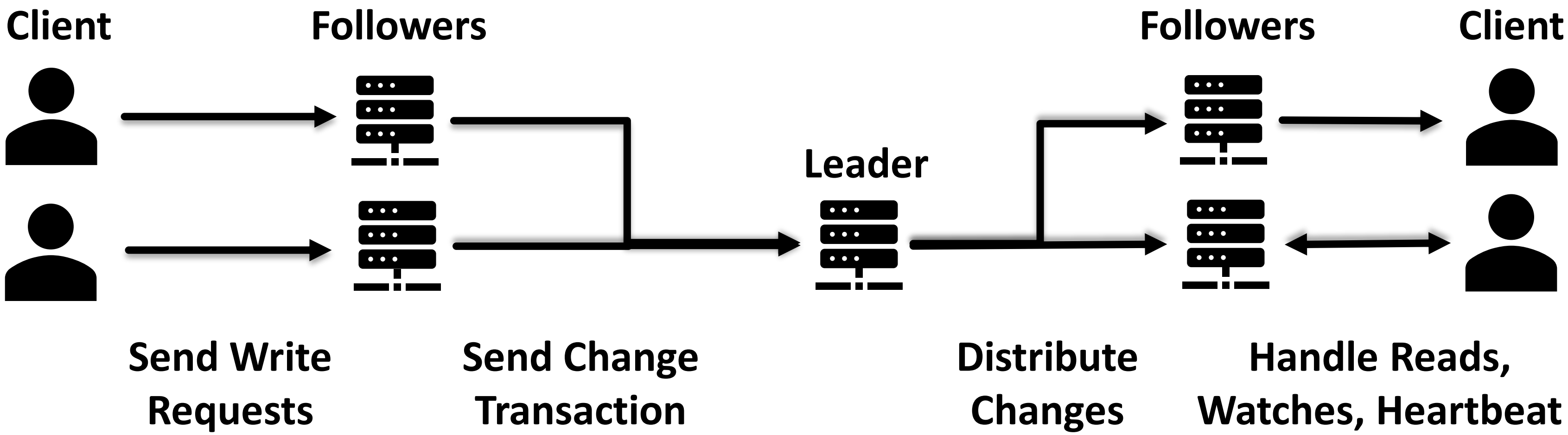
Handle Reads, Watches, Heartbeat

From ZooKeeper to FaaSKeeper



Cloud-Native

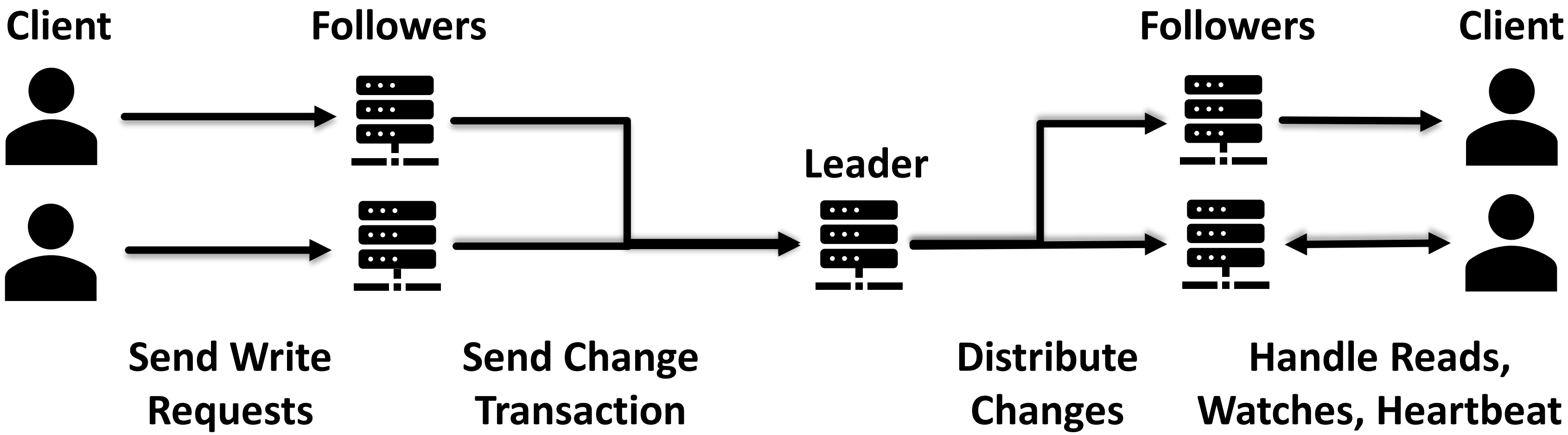
From ZooKeeper to FaaSKeeper



Cloud-Native

100% Serverless

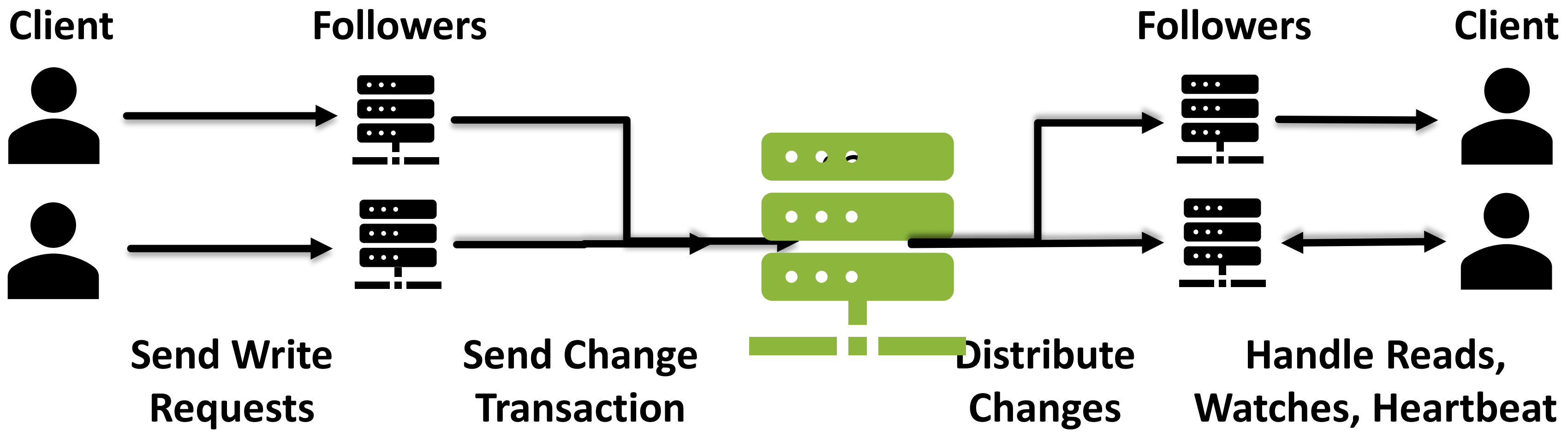
From ZooKeeper to FaaSKeeper



Cloud-Native

100% Serverless

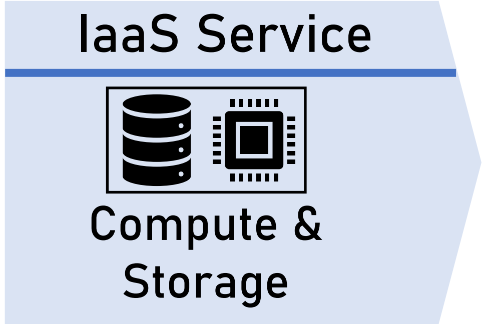
From ZooKeeper to FaaSKeeper



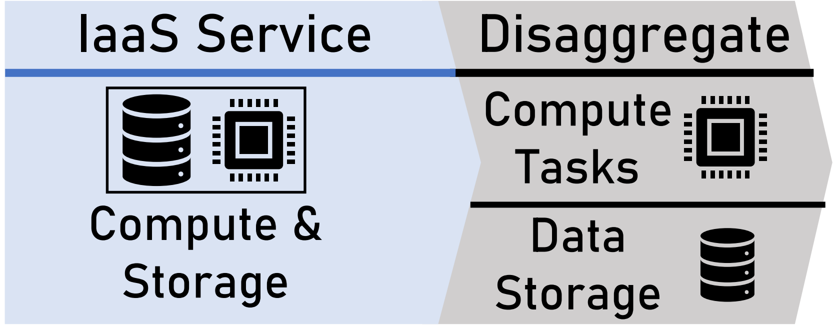
Cloud-Native

100% Serverless

From ZooKeeper to FaaSKeeper



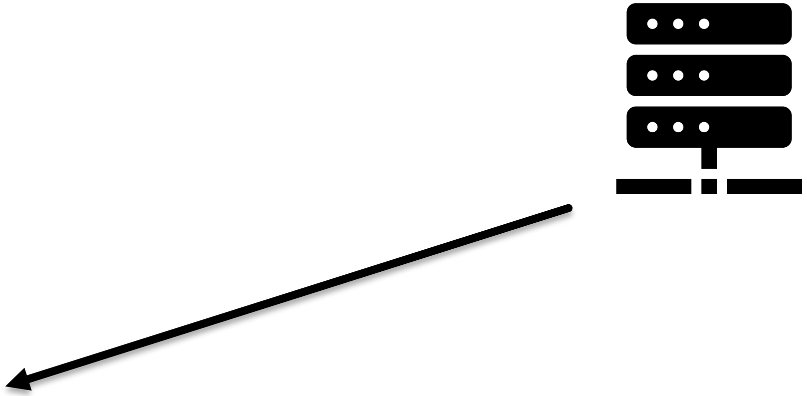
From ZooKeeper to FaaSKeeper



Disaggregate Compute & Storage

Building Serverless Services - Disaggregate

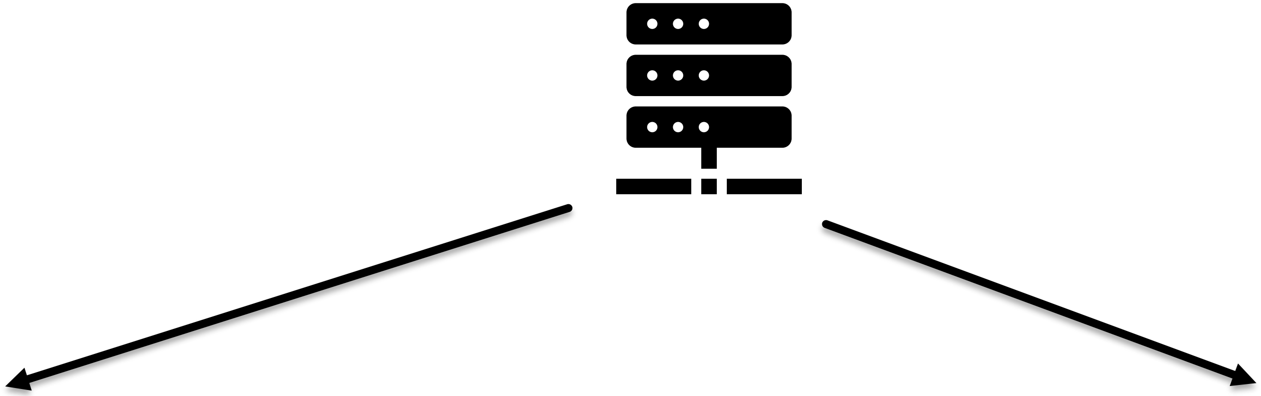
Building Serverless Services - Disaggregate



λ **Writing**

Custom logic

Building Serverless Services - Disaggregate



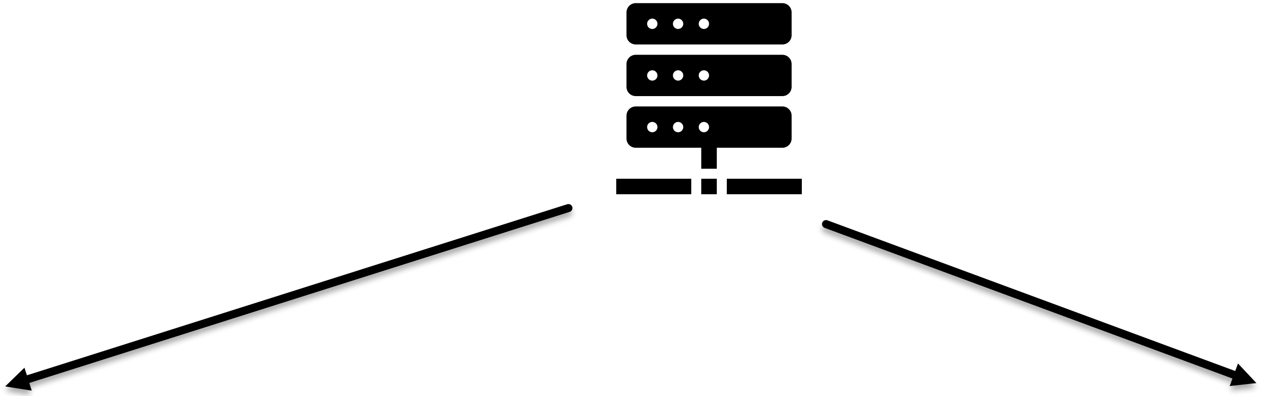
λ **Writing**

Custom logic

Reading

Bandwidth-scalable data access

Building Serverless Services - Disaggregate



λ **Writing**

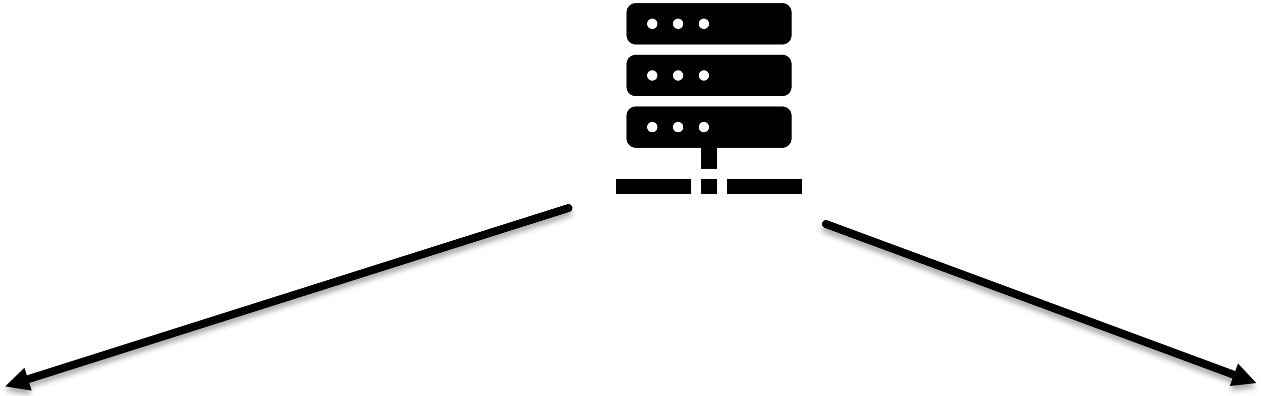
Custom logic

Reading

Bandwidth-scalable data access

10M read requests to DynamoDB: $2.5

Building Serverless Services - Disaggregate



λ **Writing**

Custom logic

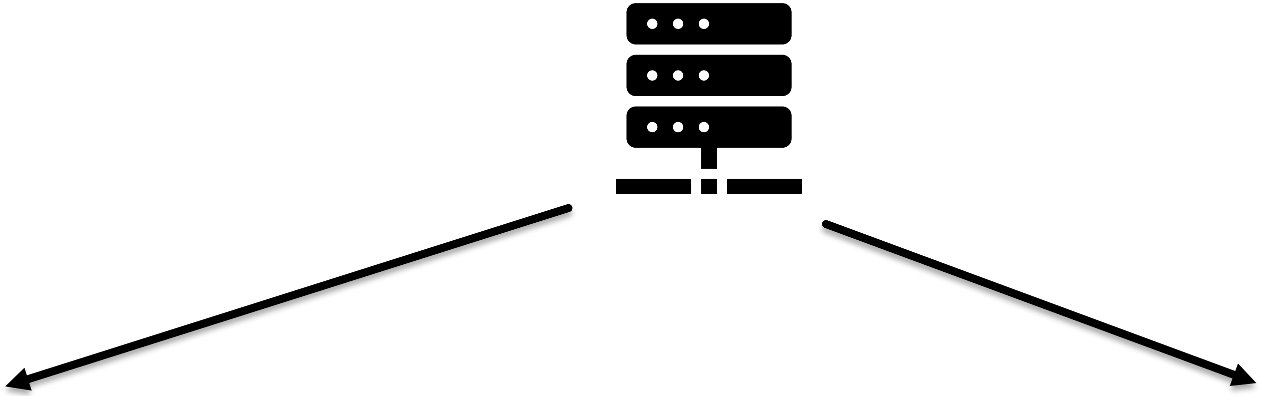
Reading

Bandwidth-scalable data access

10M read requests to DynamoDB: $2.5

10M read requests to S3: $4

Building Serverless Services - Disaggregate



λ **Writing**

Custom logic

Reading

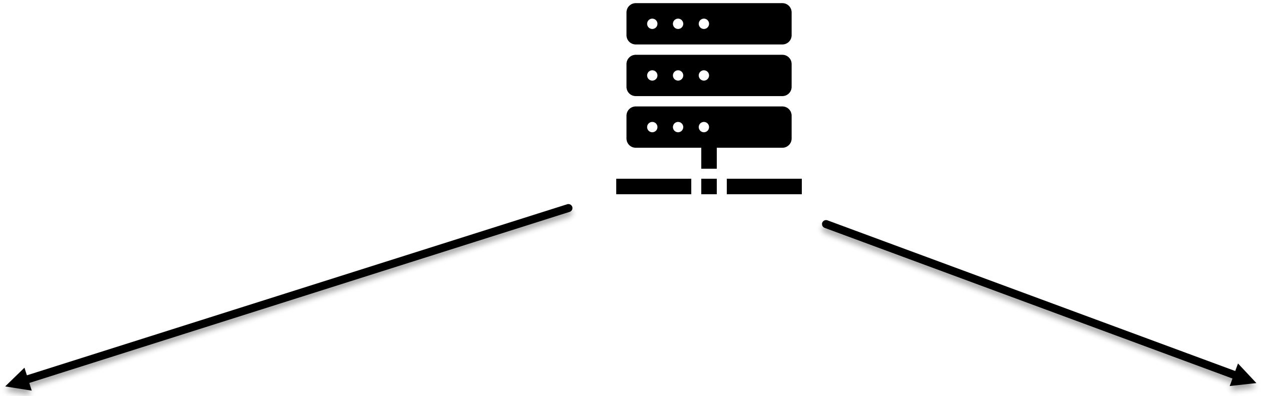
Bandwidth-scalable data access

10M read requests to DynamoDB: $2.5

10M read requests to S3: $4

10M Lambda invocations for 30 ms: $10

Building Serverless Services - Disaggregate



λ **Writing**

Custom logic

Reading

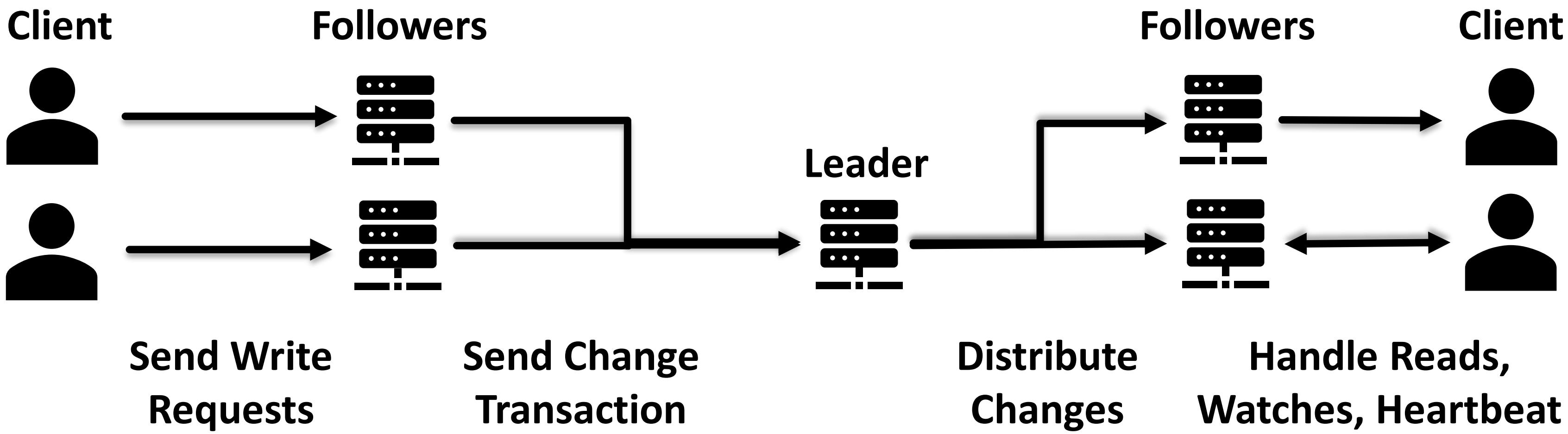
Bandwidth-scalable data access

10M read requests to DynamoDB: $2.5

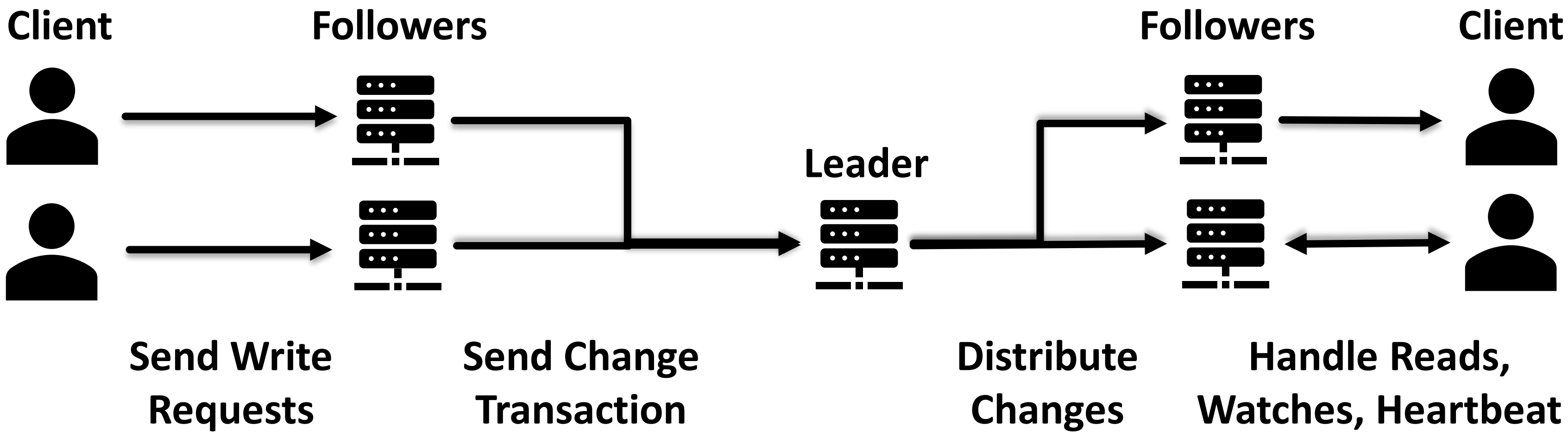
10M read requests to S3: $4

10M Lambda invocations for 30 ms: $10

From ZooKeeper to FaaSKeeper



From ZooKeeper to FaaSKeeper


Client

Followers

Leader

Followers

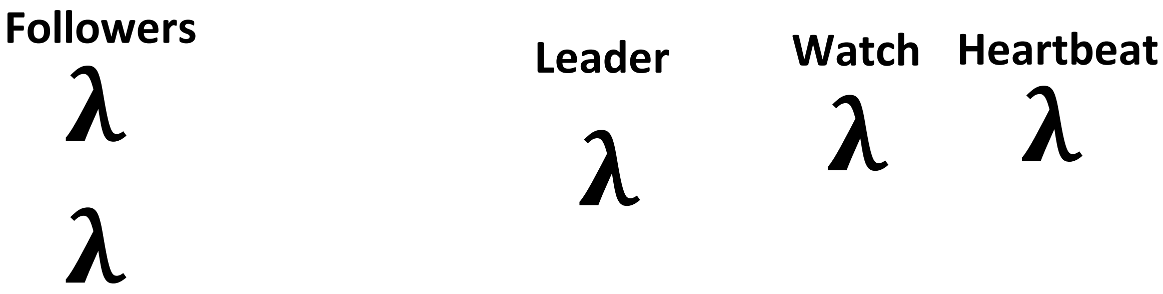
Client

Send Write Requests

Send Change Transaction

Distribute Changes

Handle Reads, Watches, Heartbeat


Followers

λ

λ

Leader

λ

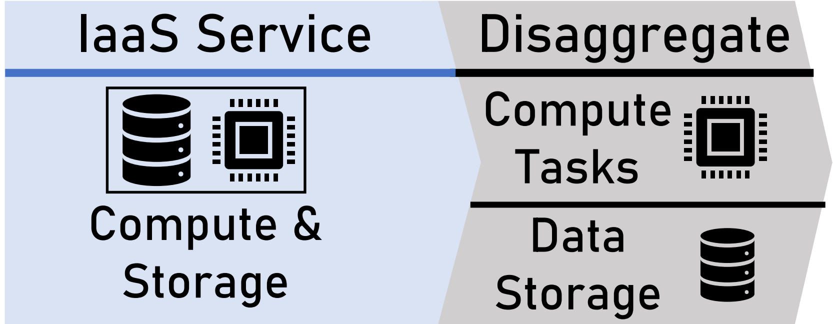
Watch

λ

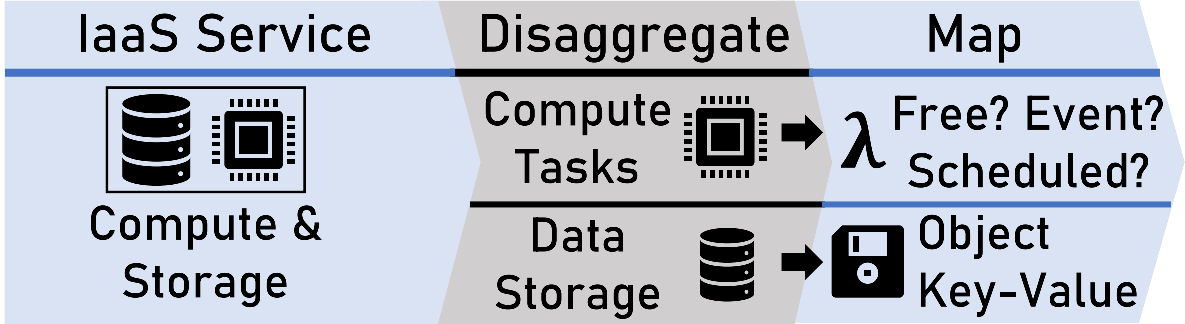
Heartbeat

λ

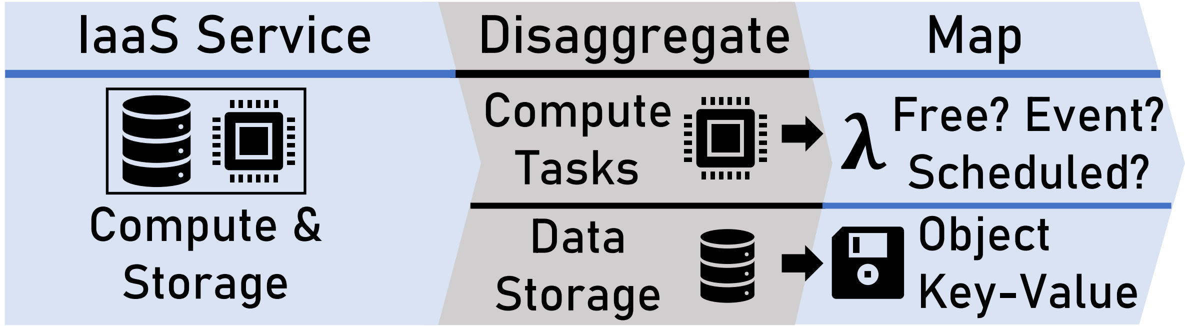
From ZooKeeper to FaaSKeeper



From ZooKeeper to FaaSKeeper



From ZooKeeper to FaaSKeeper



Map to Cloud Services

Building Serverless Services – Map Storage

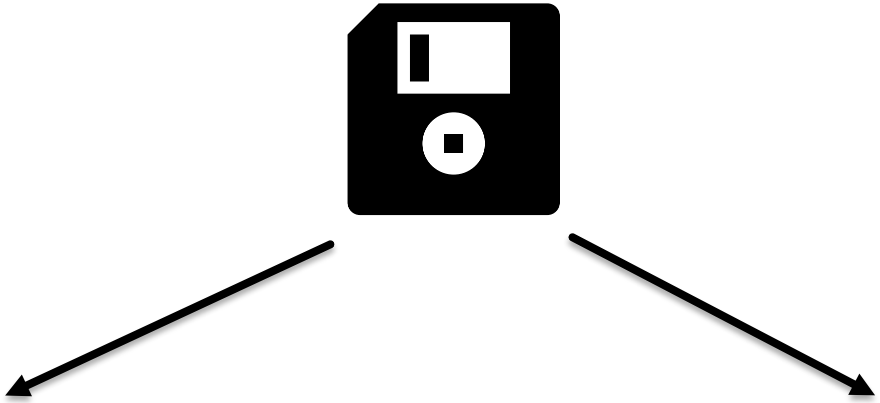
Building Serverless Services – Map Storage

System

Mostly write operations

Synchronization primitives

Building Serverless Services – Map Storage



System

Mostly write operations

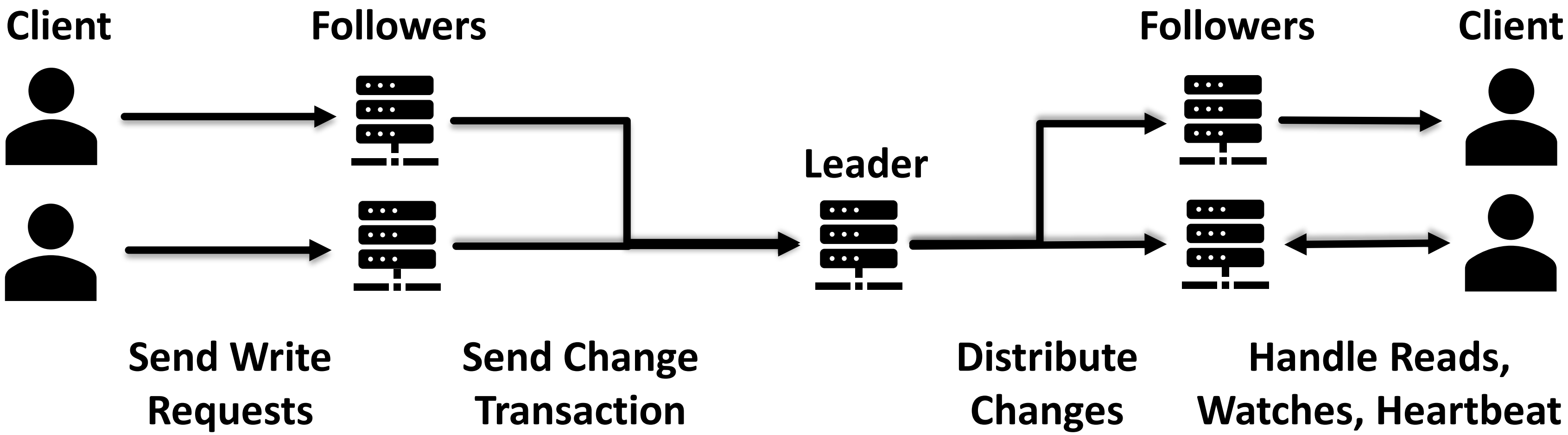
Synchronization primitives

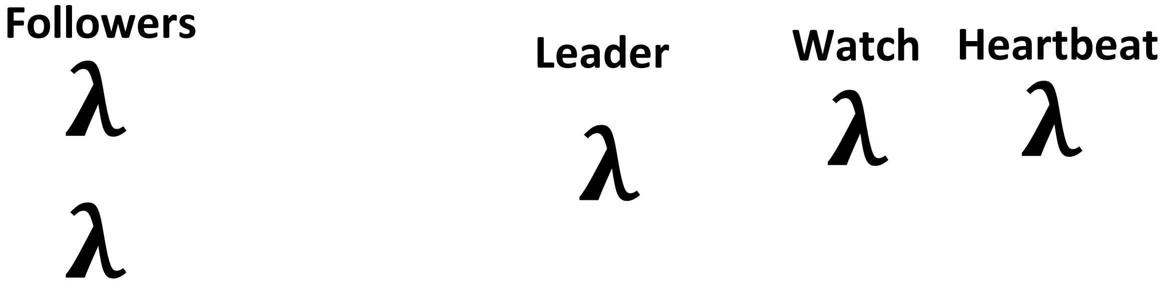
User

Frequent reading

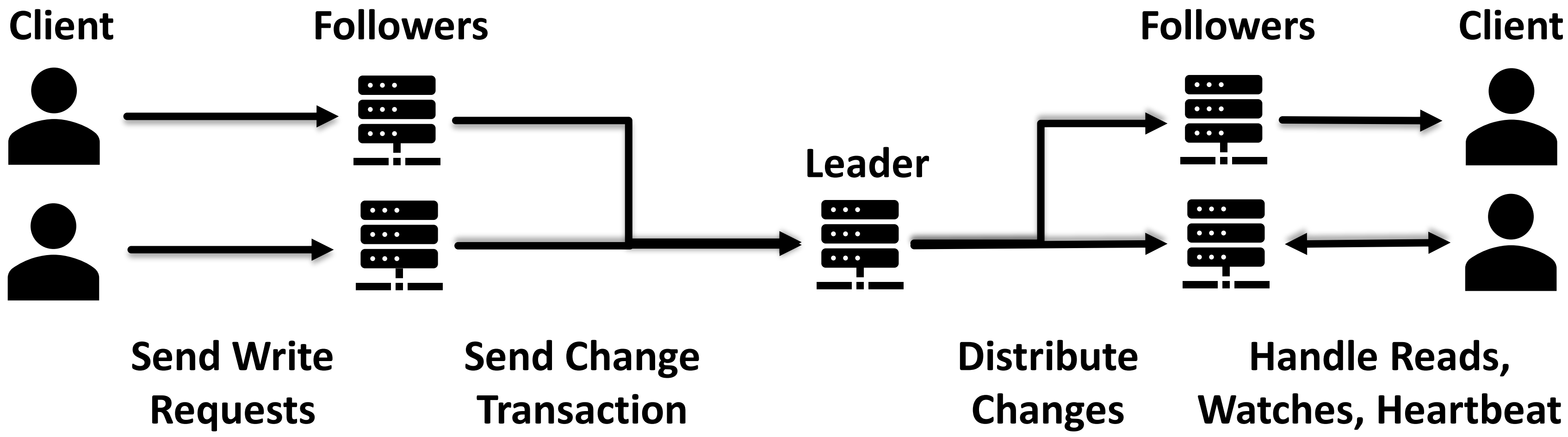
Cost efficient

From ZooKeeper to FaaSKeeper





From ZooKeeper to FaaSKeeper


Client

Followers

Leader

Followers

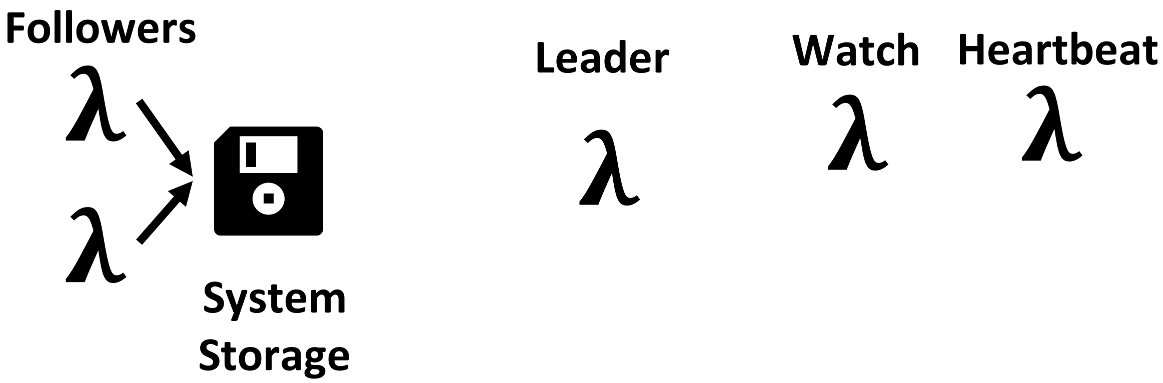
Client

Send Write Requests

Send Change Transaction

Distribute Changes

Handle Reads, Watches, Heartbeat


Followers

λ

λ

System Storage

Leader

λ

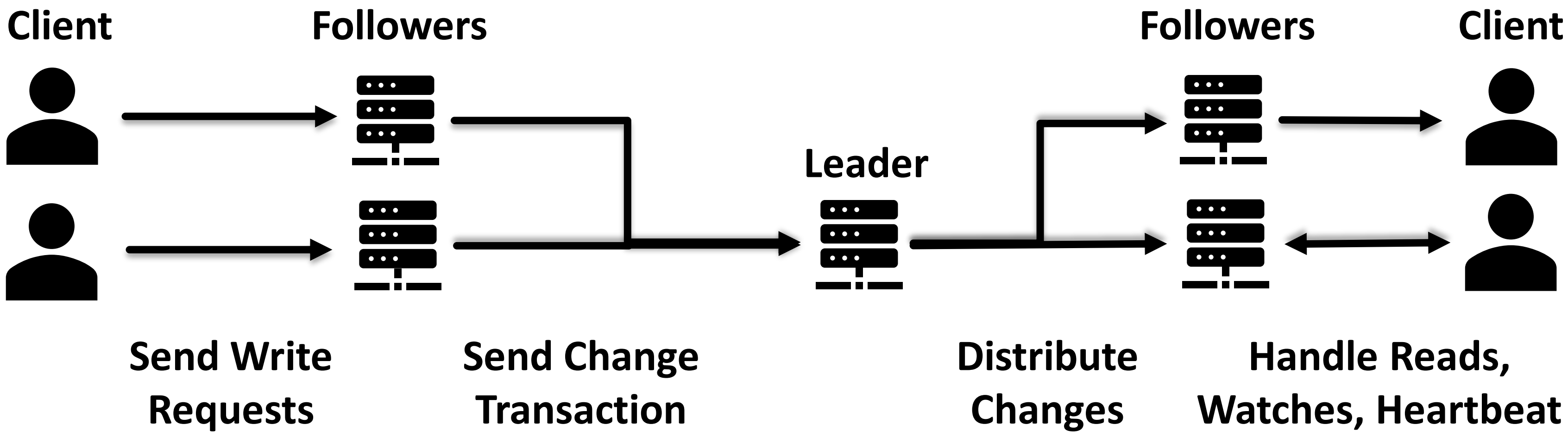
Watch

λ

Heartbeat

λ

From ZooKeeper to FaaSKeeper



Client

Followers

Leader

Followers

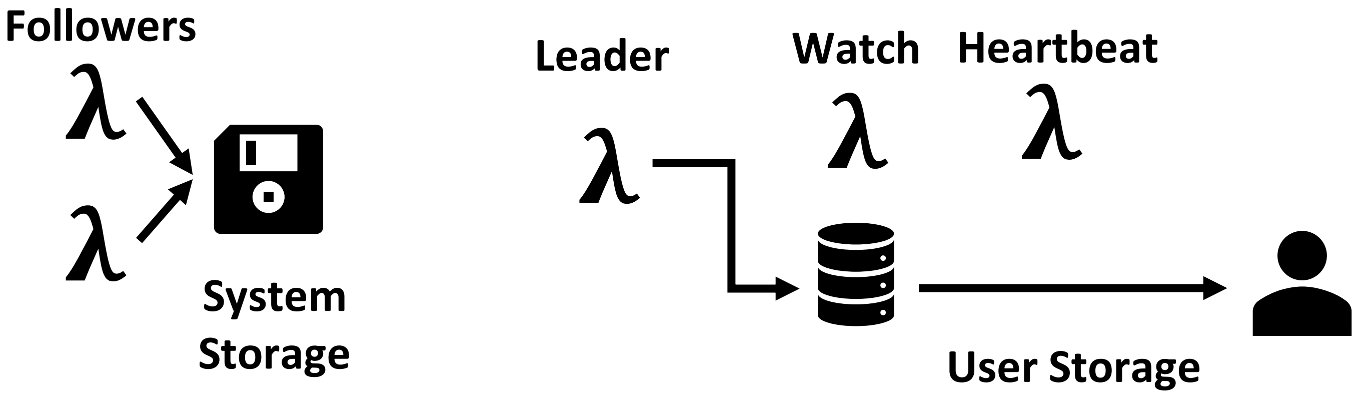
Client

Send Write Requests

Send Change Transaction

Distribute Changes

Handle Reads, Watches, Heartbeat



Followers

System Storage

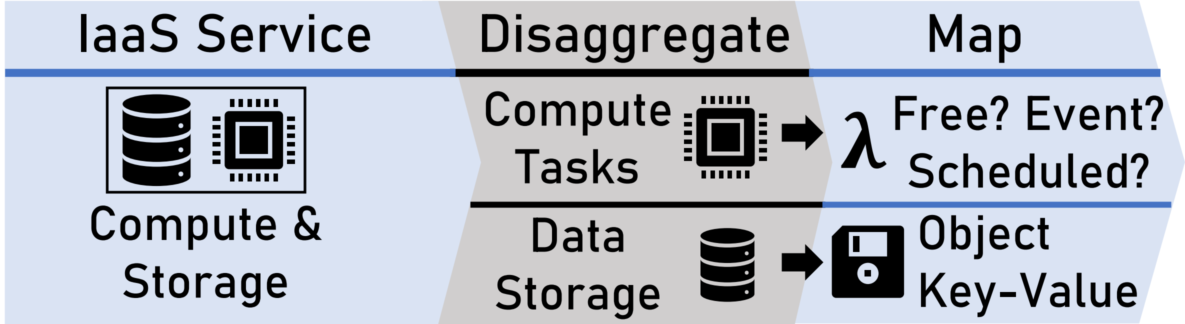
Leader

Watch

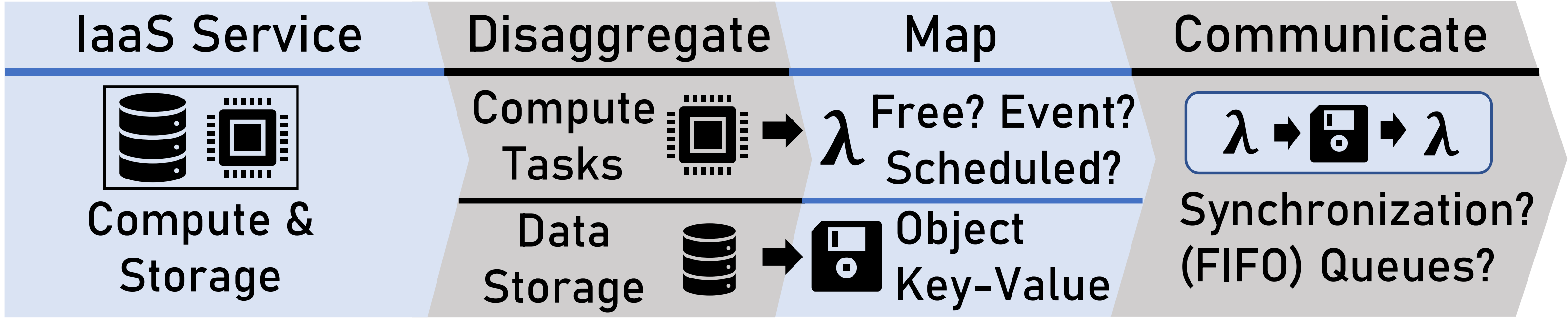
Heartbeat

User Storage

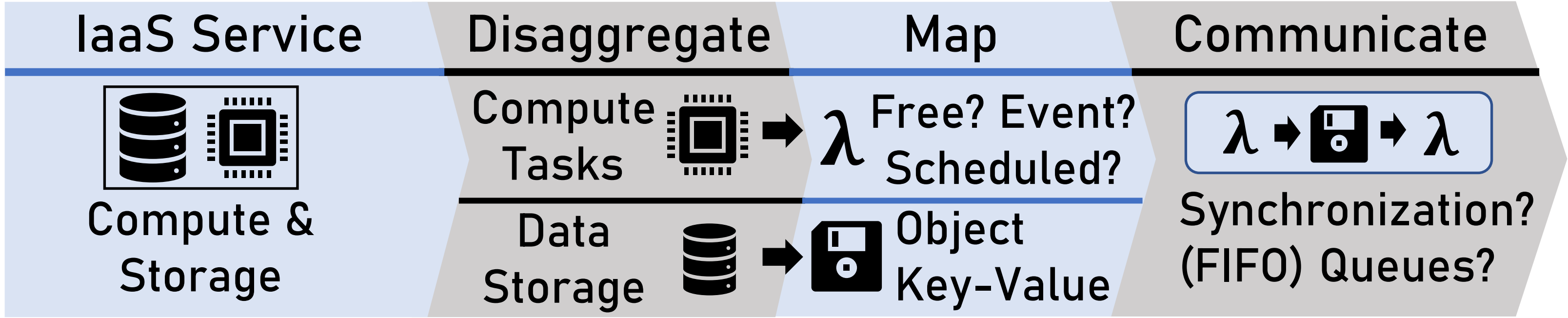
From ZooKeeper to FaaSKeeper



From ZooKeeper to FaaSKeeper

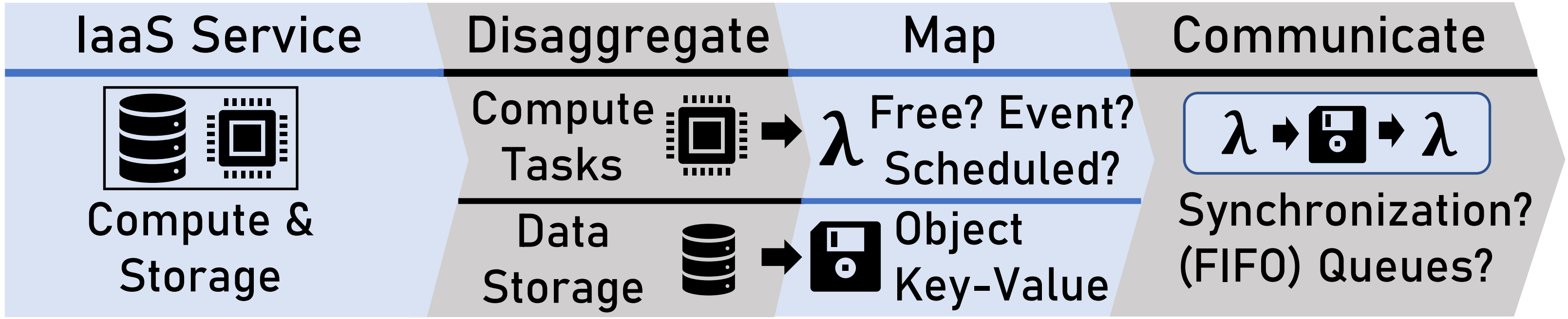


From ZooKeeper to FaaSKeeper



Event Ordering on Client

From ZooKeeper to FaaSKeeper



IaaS Service

Compute & Storage

Disaggregate

Compute Tasks →

Data Storage →

Map

λ Free? Event? Scheduled?

Object Key-Value

Communicate

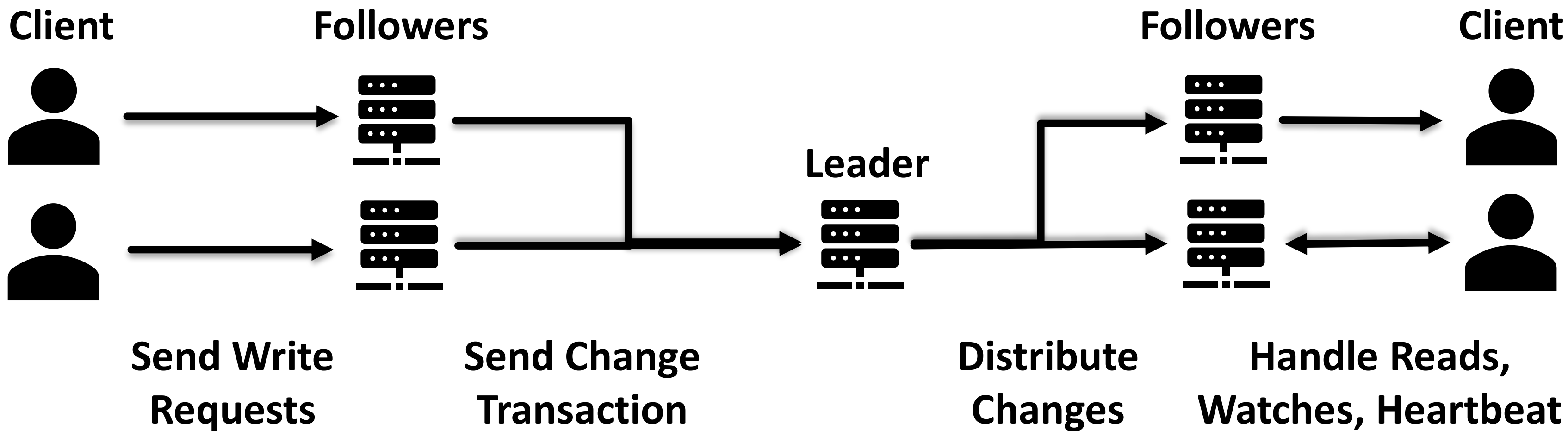
λ → 🖫 → λ

Synchronization? (FIFO) Queues?

Event Ordering on Client

Epoch Counters for Watches

From ZooKeeper to FaaSKeeper



Client

Followers

Leader

Followers

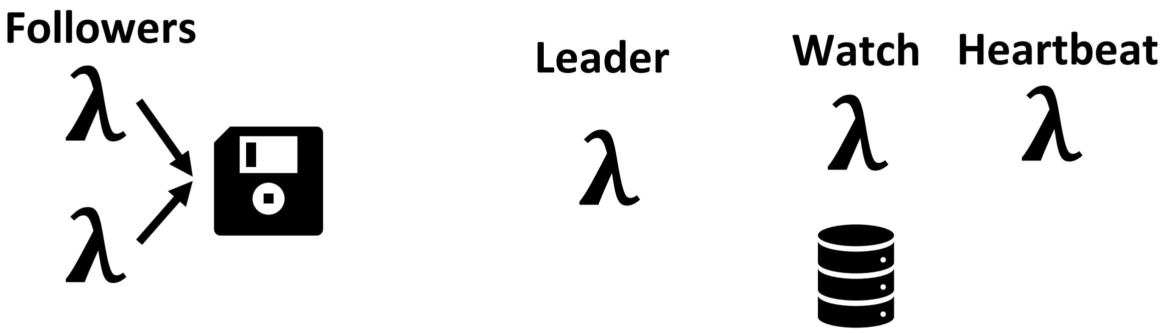
Client

Send Write Requests

Send Change Transaction

Distribute Changes

Handle Reads, Watches, Heartbeat



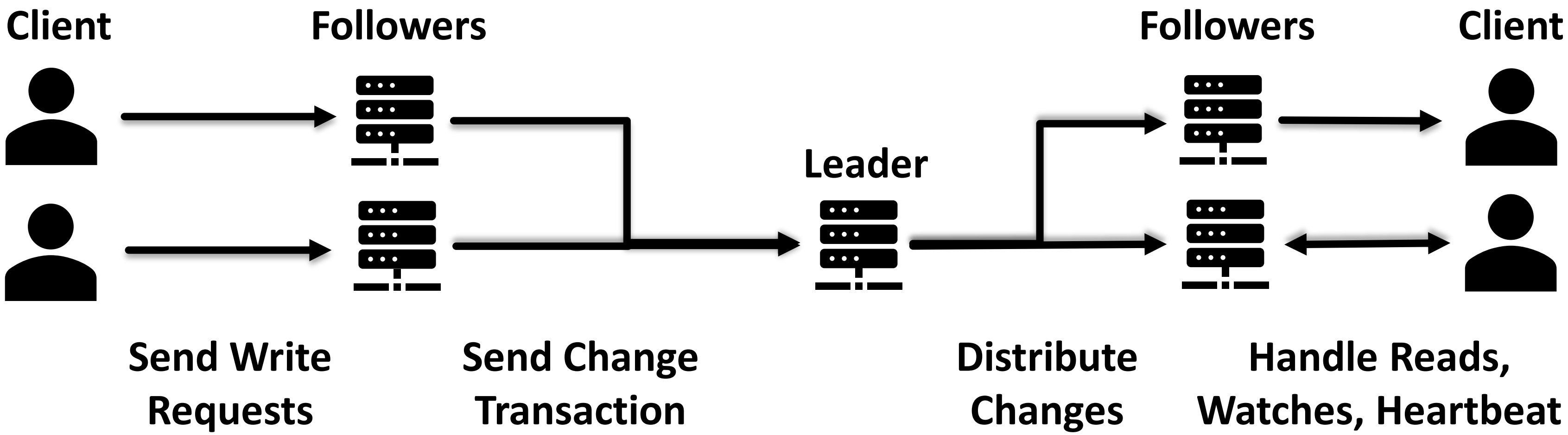
Followers

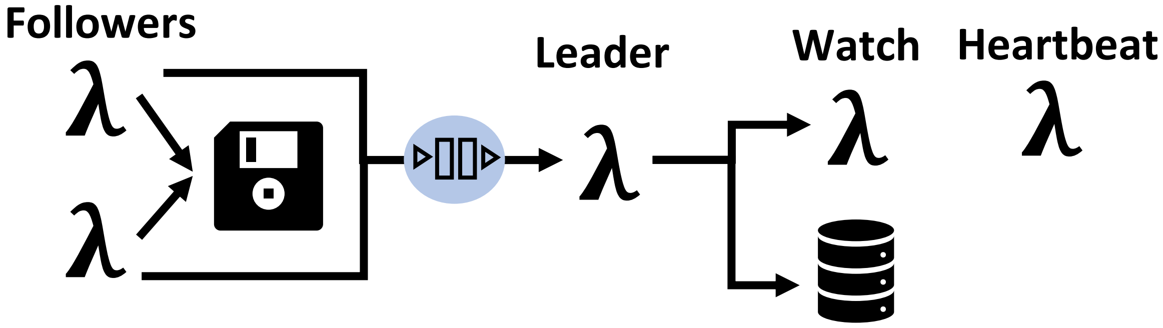
Leader

Watch

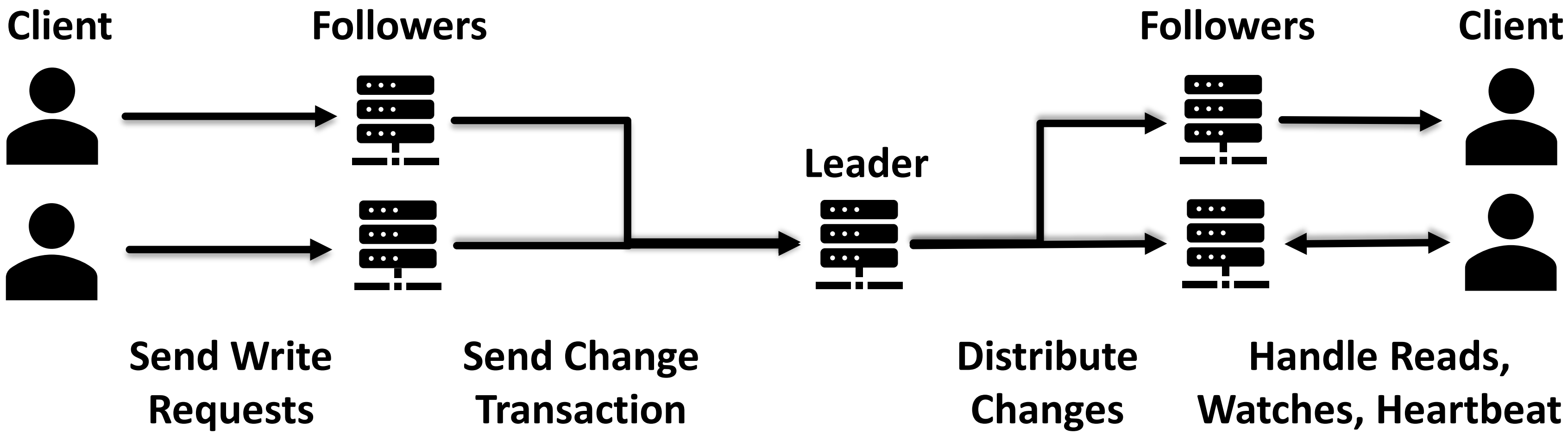
Heartbeat

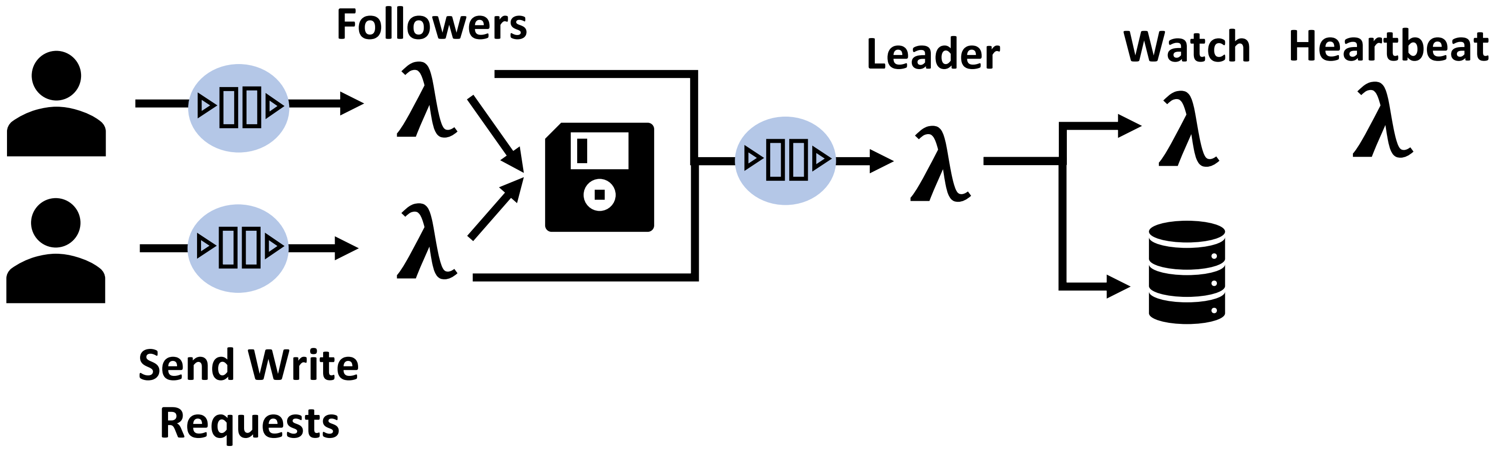
From ZooKeeper to FaaSKeeper



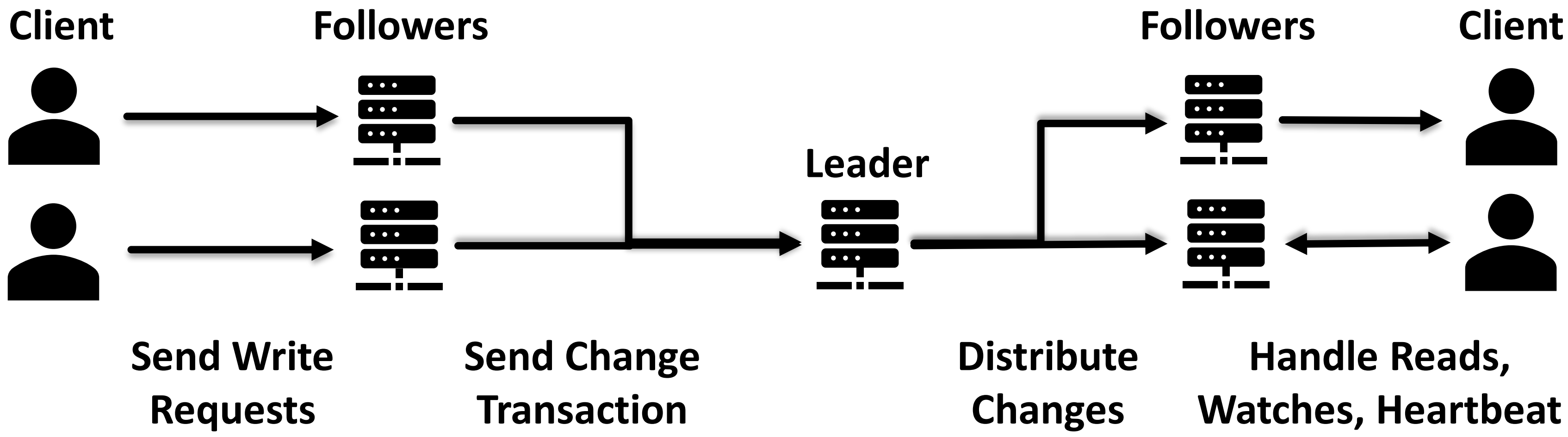


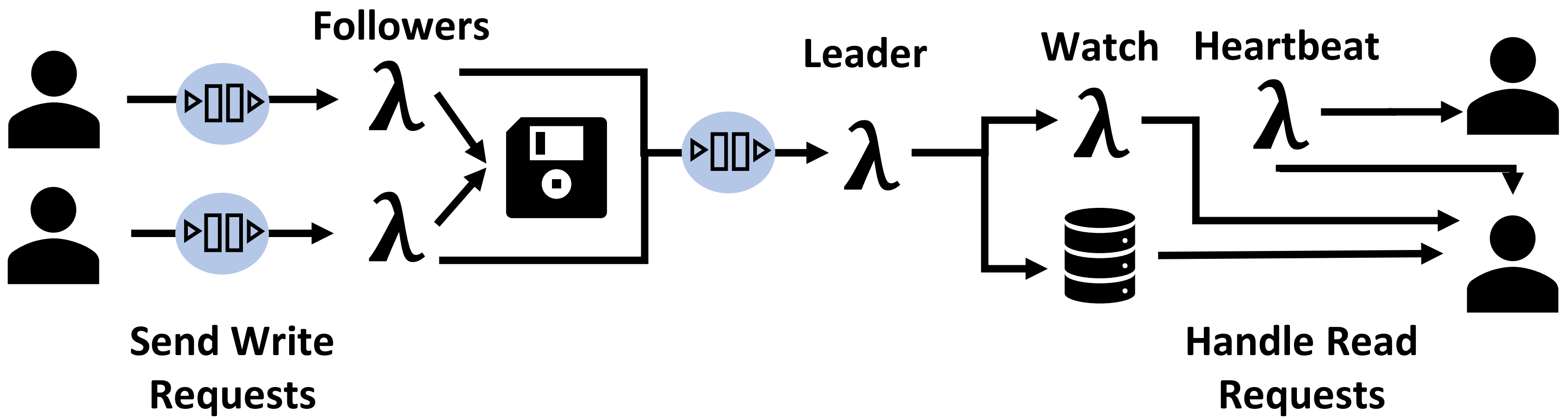
From ZooKeeper to FaaSKeeper



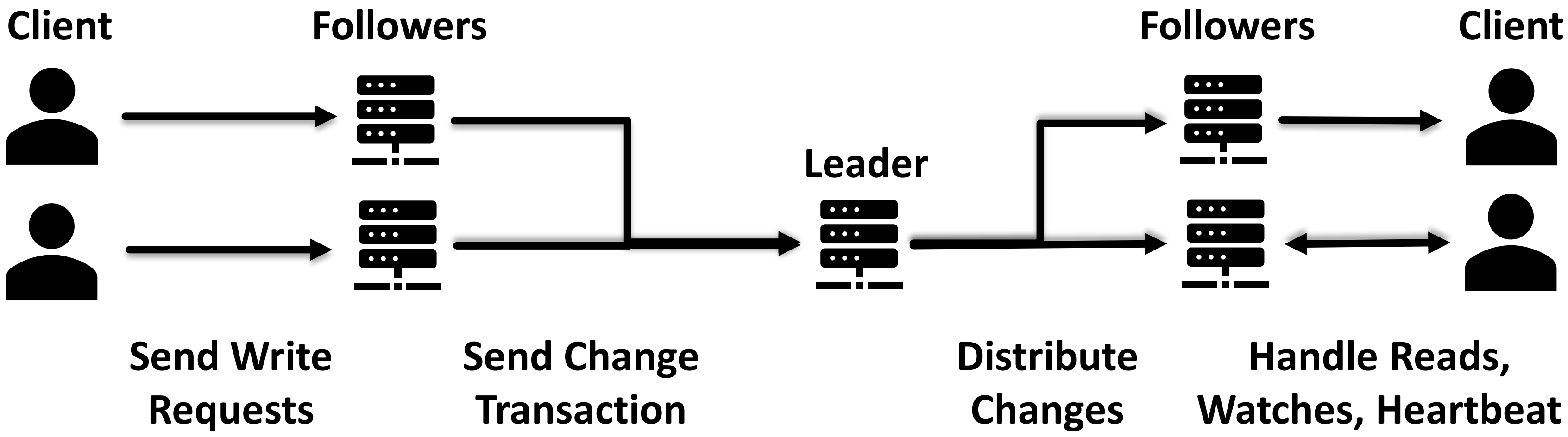


From ZooKeeper to FaaSKeeper





From ZooKeeper to FaaSKeeper



Client

Followers

Leader

Followers

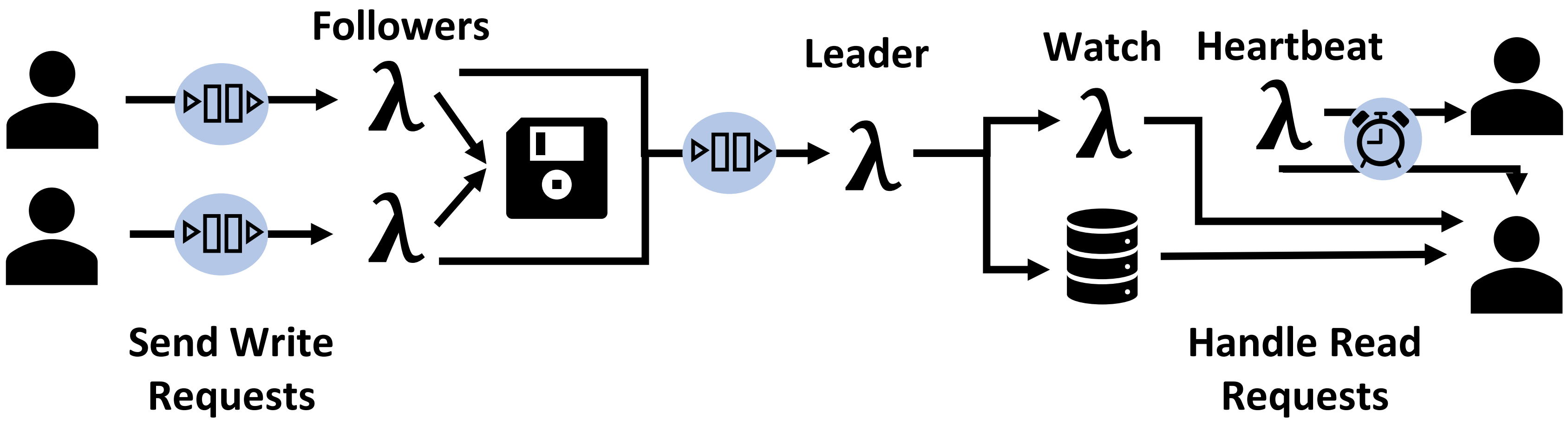
Client

Send Write Requests

Send Change Transaction

Distribute Changes

Handle Reads, Watches, Heartbeat



Followers

Leader

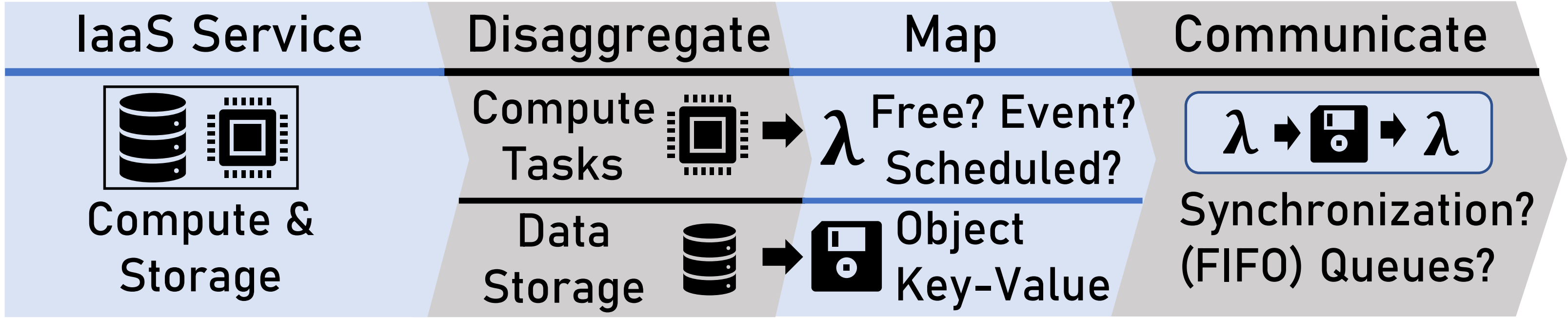
Watch

Heartbeat

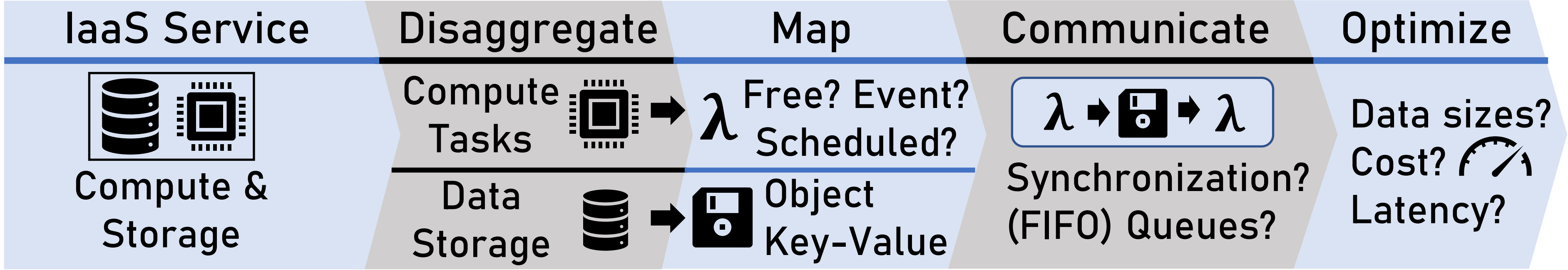
Send Write Requests

Handle Read Requests

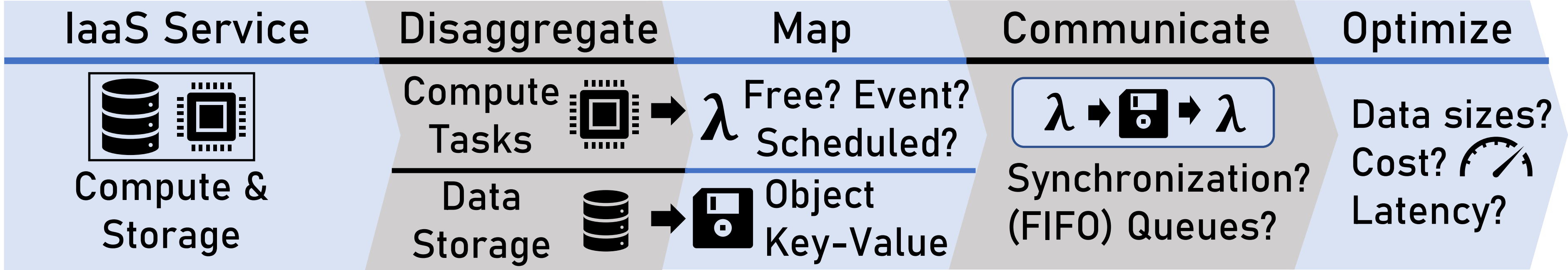
From ZooKeeper to FaaSKeeper



From ZooKeeper to FaaSKeeper



From ZooKeeper to FaaSKeeper



IaaS Service

Compute & Storage

Disaggregate

Compute Tasks

Data Storage

Map

λ Free? Event? Scheduled?

Object Key-Value

Communicate

λ → → λ

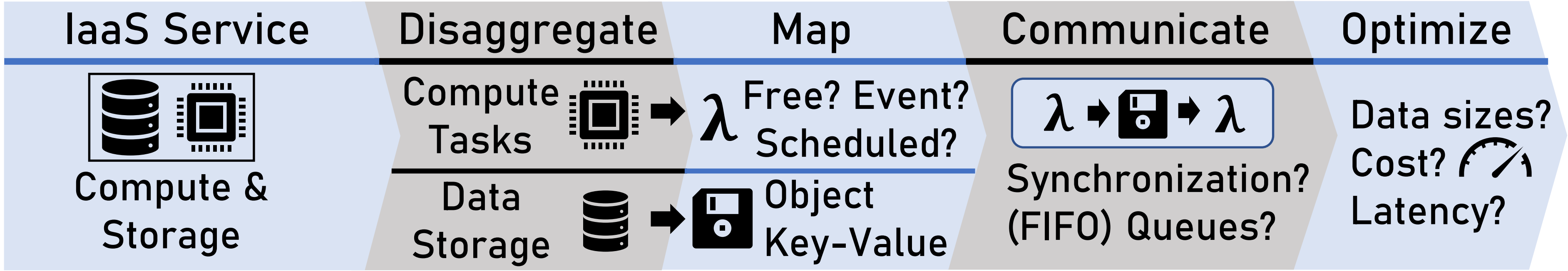
Synchronization? (FIFO) Queues?

Optimize

Data sizes? Cost? Latency?

Hybrid Storage

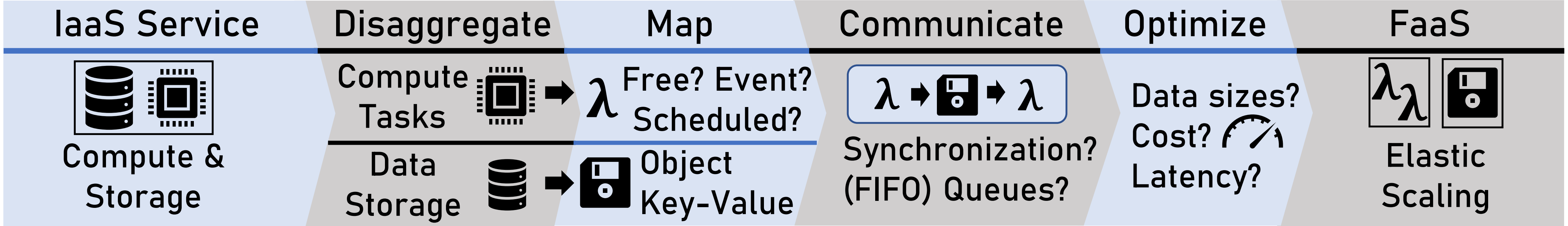
From ZooKeeper to FaaSKeeper



Hybrid Storage

Decoupled Heartbeats

From ZooKeeper to FaaSKeeper



Hybrid Storage

Decoupled Heartbeats

Consistency Model

Consistency Model

❶ **Atomicity**

Atomic updates to cloud storage

Consistency Model

APACHE ZooKeeper™

FAASKEEPER

1 **Atomicity** — **Atomic updates to cloud storage**

2 **Linearized Writes** — **Single leader with ordered queues**

Consistency Model

APACHE ZooKeeper™	FAASKEEPER
1 Atomicity	Atomic updates to cloud storage
2 Linearized Writes	Single leader with ordered queues
3 Single System Image	Strongly consistent cloud storage

Consistency Model

APACHE ZooKeeper™	FAASKEEPER
1 Atomicity	Atomic updates to cloud storage
2 Linearized Writes	Single leader with ordered queues
3 Single System Image	Strongly consistent cloud storage
4 Ordered Notifications	Watch notifications with epoch counters

Implementation & Evaluation

Implementation & Evaluation

Proof of Concept Implementation

1,350 LoC for FaaSKeeper

1,400 LoC for client library

Implementation & Evaluation

Proof of Concept Implementation

1,350 LoC for FaaSKeeper

1,400 LoC for client library

aws

Google Cloud

Implementation & Evaluation

Proof of Concept Implementation
1,350 LoC for FaaSKeeper
1,400 LoC for client library

aws

Google Cloud

1. **How does read performance compare to ZooKeeper?**

2. **How does write performance compare to ZooKeeper?**

Implementation & Evaluation

Proof of Concept Implementation
1,350 LoC for FaaSKeeper
1,400 LoC for client library

aws

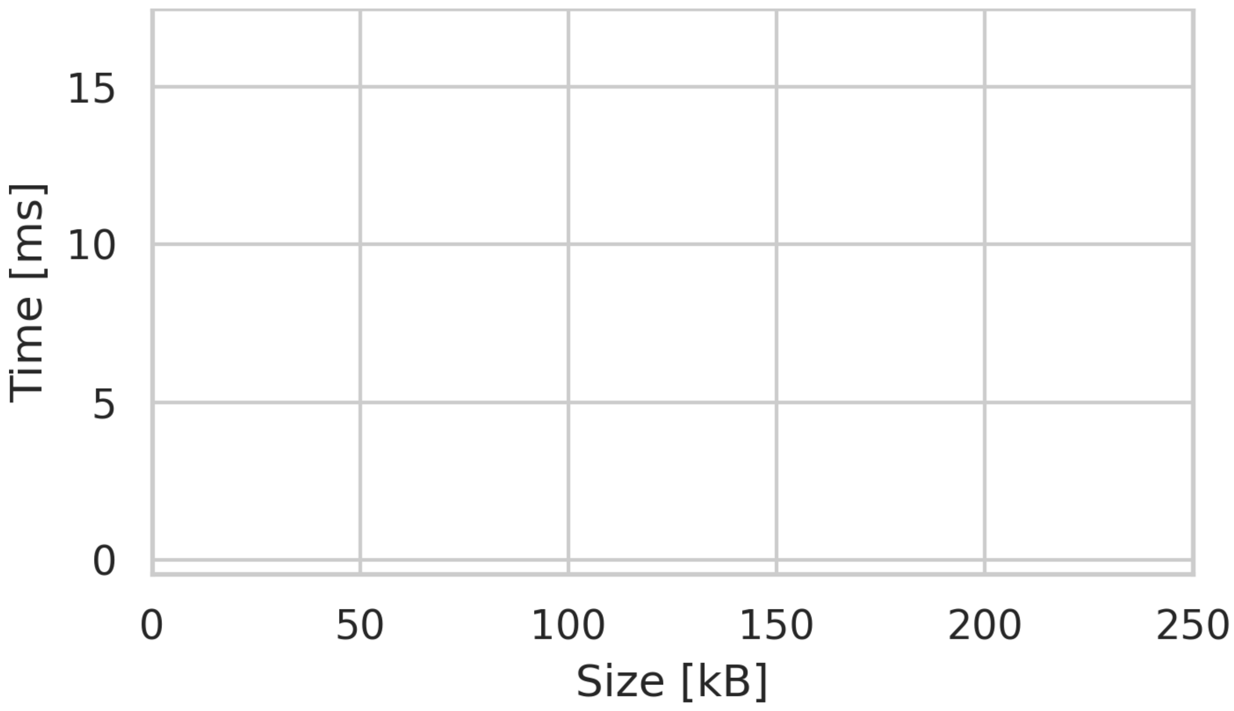
Google Cloud

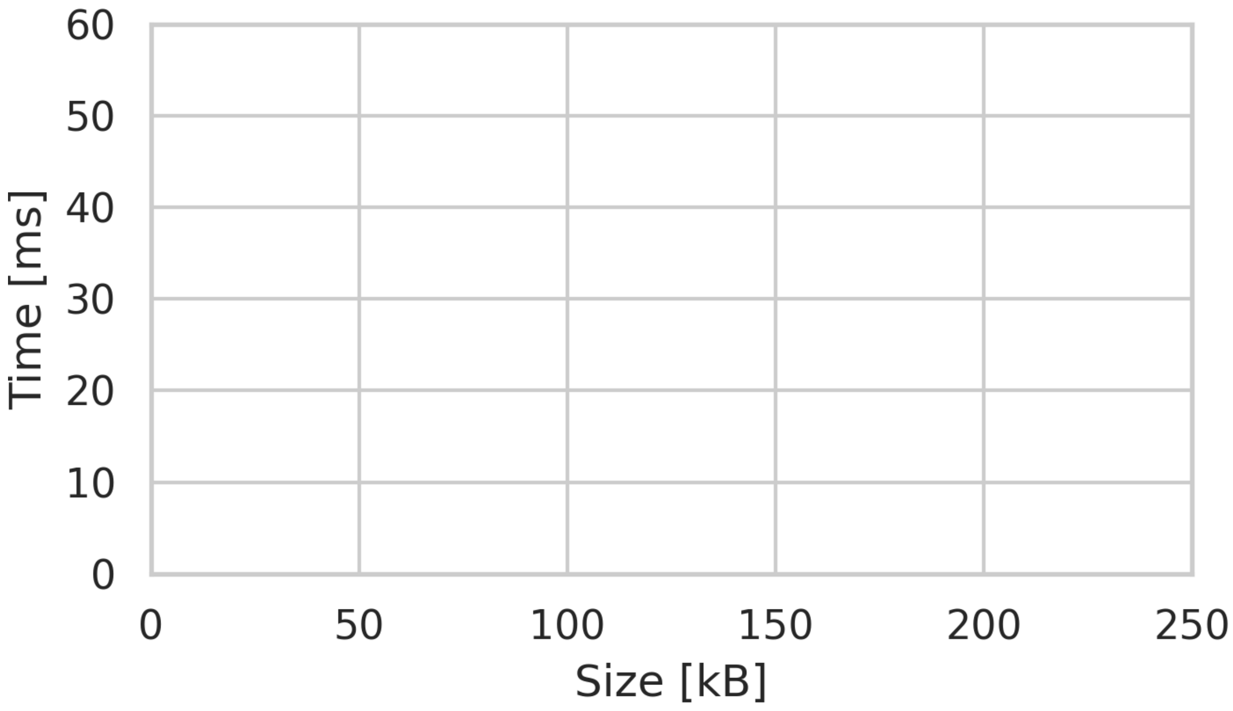
1. **How does read performance compare to ZooKeeper?**
2. **How does write performance compare to ZooKeeper?**
3. **By how much can FaaSKeeper decrease costs?**

Evaluation: Read Performance 1

aws

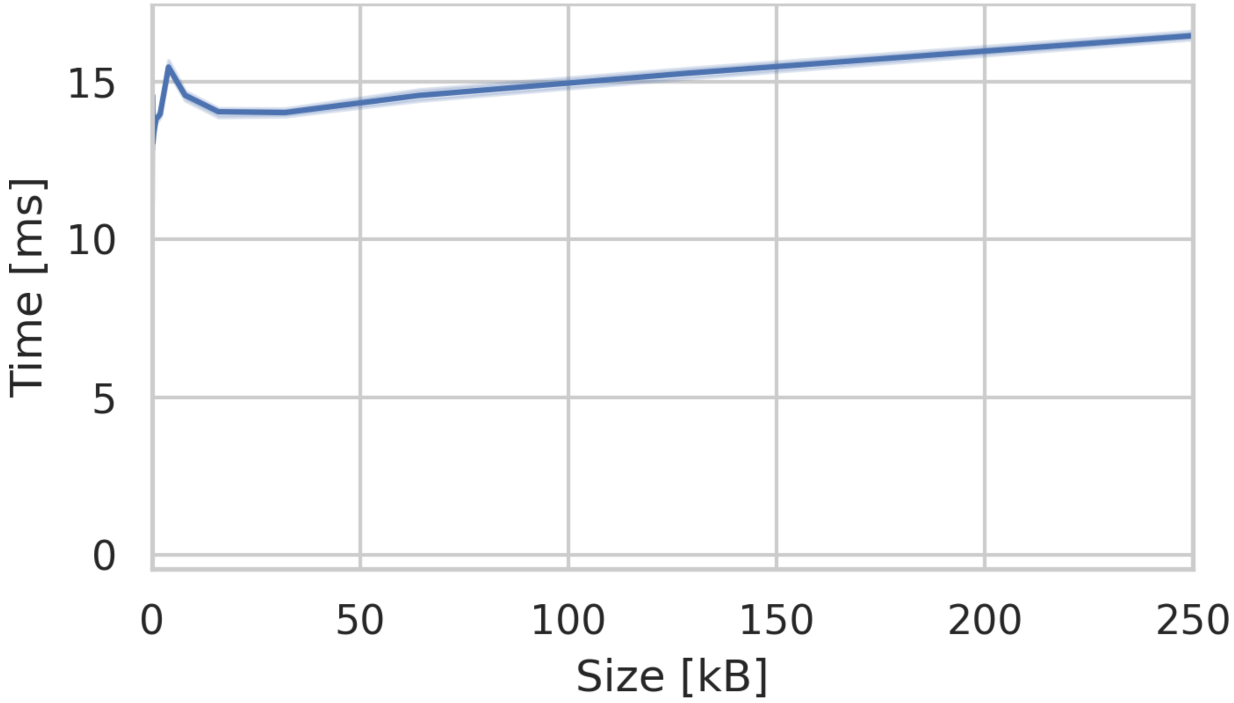
Google Cloud



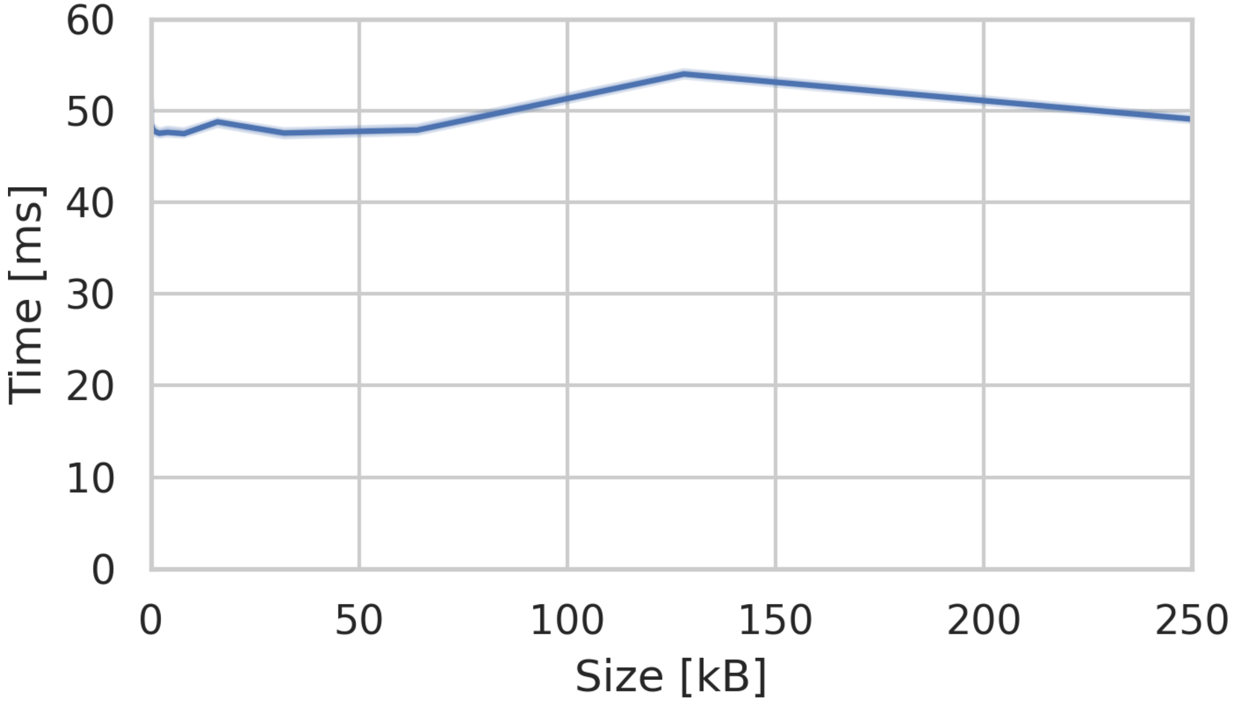


Evaluation: Read Performance 1

aws

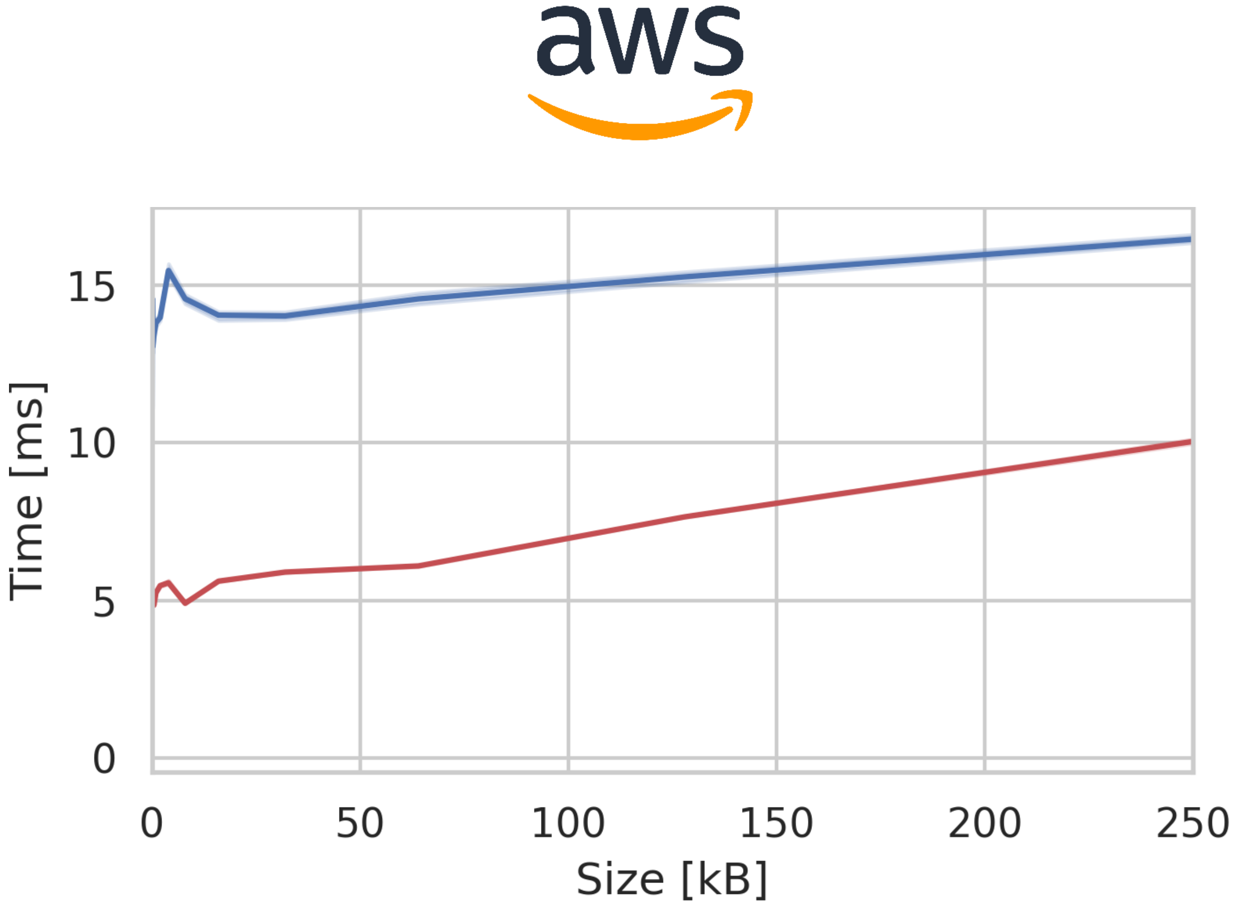


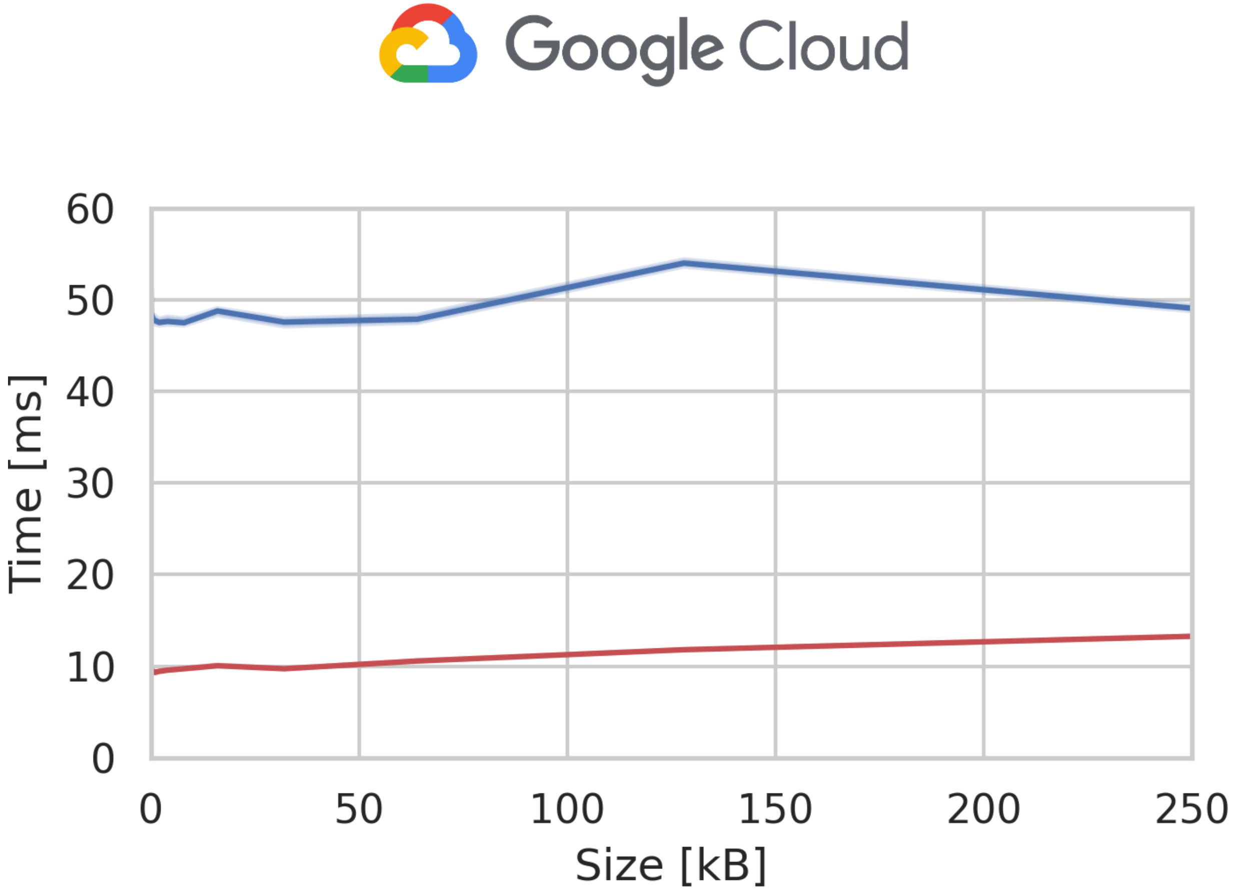
Google Cloud



FaaSKeeper
Object Storage

Evaluation: Read Performance 1



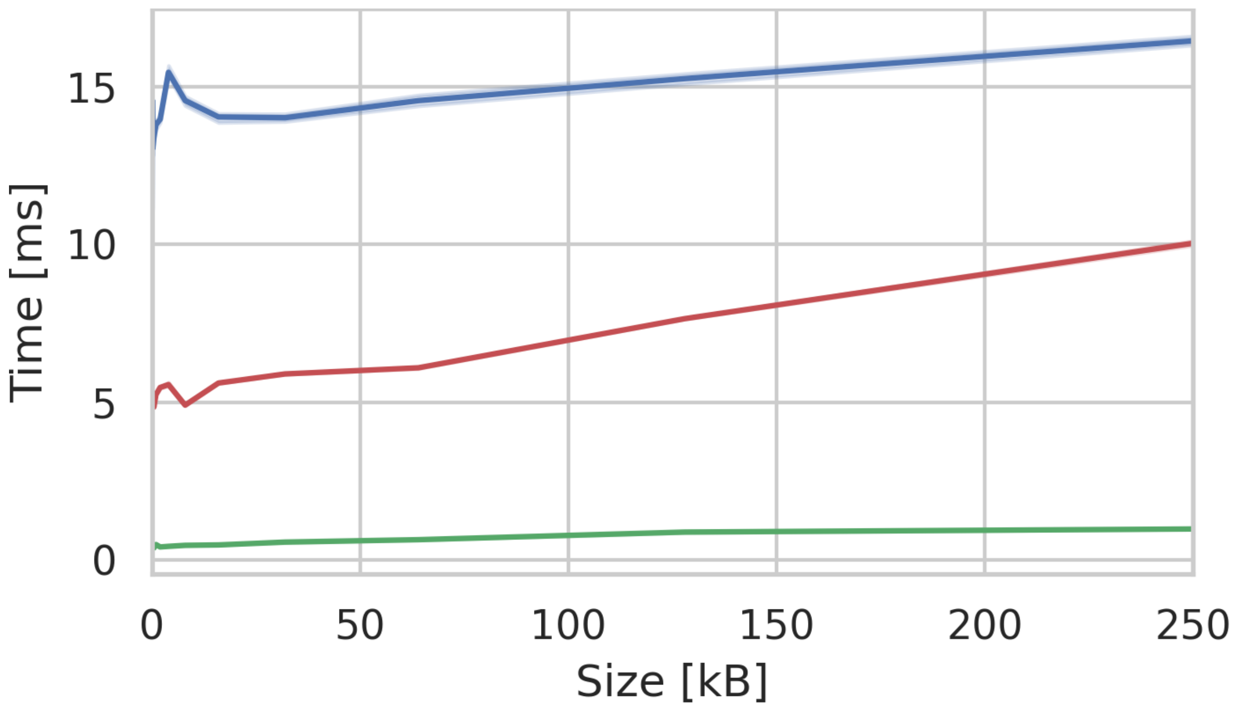


FaaSKeeper Object Storage

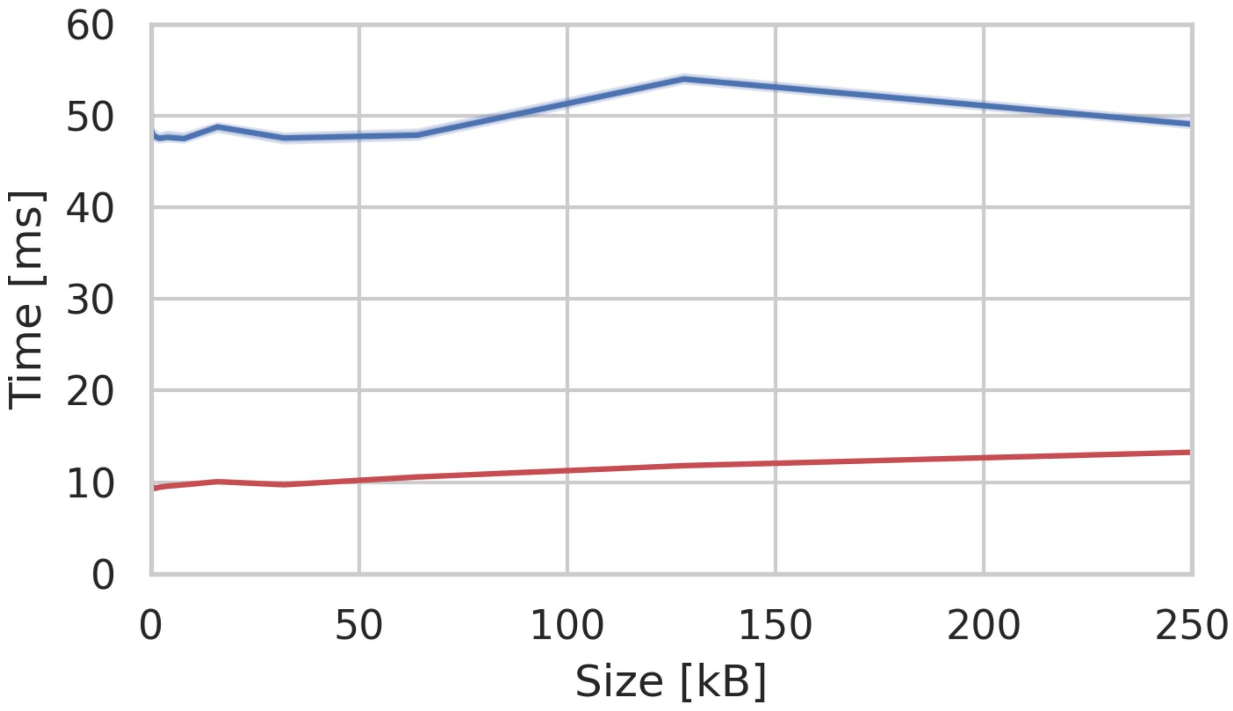
FaaSKeeper Key-Value Storage

Evaluation: Read Performance 1

aws



Google Cloud

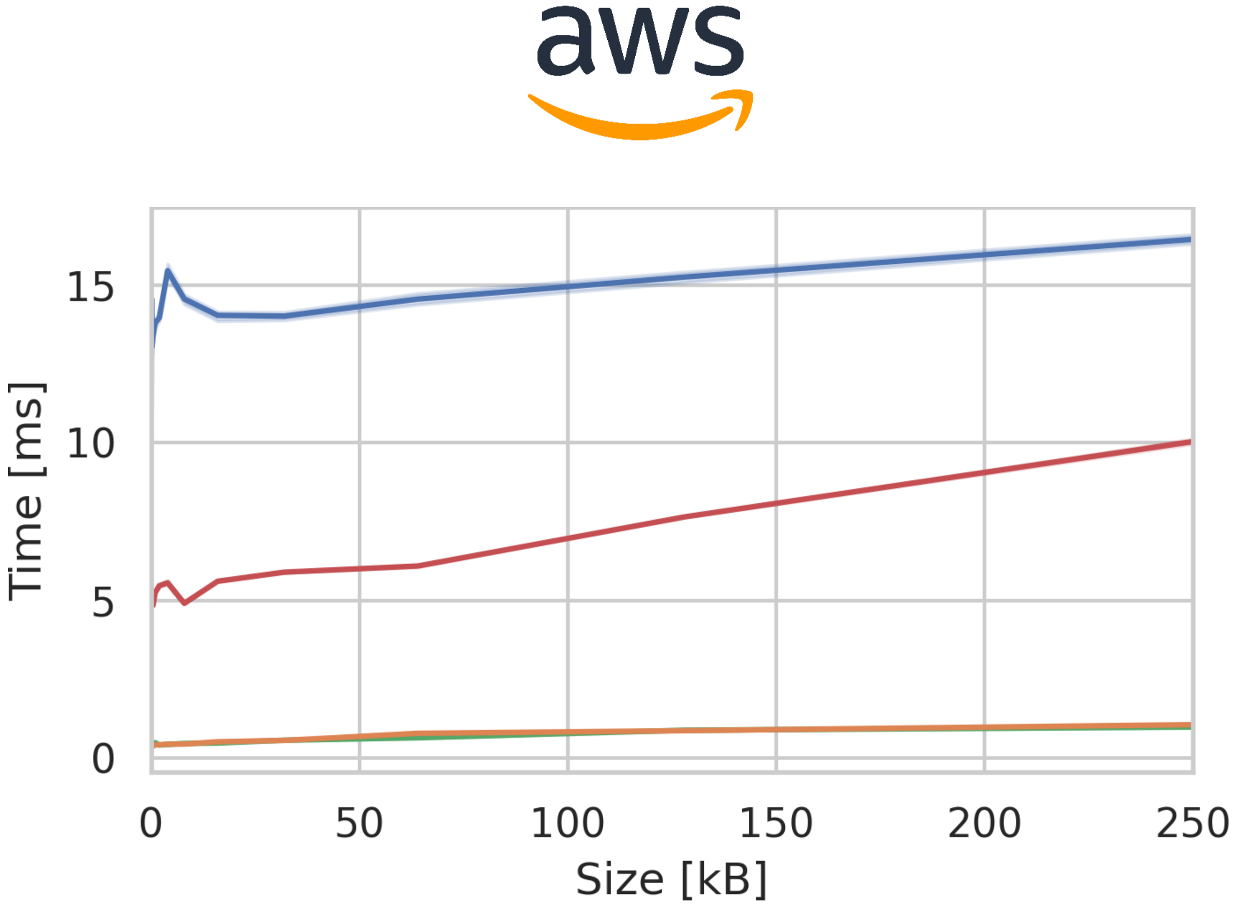


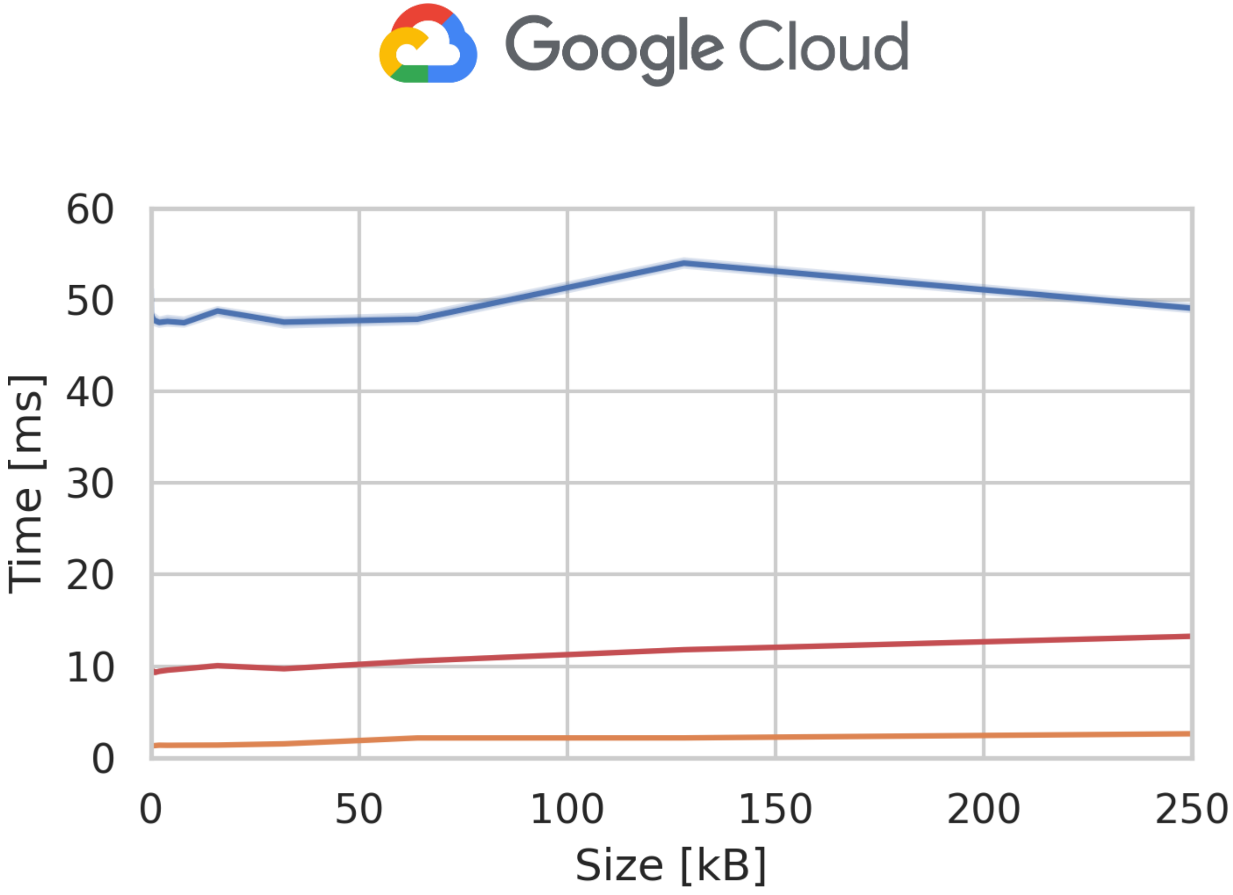
FaaSKeeper Object Storage

FaaSKeeper Key-Value Storage

FaaSKeeper Redis

Evaluation: Read Performance 1





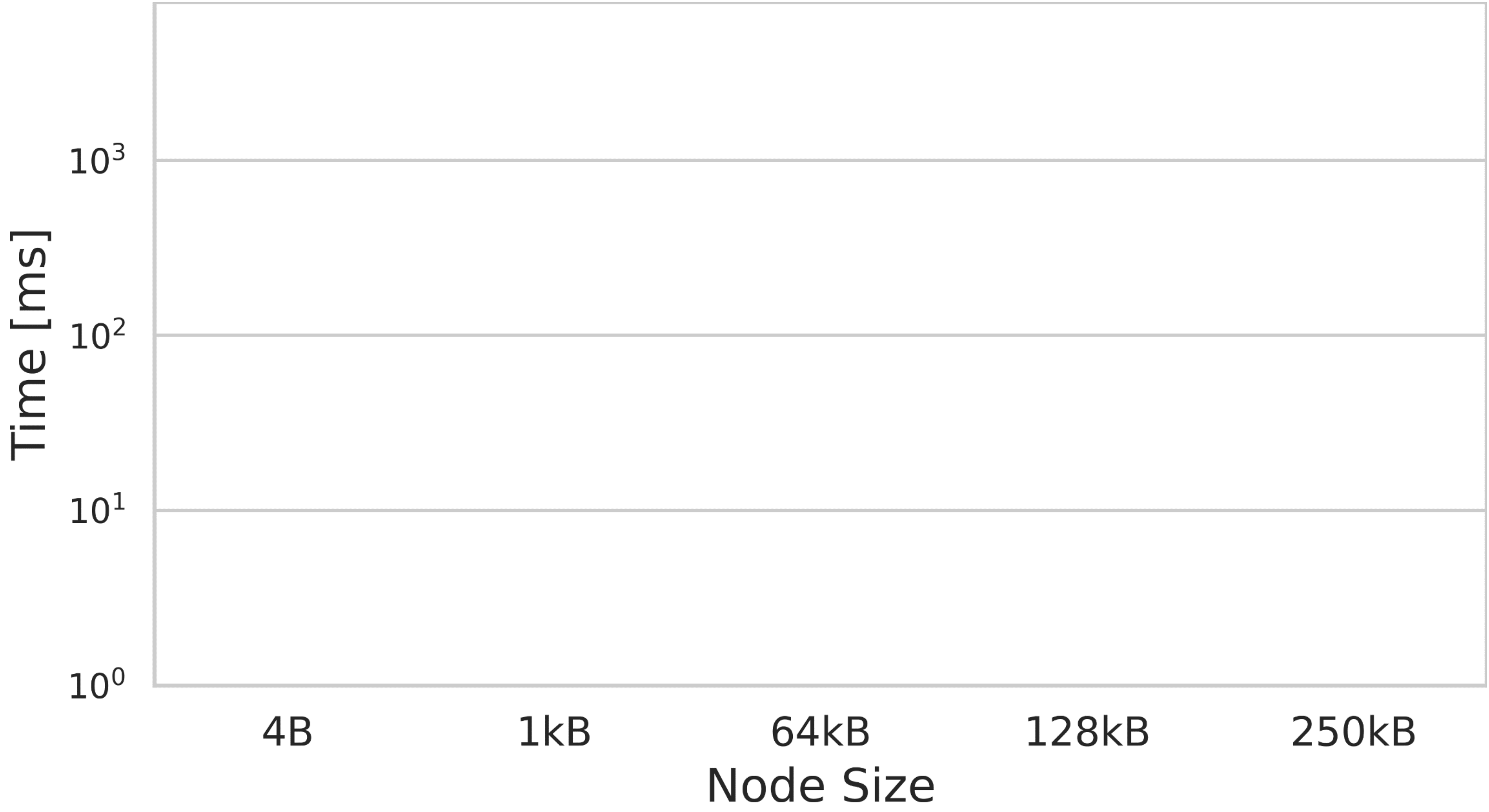
FaaSKeeper Object Storage

FaaSKeeper Key-Value Storage

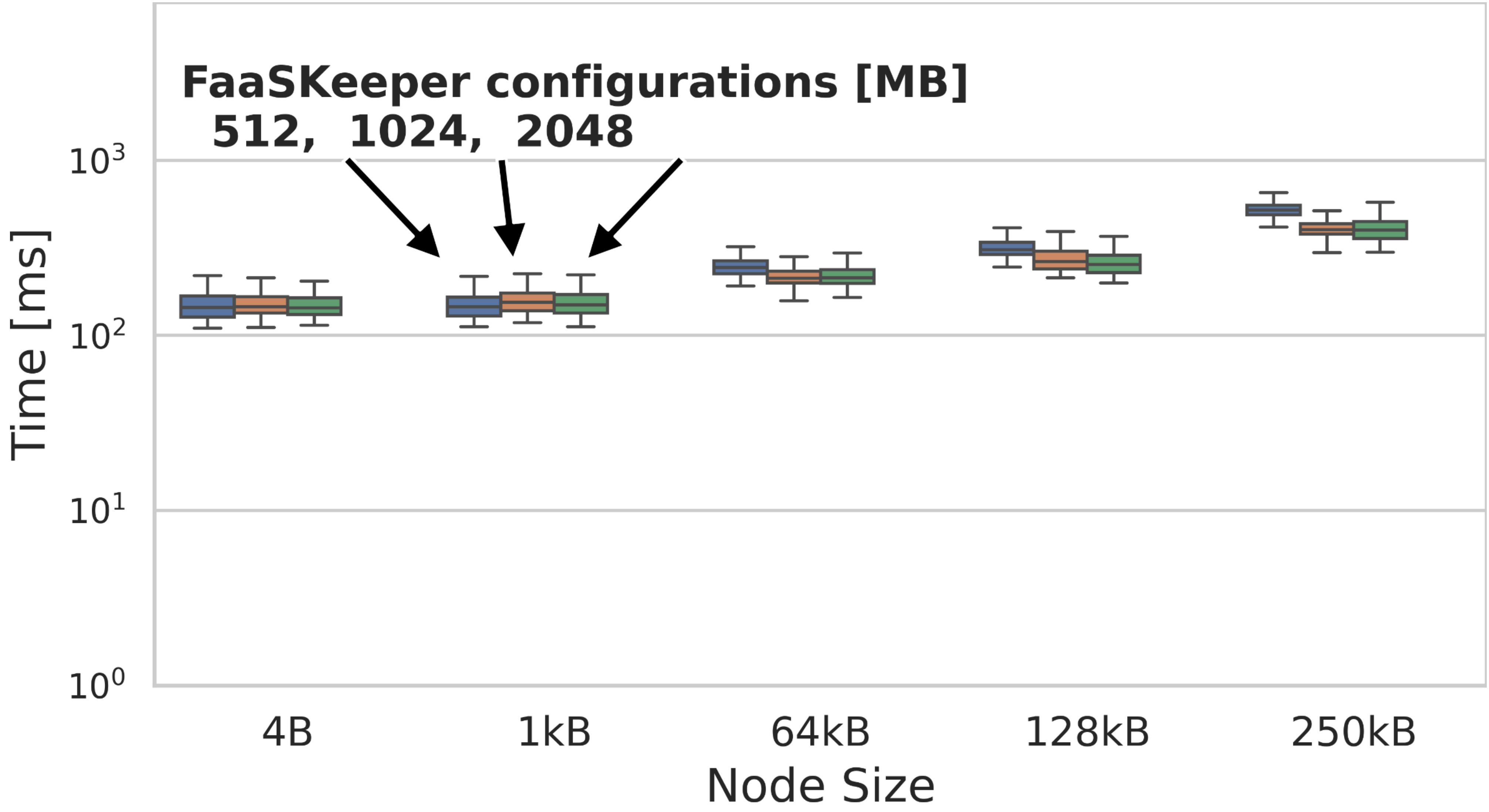
FaaSKeeper Redis

ZooKeeper

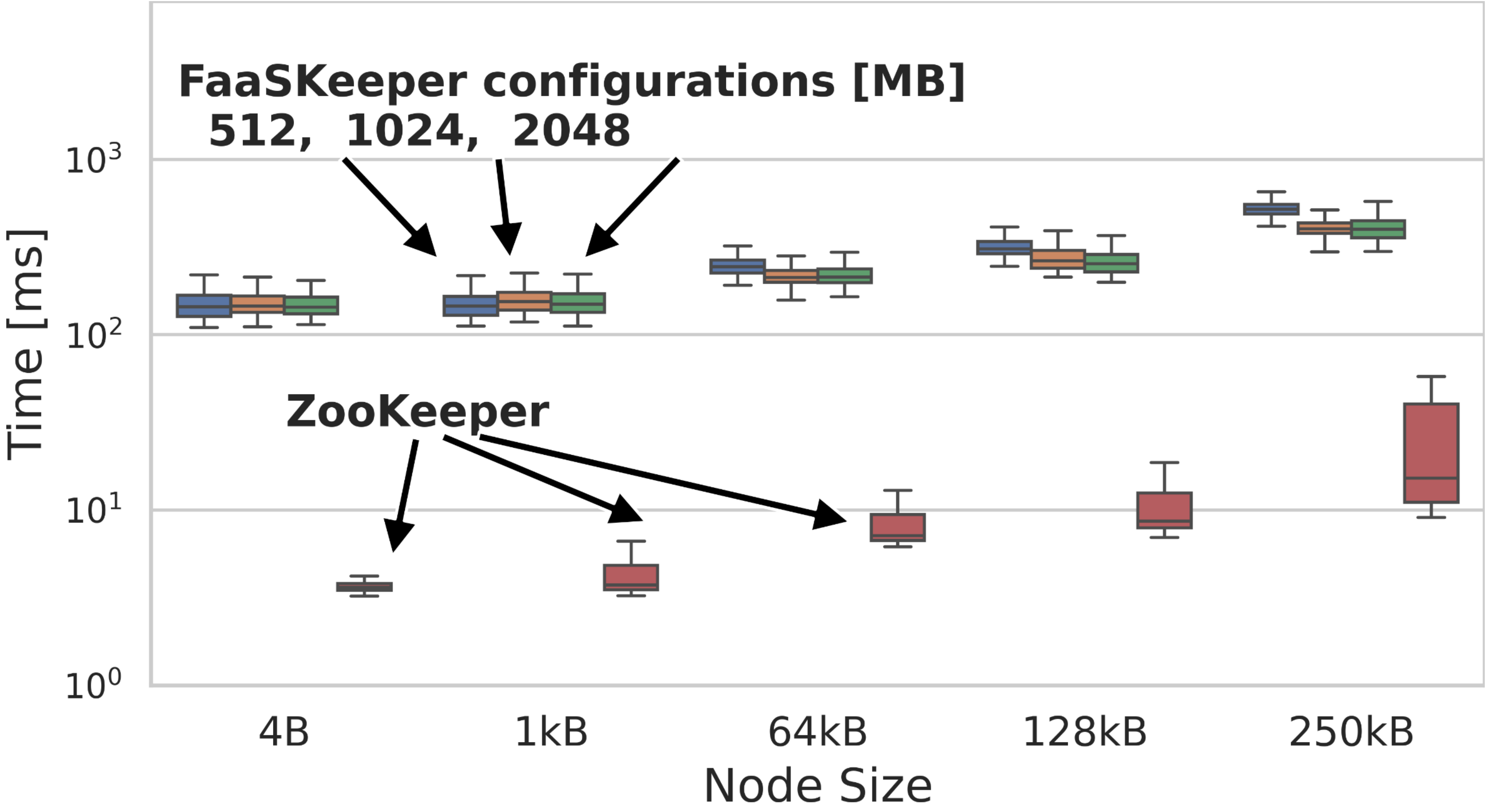
Evaluation: Write Performance 2



Evaluation: Write Performance 2



Evaluation: Write Performance 2



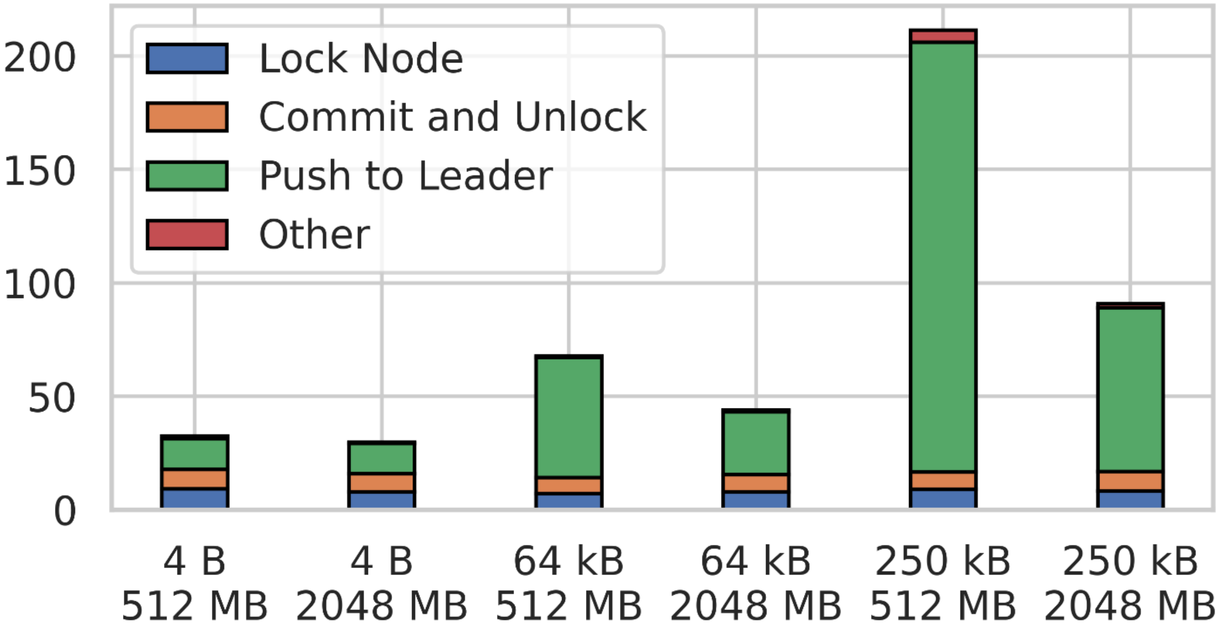
Evaluation: Write Performance 2

Follower Function

Leader Function

Evaluation: Write Performance 2

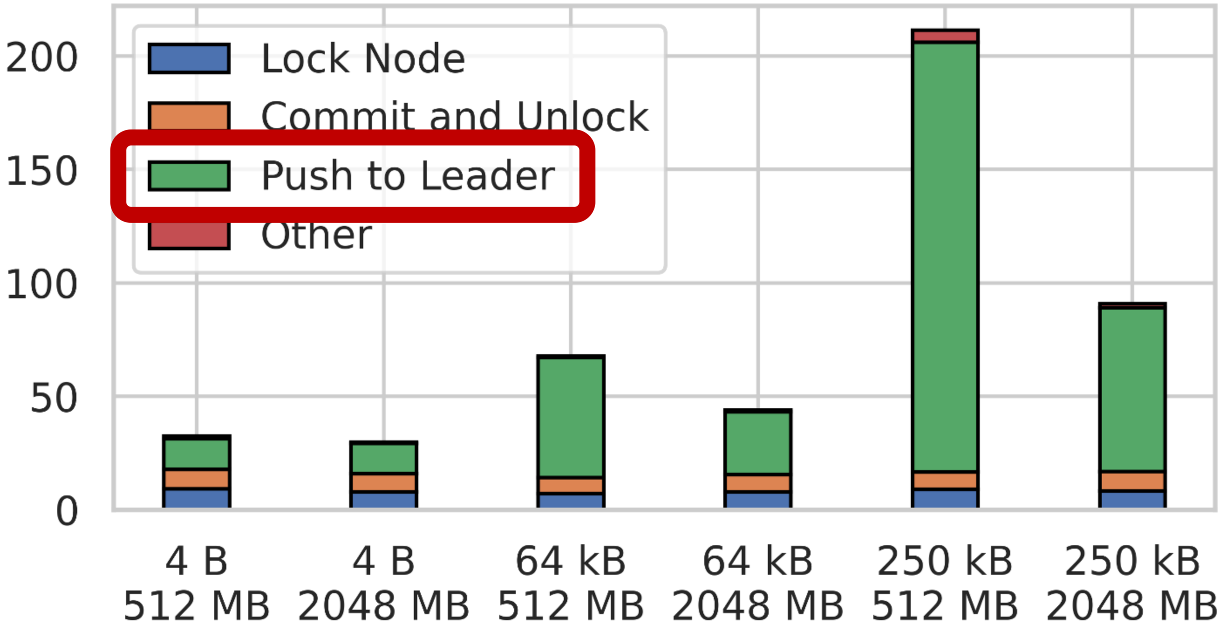
Follower Function



Leader Function

Evaluation: Write Performance 2

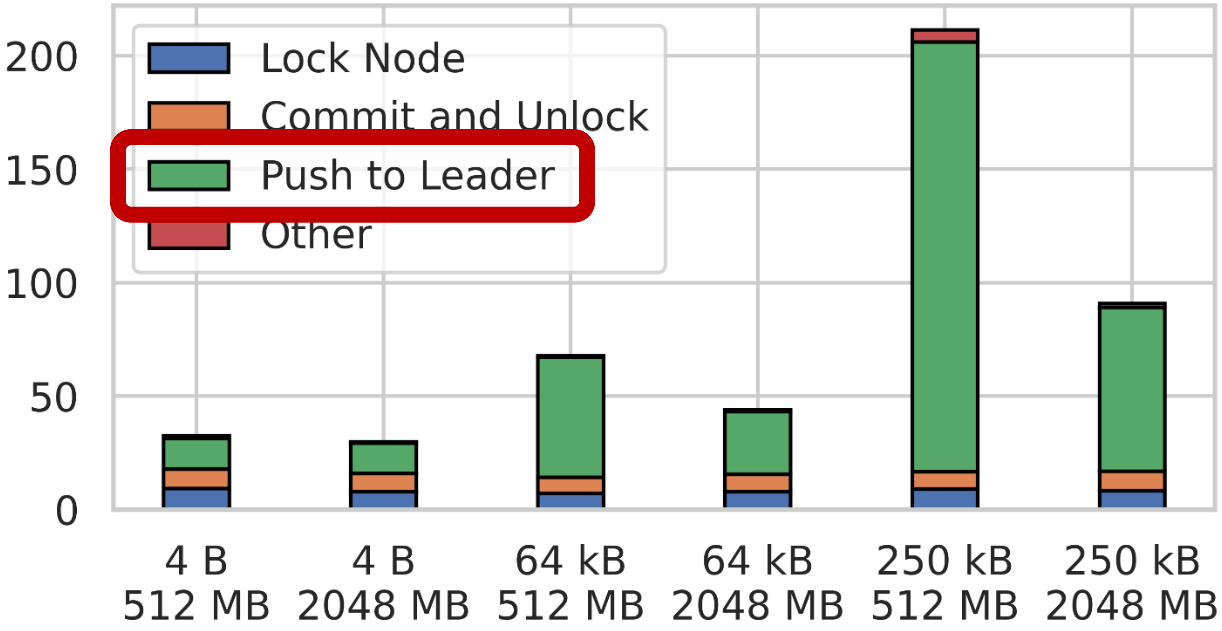
Follower Function



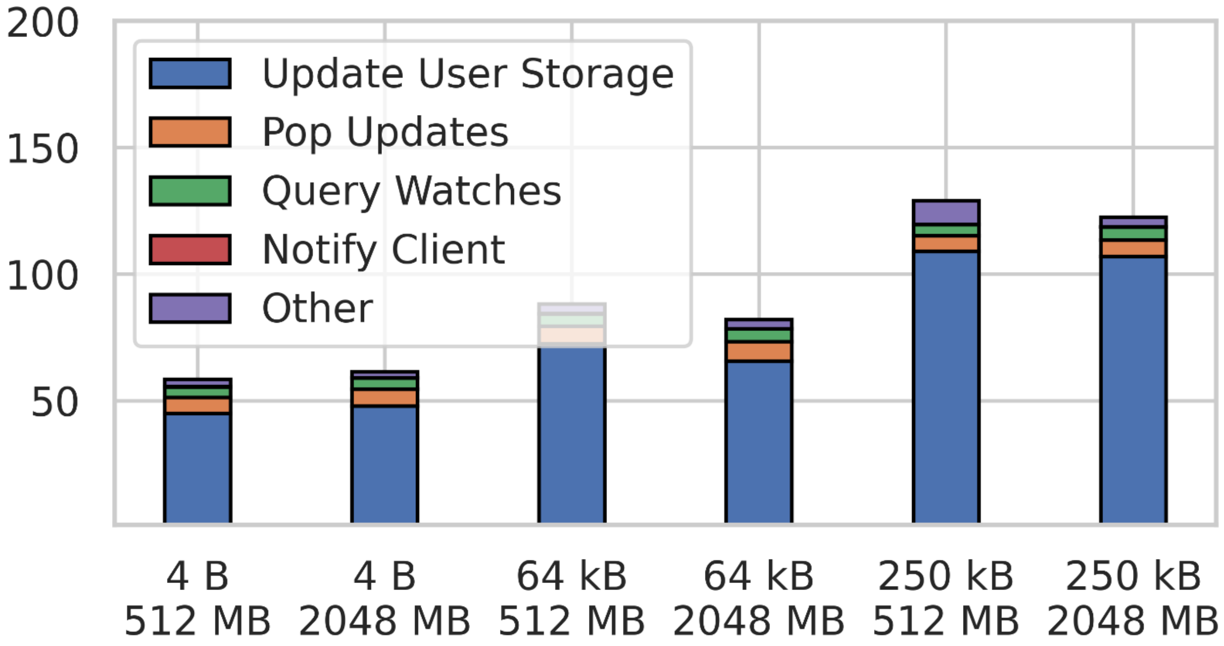
Leader Function

Evaluation: Write Performance 2

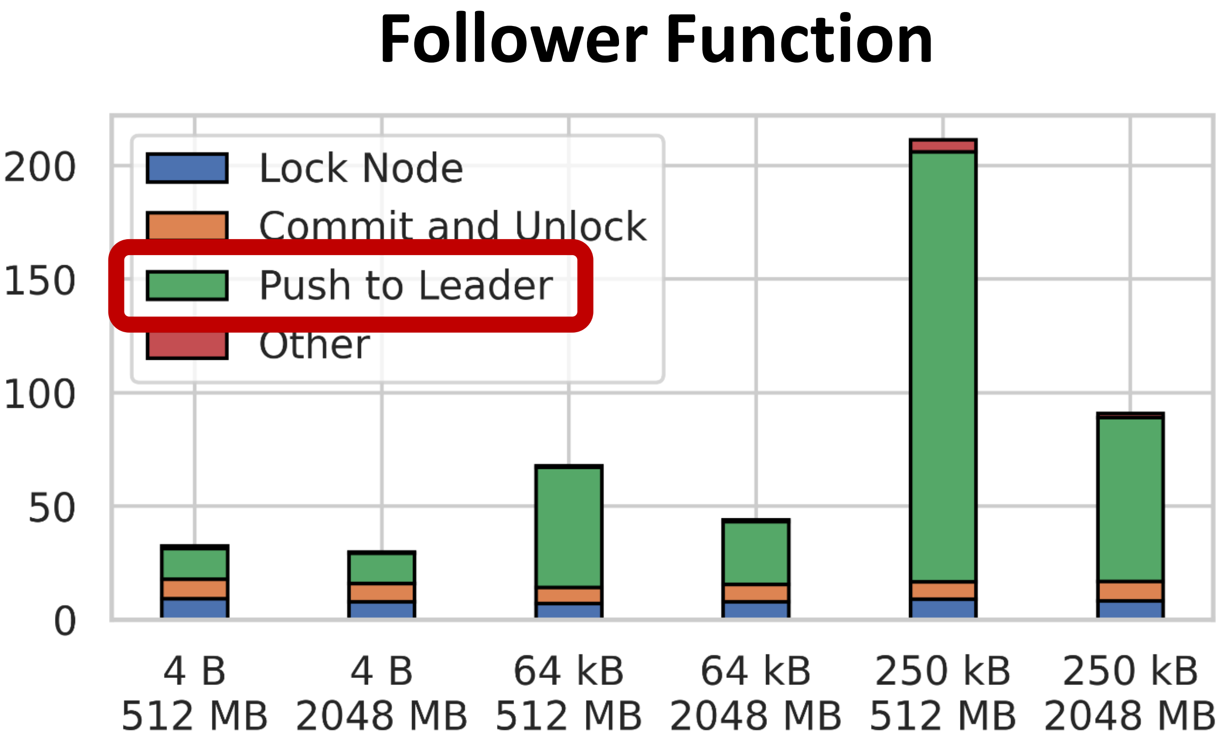
Follower Function

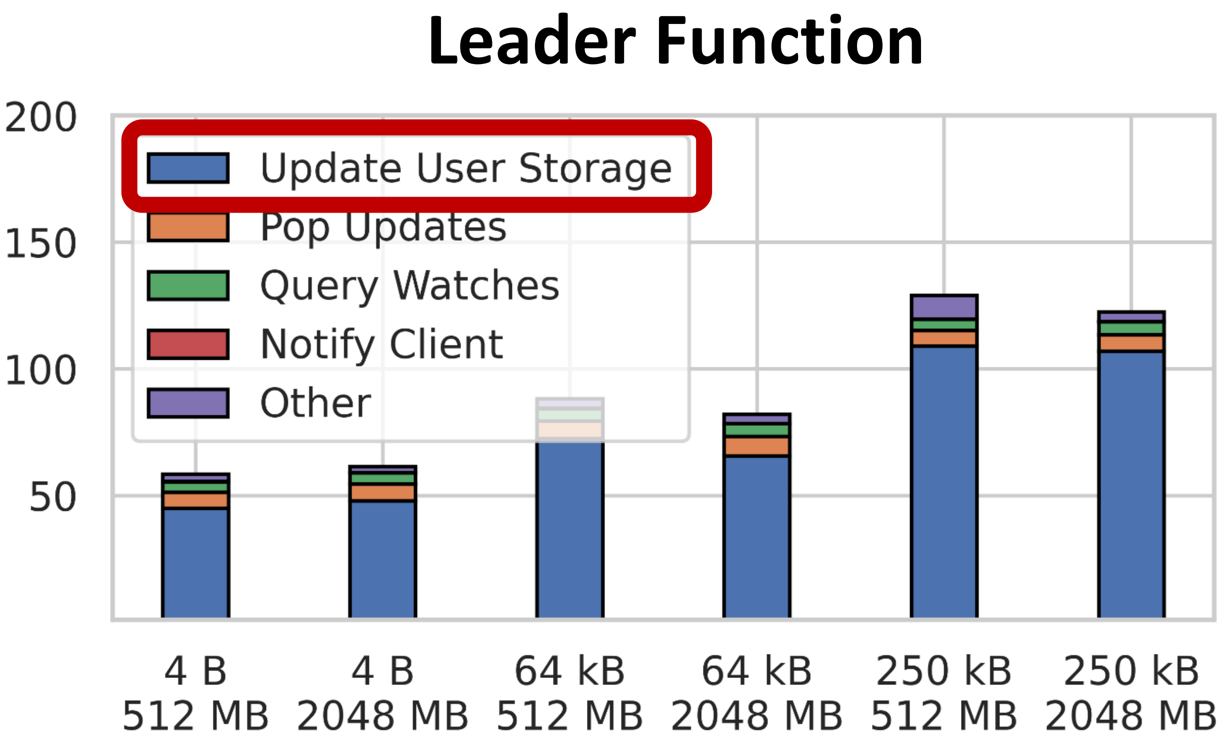


Leader Function



Evaluation: Write Performance 2





Evaluation: Cost Efficiency

3

Cost ratio of ZooKeeper and FaaSKeeper, 90% reads.

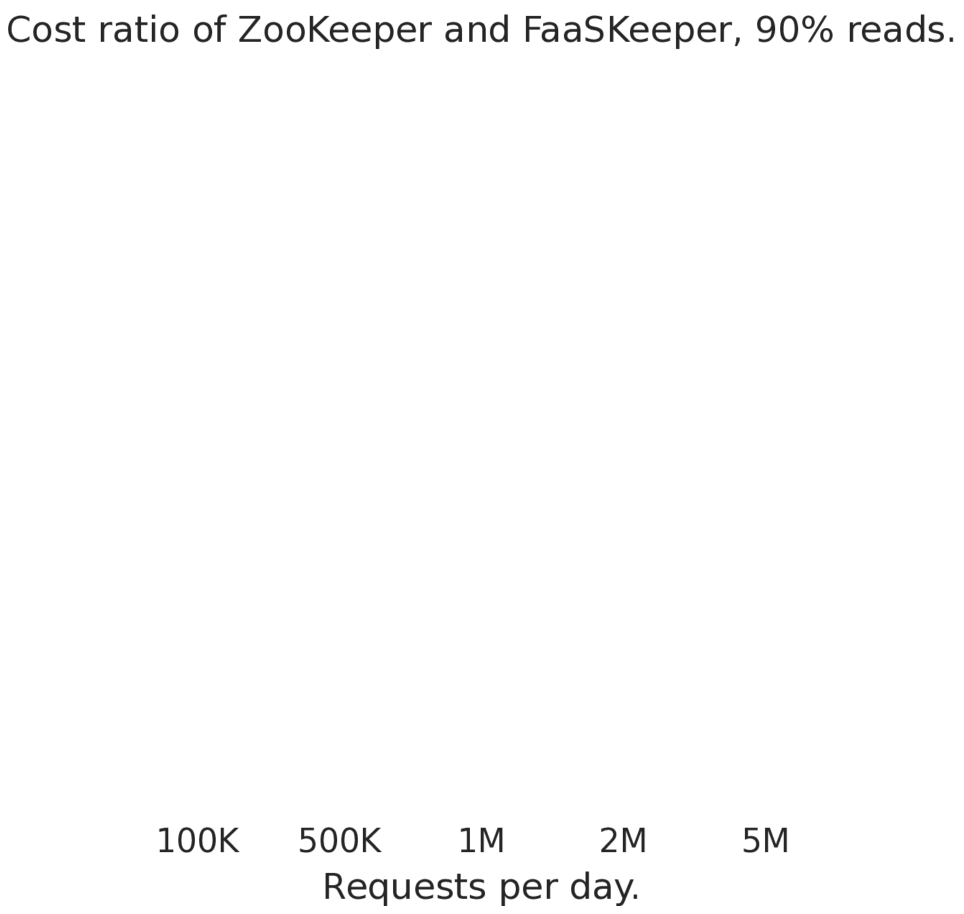
Cost ratio of ZooKeeper and FaaSKeeper, 80% reads.

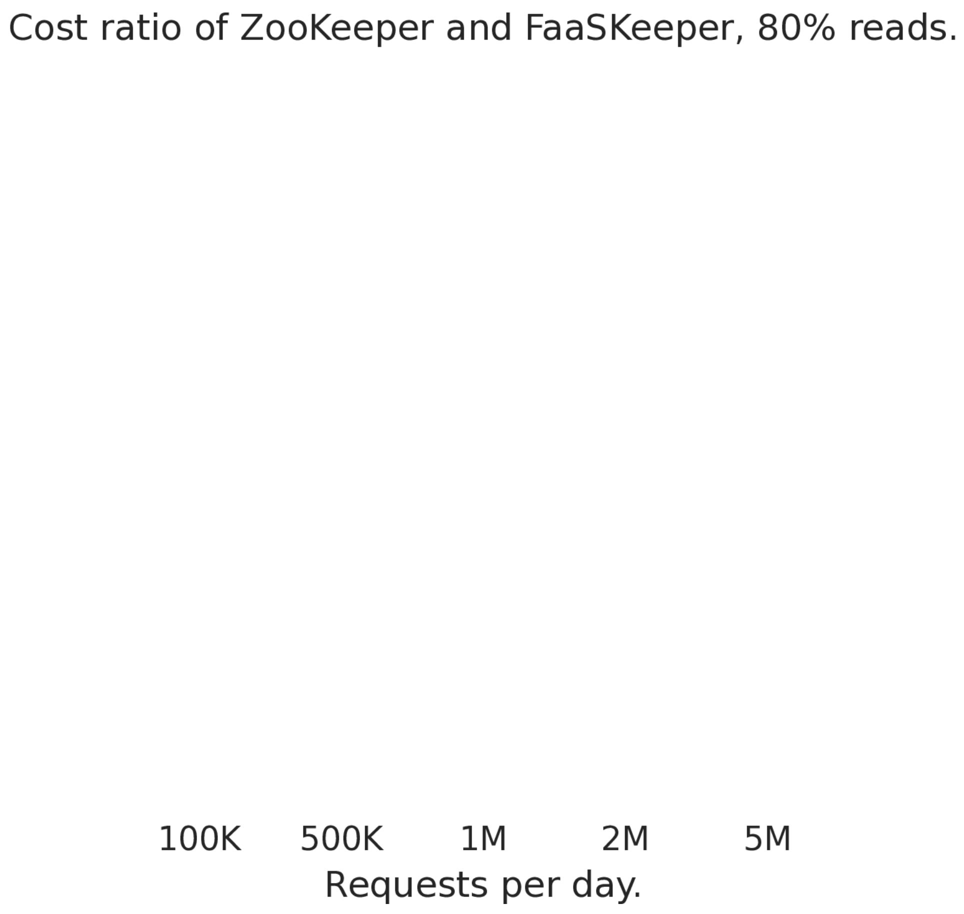
ZooKeeper – constant cost for VMs.
FaaSKeeper – pay per each request.

Set node data of 1 kB, no watches, single request per invocation.

Evaluation: Cost Efficiency

3



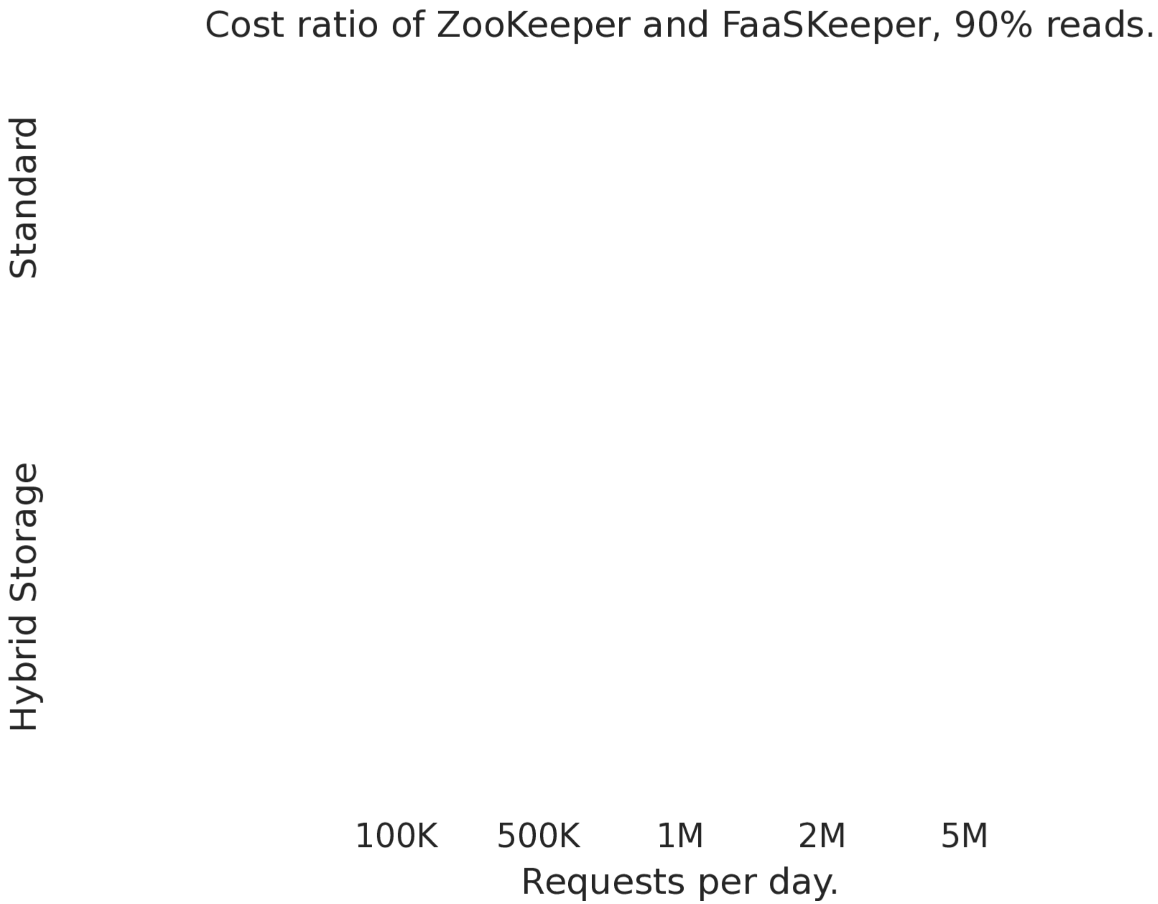


ZooKeeper – constant cost for VMs.
FaaSKeeper – pay per each request.

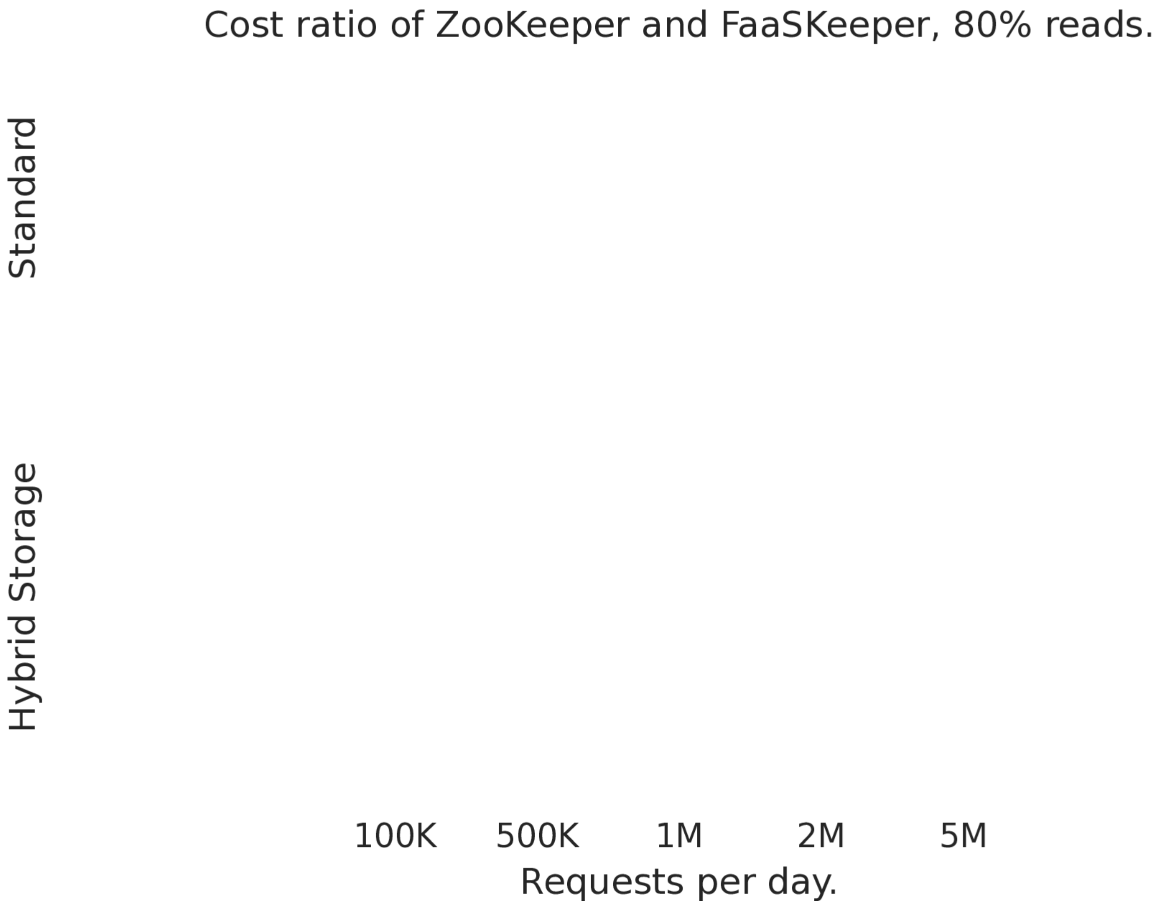
Set node data of 1 kB, no watches, single request per invocation.

Evaluation: Cost Efficiency

3



Cost ratio of ZooKeeper and FaaSKeeper, 90% reads.



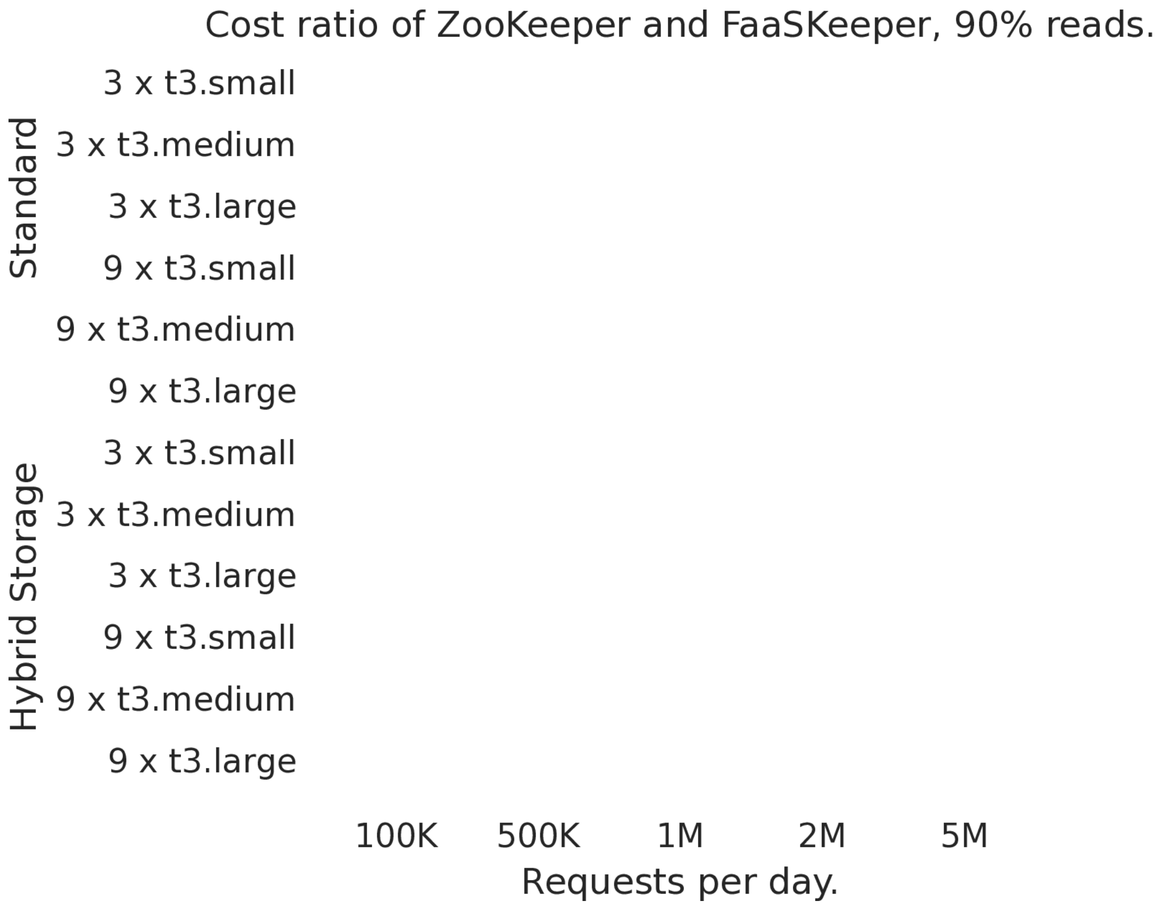
Cost ratio of ZooKeeper and FaaSKeeper, 80% reads.

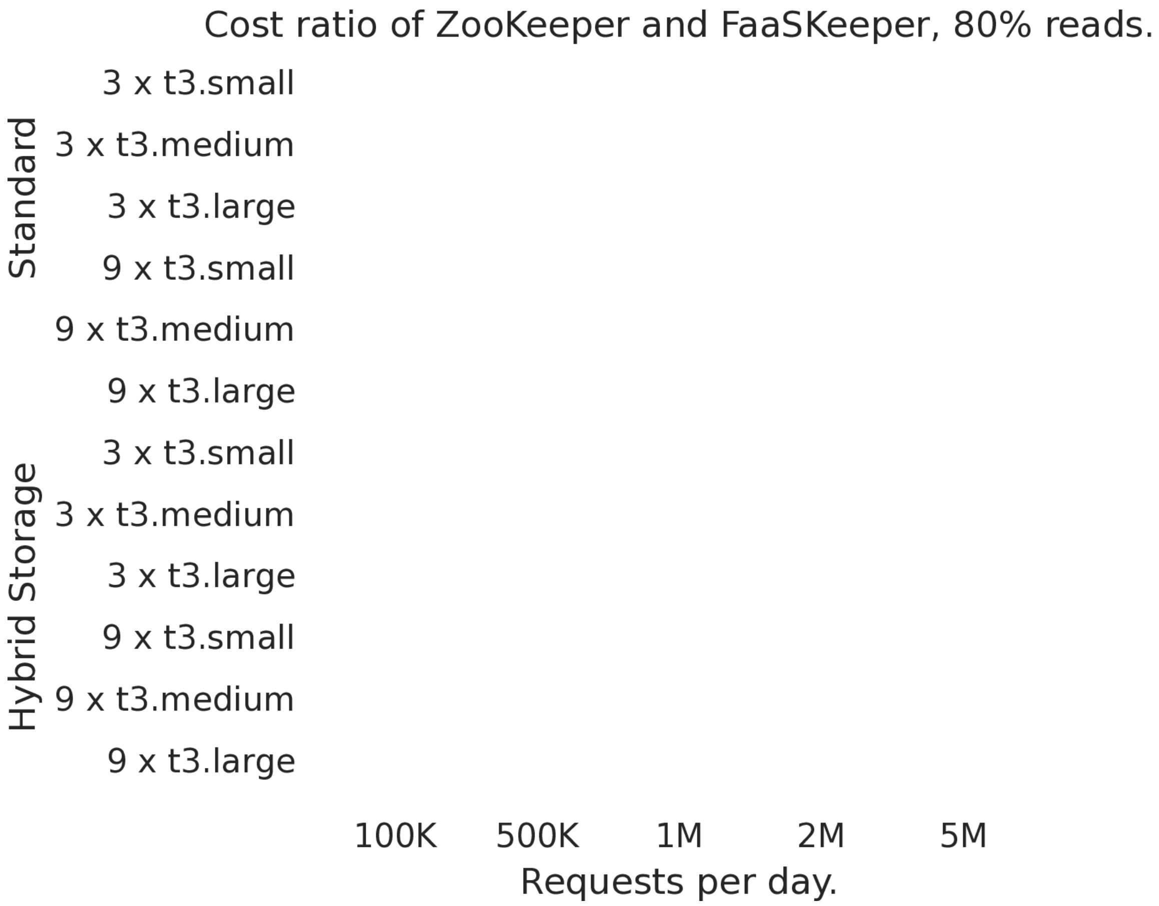
ZooKeeper – constant cost for VMs.
FaaSKeeper – pay per each request.

Set node data of 1 kB, no watches, single request per invocation.

Evaluation: Cost Efficiency

3


Cost ratio of ZooKeeper and FaaSKeeper, 90% reads.


Cost ratio of ZooKeeper and FaaSKeeper, 80% reads.

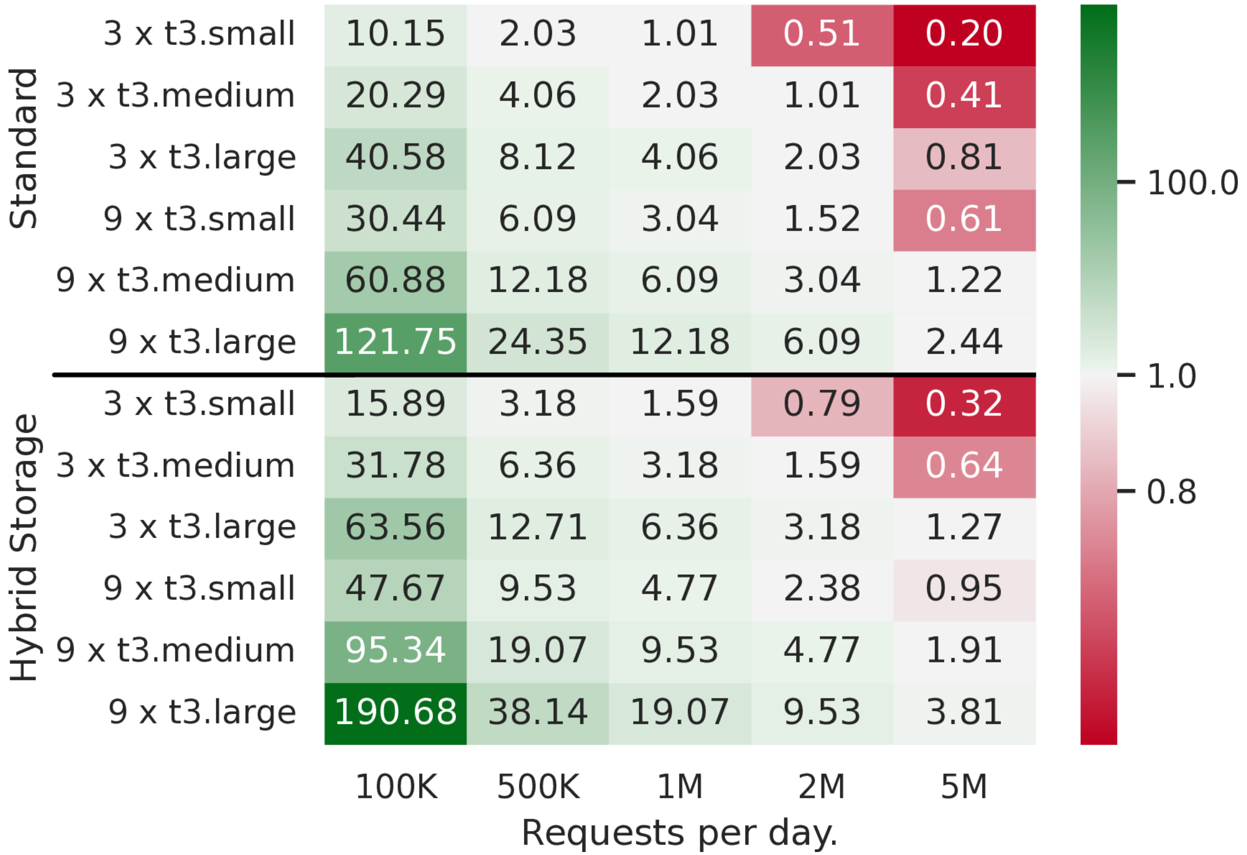
ZooKeeper – constant cost for VMs.
FaaSKeeper – pay per each request.

Set node data of 1 kB, no watches, single request per invocation.

Evaluation: Cost Efficiency

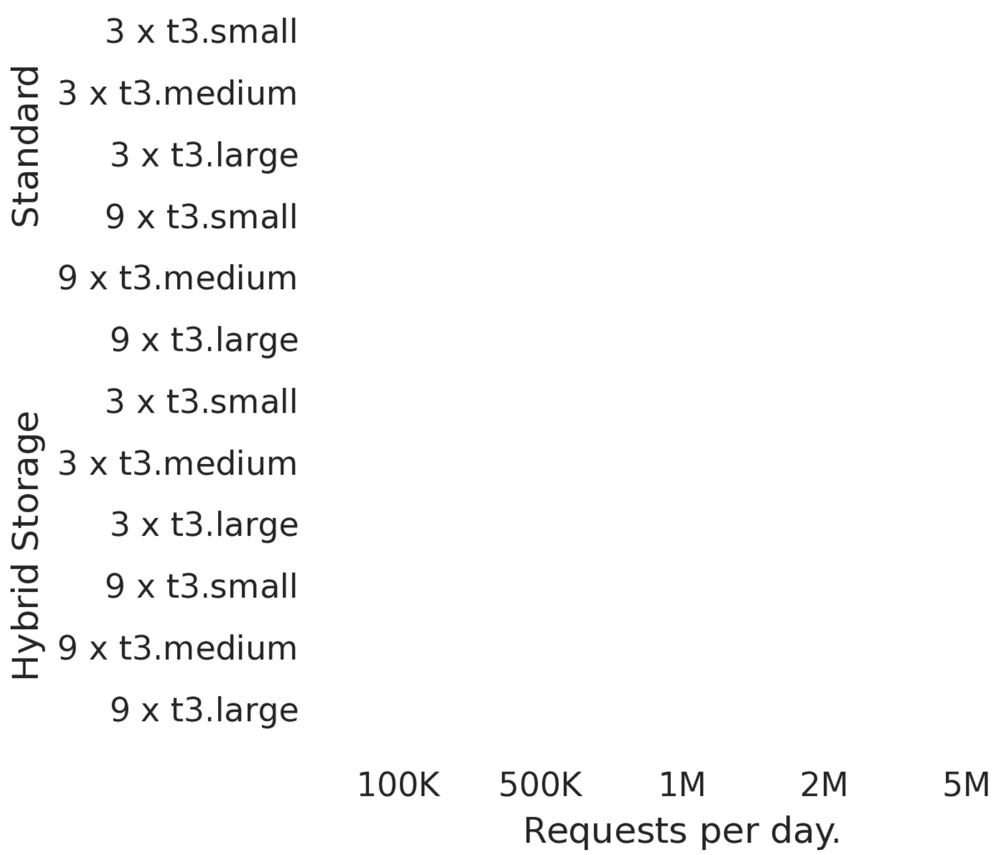
3

Cost ratio of ZooKeeper and FaaSKeeper, 90% reads.



		100K	500K	1M	2M	5M
Standard	3 x t3.small	10.15	2.03	1.01	0.51	0.20
	3 x t3.medium	20.29	4.06	2.03	1.01	0.41
	3 x t3.large	40.58	8.12	4.06	2.03	0.81
	9 x t3.small	30.44	6.09	3.04	1.52	0.61
	9 x t3.medium	60.88	12.18	6.09	3.04	1.22
	9 x t3.large	121.75	24.35	12.18	6.09	2.44
Hybrid Storage	3 x t3.small	15.89	3.18	1.59	0.79	0.32
	3 x t3.medium	31.78	6.36	3.18	1.59	0.64
	3 x t3.large	63.56	12.71	6.36	3.18	1.27
	9 x t3.small	47.67	9.53	4.77	2.38	0.95
	9 x t3.medium	95.34	19.07	9.53	4.77	1.91
	9 x t3.large	190.68	38.14	19.07	9.53	3.81

Cost ratio of ZooKeeper and FaaSKeeper, 80% reads.

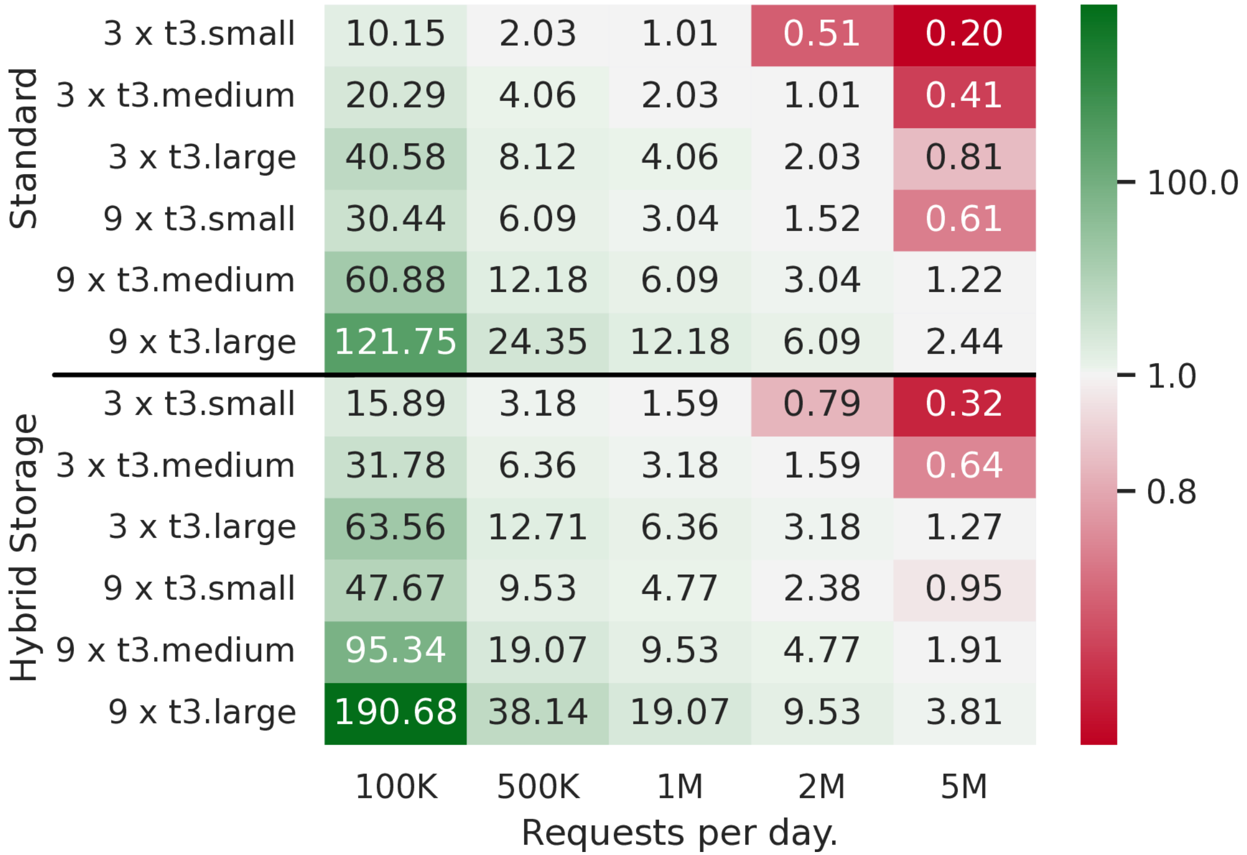


ZooKeeper – constant cost for VMs.
FaaSKeeper – pay per each request.

Set node data of 1 kB, no watches, single request per invocation.

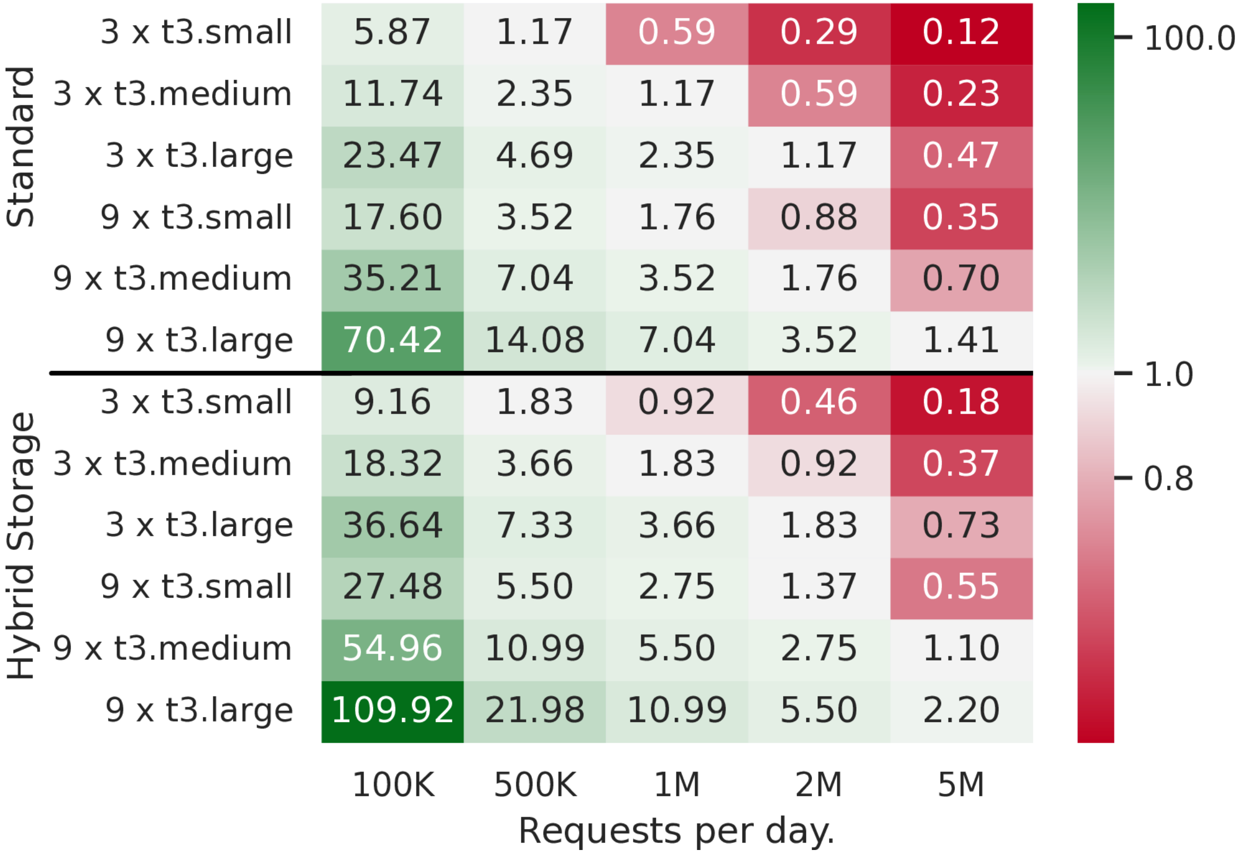
Evaluation: Cost Efficiency

3



Cost ratio of ZooKeeper and FaaSKeeper, 90% reads.

		100K	500K	1M	2M	5M
Standard	3 x t3.small	10.15	2.03	1.01	0.51	0.20
	3 x t3.medium	20.29	4.06	2.03	1.01	0.41
	3 x t3.large	40.58	8.12	4.06	2.03	0.81
	9 x t3.small	30.44	6.09	3.04	1.52	0.61
	9 x t3.medium	60.88	12.18	6.09	3.04	1.22
	9 x t3.large	121.75	24.35	12.18	6.09	2.44
Hybrid Storage	3 x t3.small	15.89	3.18	1.59	0.79	0.32
	3 x t3.medium	31.78	6.36	3.18	1.59	0.64
	3 x t3.large	63.56	12.71	6.36	3.18	1.27
	9 x t3.small	47.67	9.53	4.77	2.38	0.95
	9 x t3.medium	95.34	19.07	9.53	4.77	1.91
	9 x t3.large	190.68	38.14	19.07	9.53	3.81



Cost ratio of ZooKeeper and FaaSKeeper, 80% reads.

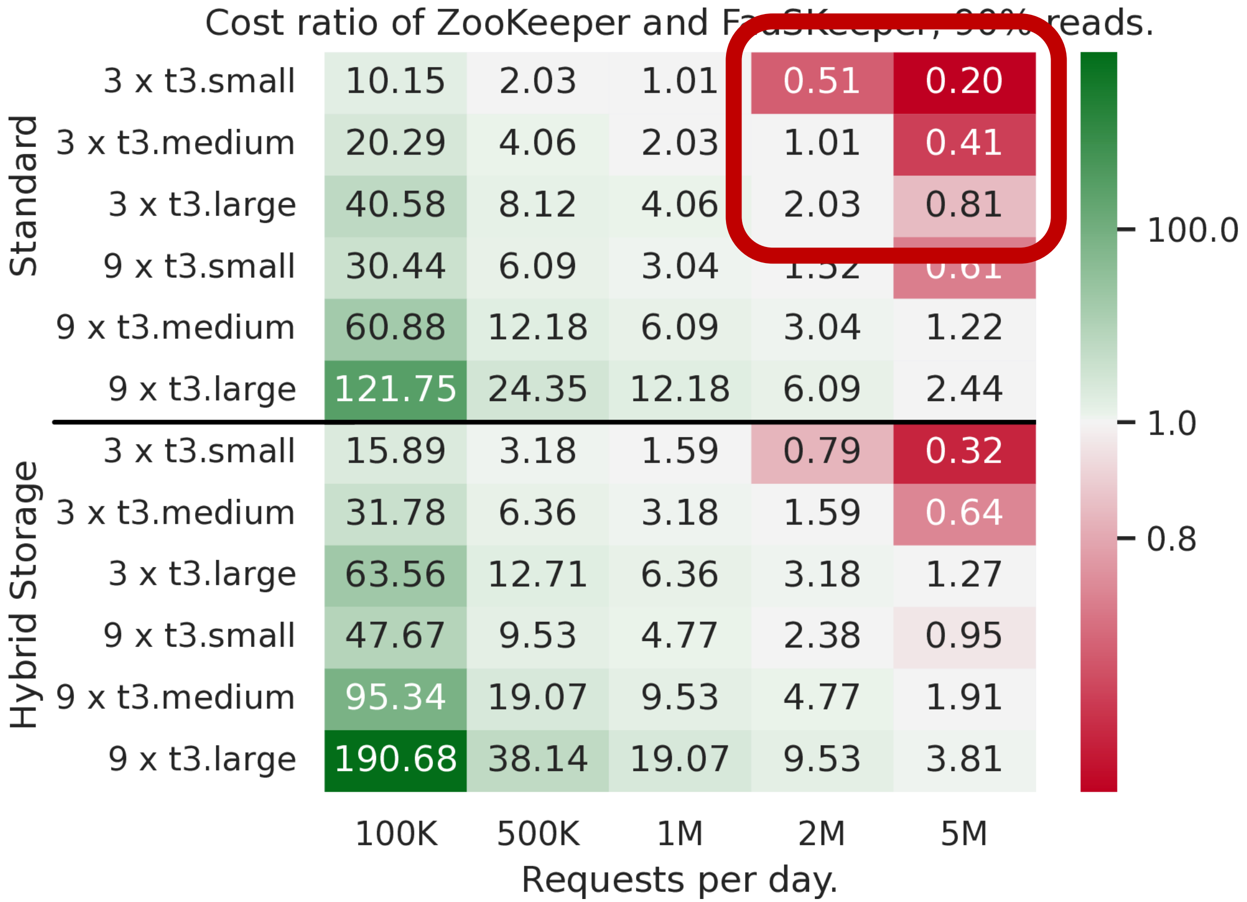
		100K	500K	1M	2M	5M
Standard	3 x t3.small	5.87	1.17	0.59	0.29	0.12
	3 x t3.medium	11.74	2.35	1.17	0.59	0.23
	3 x t3.large	23.47	4.69	2.35	1.17	0.47
	9 x t3.small	17.60	3.52	1.76	0.88	0.35
	9 x t3.medium	35.21	7.04	3.52	1.76	0.70
	9 x t3.large	70.42	14.08	7.04	3.52	1.41
Hybrid Storage	3 x t3.small	9.16	1.83	0.92	0.46	0.18
	3 x t3.medium	18.32	3.66	1.83	0.92	0.37
	3 x t3.large	36.64	7.33	3.66	1.83	0.73
	9 x t3.small	27.48	5.50	2.75	1.37	0.55
	9 x t3.medium	54.96	10.99	5.50	2.75	1.10
	9 x t3.large	109.92	21.98	10.99	5.50	2.20

ZooKeeper – constant cost for VMs.
FaaSKeeper – pay per each request.

Set node data of 1 kB, no watches, single request per invocation.

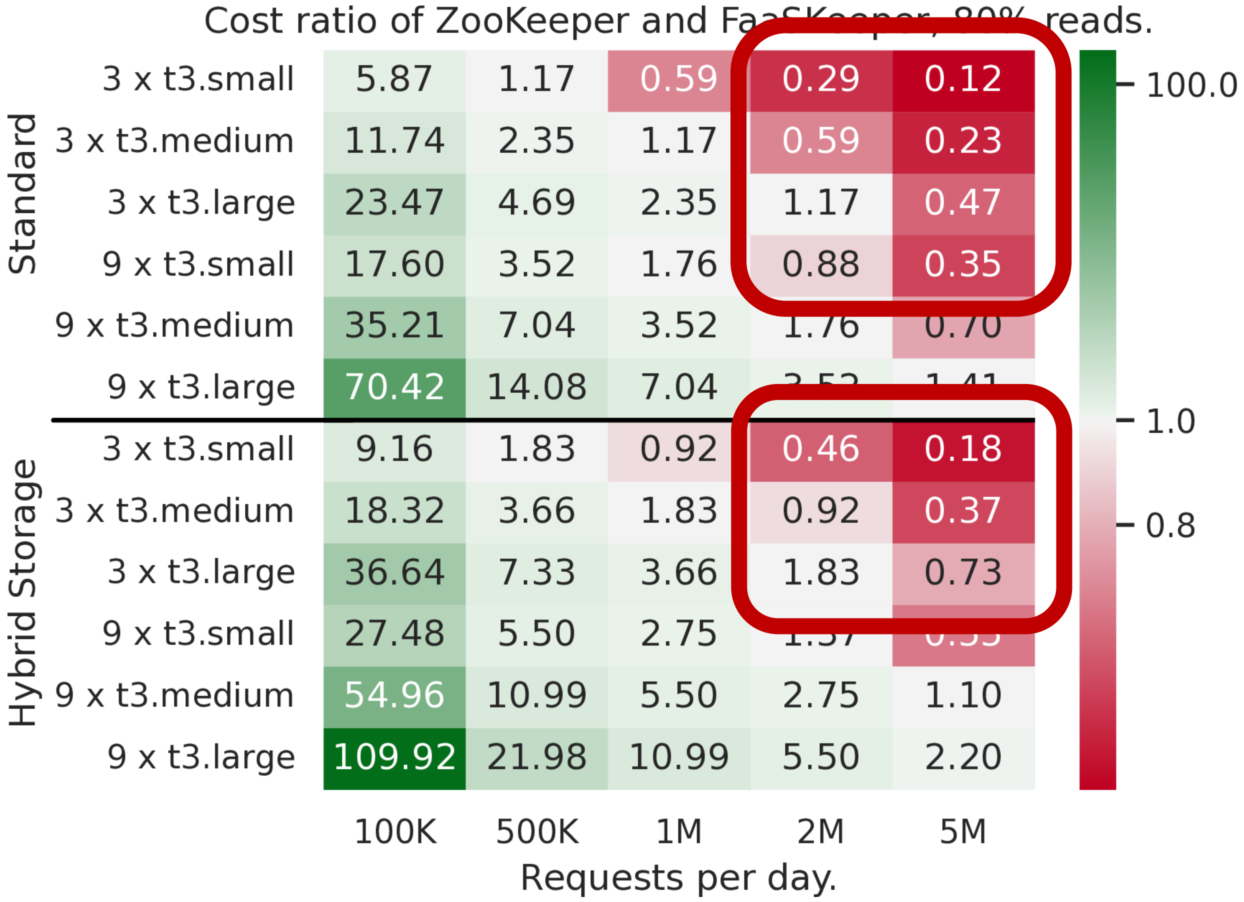
Evaluation: Cost Efficiency

3


Cost ratio of ZooKeeper and FaaSKeeper, 90% reads.

		100K	500K	1M	2M	5M
Standard	3 x t3.small	10.15	2.03	1.01	0.51	0.20
	3 x t3.medium	20.29	4.06	2.03	1.01	0.41
	3 x t3.large	40.58	8.12	4.06	2.03	0.81
	9 x t3.small	30.44	6.09	3.04	1.52	0.61
	9 x t3.medium	60.88	12.18	6.09	3.04	1.22
	9 x t3.large	121.75	24.35	12.18	6.09	2.44
Hybrid Storage	3 x t3.small	15.89	3.18	1.59	0.79	0.32
	3 x t3.medium	31.78	6.36	3.18	1.59	0.64
	3 x t3.large	63.56	12.71	6.36	3.18	1.27
	9 x t3.small	47.67	9.53	4.77	2.38	0.95
	9 x t3.medium	95.34	19.07	9.53	4.77	1.91
	9 x t3.large	190.68	38.14	19.07	9.53	3.81

Requests per day.


Cost ratio of ZooKeeper and FaaSKeeper, 80% reads.

		100K	500K	1M	2M	5M
Standard	3 x t3.small	5.87	1.17	0.59	0.29	0.12
	3 x t3.medium	11.74	2.35	1.17	0.59	0.23
	3 x t3.large	23.47	4.69	2.35	1.17	0.47
	9 x t3.small	17.60	3.52	1.76	0.88	0.35
	9 x t3.medium	35.21	7.04	3.52	1.76	0.70
	9 x t3.large	70.42	14.08	7.04	3.52	1.41
Hybrid Storage	3 x t3.small	9.16	1.83	0.92	0.46	0.18
	3 x t3.medium	18.32	3.66	1.83	0.92	0.37
	3 x t3.large	36.64	7.33	3.66	1.83	0.73
	9 x t3.small	27.48	5.50	2.75	1.37	0.55
	9 x t3.medium	54.96	10.99	5.50	2.75	1.10
	9 x t3.large	109.92	21.98	10.99	5.50	2.20

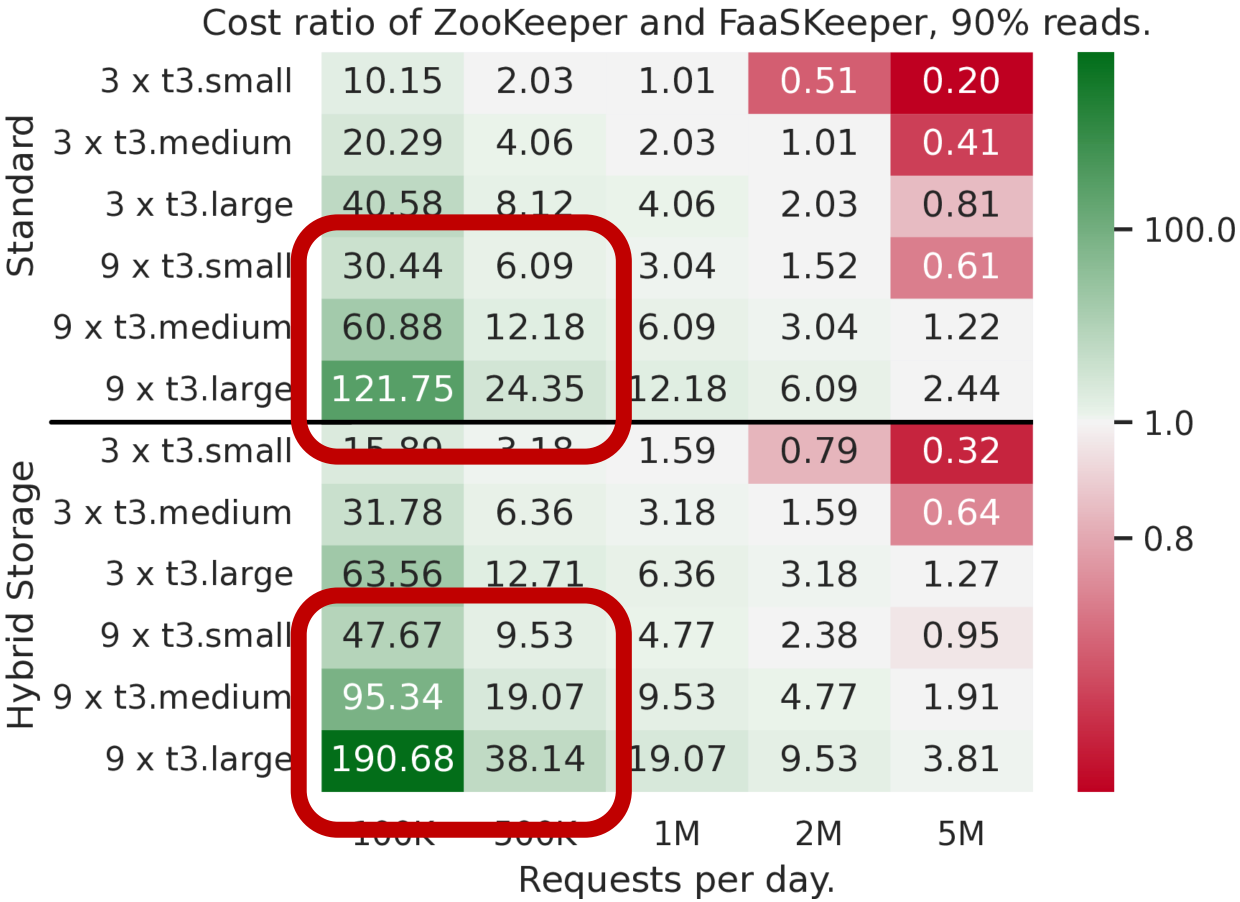
Requests per day.

ZooKeeper – constant cost for VMs.
FaaSKeeper – pay per each request.

Set node data of 1 kB, no watches, single request per invocation.

Evaluation: Cost Efficiency

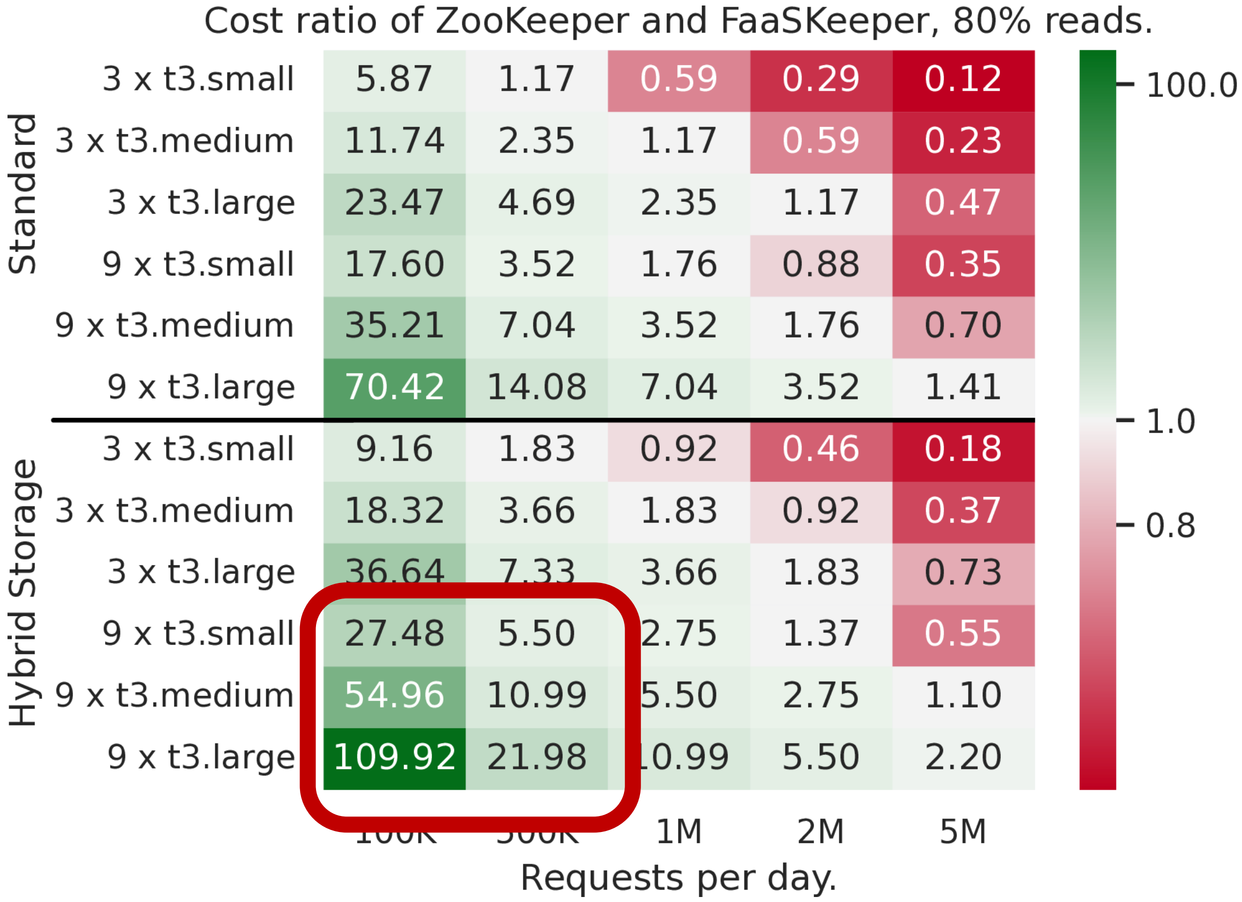
3



Cost ratio of ZooKeeper and FaaSKeeper, 90% reads.

		100K	500K	1M	2M	5M
Standard	3 x t3.small	10.15	2.03	1.01	0.51	0.20
	3 x t3.medium	20.29	4.06	2.03	1.01	0.41
	3 x t3.large	40.58	8.12	4.06	2.03	0.81
	9 x t3.small	30.44	6.09	3.04	1.52	0.61
	9 x t3.medium	60.88	12.18	6.09	3.04	1.22
	9 x t3.large	121.75	24.35	12.18	6.09	2.44
Hybrid Storage	3 x t3.small	15.89	3.18	1.59	0.79	0.32
	3 x t3.medium	31.78	6.36	3.18	1.59	0.64
	3 x t3.large	63.56	12.71	6.36	3.18	1.27
	9 x t3.small	47.67	9.53	4.77	2.38	0.95
	9 x t3.medium	95.34	19.07	9.53	4.77	1.91
	9 x t3.large	190.68	38.14	19.07	9.53	3.81

Requests per day.



Cost ratio of ZooKeeper and FaaSKeeper, 80% reads.

		100K	500K	1M	2M	5M
Standard	3 x t3.small	5.87	1.17	0.59	0.29	0.12
	3 x t3.medium	11.74	2.35	1.17	0.59	0.23
	3 x t3.large	23.47	4.69	2.35	1.17	0.47
	9 x t3.small	17.60	3.52	1.76	0.88	0.35
	9 x t3.medium	35.21	7.04	3.52	1.76	0.70
	9 x t3.large	70.42	14.08	7.04	3.52	1.41
Hybrid Storage	3 x t3.small	9.16	1.83	0.92	0.46	0.18
	3 x t3.medium	18.32	3.66	1.83	0.92	0.37
	3 x t3.large	36.64	7.33	3.66	1.83	0.73
	9 x t3.small	27.48	5.50	2.75	1.37	0.55
	9 x t3.medium	54.96	10.99	5.50	2.75	1.10
	9 x t3.large	109.92	21.98	10.99	5.50	2.20

Requests per day.

ZooKeeper – constant cost for VMs.
FaaSKeeper – pay per each request.

Set node data of 1 kB, no watches, single request per invocation.

Availability and Acknowledgments

Availability and Acknowledgments

spcl/FaaSKeeper

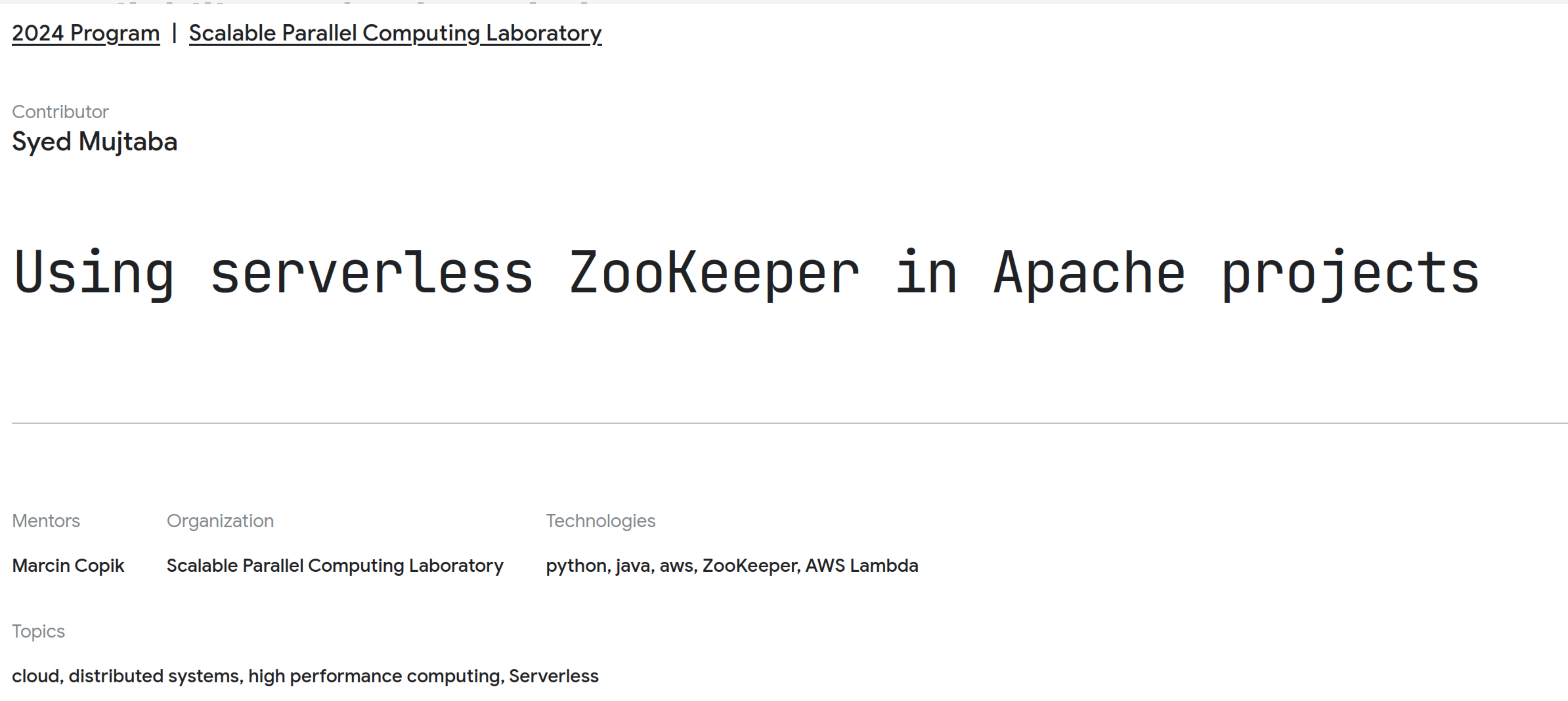
spcl/FaaSKeeper-Python

Availability and Acknowledgments

spcl/FaaSKeeper

spcl/FaaSKeeper-Python

aws Google Cloud Google Summer of Code



[2024 Program](#) | [Scalable Parallel Computing Laboratory](#)

Contributor

Syed Mujtaba

Using serverless ZooKeeper in Apache projects

Mentors	Organization	Technologies
Marcin Copik	Scalable Parallel Computing Laboratory	python, java, aws, ZooKeeper, AWS Lambda

Topics

cloud, distributed systems, high performance computing, Serverless

Conclusions

More of SPCL's research:

youtube.com/@spcl	180+ Talks
twitter.com/spcl_eth	1.4K+ Followers
github.com/spcl	3.8K+ Stars

… or spcl.ethz.ch

Paper

Projects

Conclusions

What is ZooKeeper?

APACHE ZooKeeper™

APACHE HBASE

APACHE Spark™

Apache BookKeeper™

More of SPCL's research:

youtube.com/@spcl	180+ Talks
twitter.com/spcl_eth	1.4K+ Followers
github.com/spcl	3.8K+ Stars

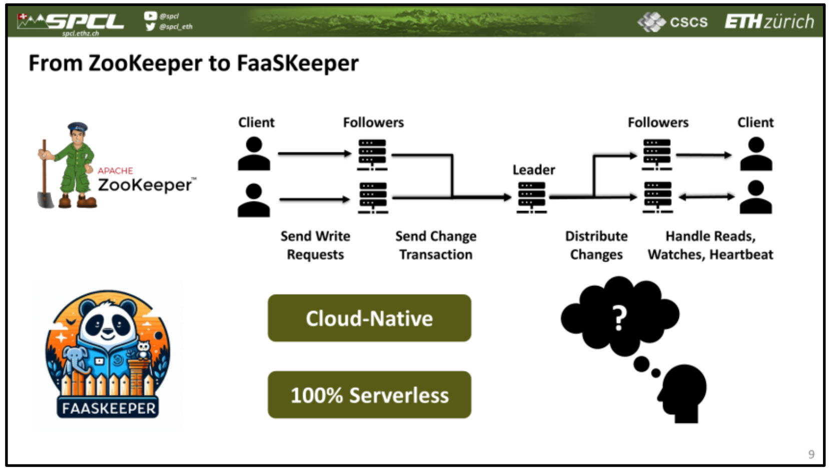
… or spcl.ethz.ch

Paper

Projects

Conclusions

What is ZooKeeper?



From ZooKeeper to FaaSKeeper

Cloud-Native

100% Serverless

More of SPCL's research:

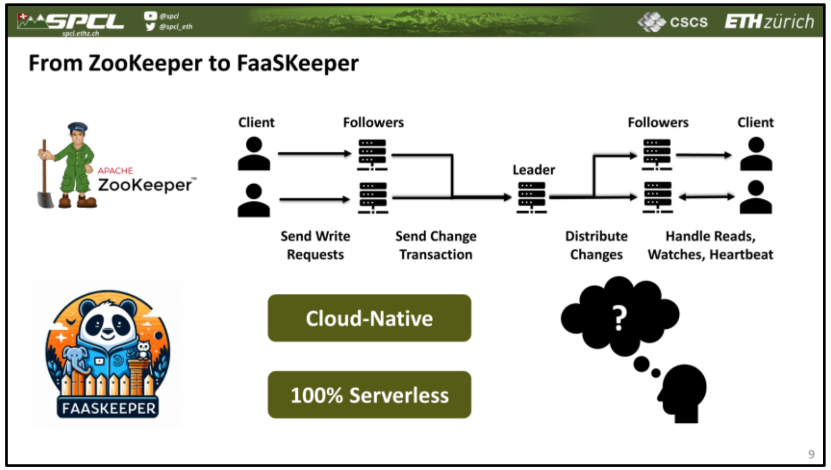
youtube.com/@spcl	180+ Talks
twitter.com/spcl_eth	1.4K+ Followers
github.com/spcl	3.8K+ Stars

… or spcl.ethz.ch

Paper **Projects**

Conclusions

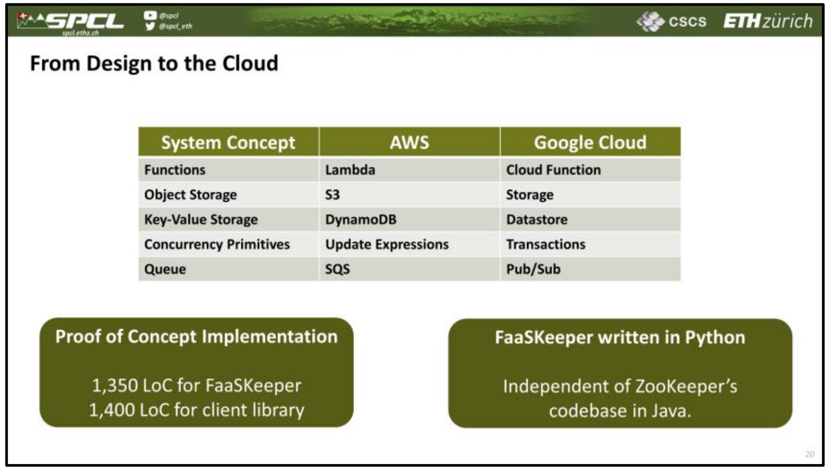
What is ZooKeeper?



From ZooKeeper to FaaSKeeper

Cloud-Native

100% Serverless



From Design to the Cloud

System Concept	AWS	Google Cloud
Functions	Lambda	Cloud Function
Object Storage	S3	Storage
Key-Value Storage	DynamoDB	Datastore
Concurrency Primitives	Update Expressions	Transactions
Queue	SQS	Pub/Sub

Proof of Concept Implementation

1,350 LoC for FaaSKeeper
1,400 LoC for client library

FaaSKeeper written in Python

Independent of ZooKeeper's codebase in Java.

More of SPCL's research:

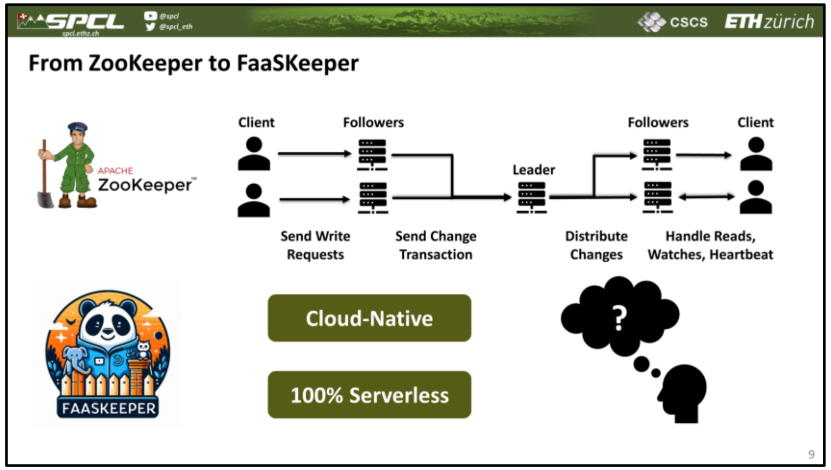
youtube.com/@spcl	180+ Talks
twitter.com/spcl_eth	1.4K+ Followers
github.com/spcl	3.8K+ Stars

… or spcl.ethz.ch

Paper **Projects**

Conclusions

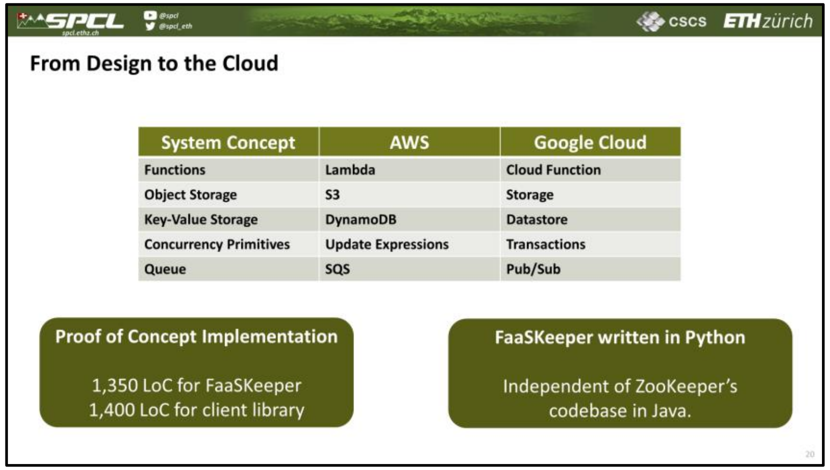
What is ZooKeeper?



From ZooKeeper to FaaSKeeper

Cloud-Native

100% Serverless



From Design to the Cloud

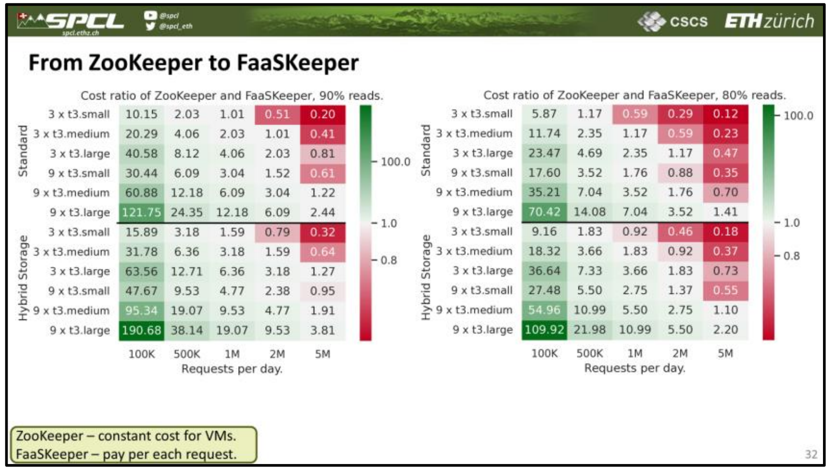
System Concept	AWS	Google Cloud
Functions	Lambda	Cloud Function
Object Storage	S3	Storage
Key-Value Storage	DynamoDB	Datastore
Concurrency Primitives	Update Expressions	Transactions
Queue	SQS	Pub/Sub

Proof of Concept Implementation

1,350 LoC for FaaSKeeper
1,400 LoC for client library

FaaSKeeper written in Python

Independent of ZooKeeper's codebase in Java.



From ZooKeeper to FaaSKeeper

Cost ratio of ZooKeeper and FaaSKeeper, 90% reads.

		100K	500K	1M	2M	5M
Standard	3 x t3.small	10.15	2.03	1.01	0.51	0.20
	3 x t3.medium	20.29	4.06	2.03	1.01	0.41
	3 x t3.large	40.58	8.12	4.06	2.03	0.81
	9 x t3.small	30.44	6.09	3.04	1.52	0.61
	9 x t3.medium	60.88	12.18	6.09	3.04	1.22
	9 x t3.large	121.75	24.35	12.18	6.09	2.44
Hybrid Storage	3 x t3.small	15.89	3.18	1.59	0.79	0.32
	3 x t3.medium	31.78	6.36	3.18	1.59	0.64
	3 x t3.large	63.56	12.71	6.36	3.18	1.27
	9 x t3.small	47.67	9.53	4.77	2.38	0.95
	9 x t3.medium	95.34	19.07	9.53	4.77	1.91
	9 x t3.large	190.68	38.14	19.07	9.53	3.81

Requests per day.

Cost ratio of ZooKeeper and FaaSKeeper, 80% reads.

		100K	500K	1M	2M	5M
Standard	3 x t3.small	5.87	1.17	0.59	0.29	0.12
	3 x t3.medium	11.74	2.35	1.17	0.59	0.23
	3 x t3.large	23.47	4.69	2.35	1.17	0.47
	9 x t3.small	17.60	3.52	1.76	0.88	0.35
	9 x t3.medium	35.21	7.04	3.52	1.76	0.70
	9 x t3.large	70.42	14.08	7.04	3.52	1.41
Hybrid Storage	3 x t3.small	9.16	1.83	0.92	0.46	0.18
	3 x t3.medium	18.32	3.66	1.83	0.92	0.37
	3 x t3.large	36.64	7.33	3.66	1.83	0.73
	9 x t3.small	27.48	5.50	2.75	1.37	0.55
	9 x t3.medium	54.96	10.99	5.50	2.75	1.10
	9 x t3.large	109.92	21.98	10.99	5.50	2.20

Requests per day.

ZooKeeper – constant cost for VMs.
FaaSKeeper – pay per each request.

More of SPCL's research:

youtube.com/@spcl — 180+ Talks

twitter.com/spcl_eth — 1.4K+ Followers

github.com/spcl — 3.8K+ Stars

… or spcl.ethz.ch

Paper

Projects