
AWS
re:Invent
NOV. 27 – DEC. 1, 2023 | LAS VEGAS, NV

NFX304

How Netflix leverages AWS for multi-Region cache replication

Prudhviraj Karumanchi
(He/Him)
Staff Software Engineer
Netflix

Sriram Rangarajan
(He/Him)
Senior Software Engineer
Netflix

Prateek Sharma
(He/Him)
Principal Solutions Architect
AWS

Spark joy

Agenda

Introduction

Global Replication

Design and Architecture

Observability

Efficiency Improvements

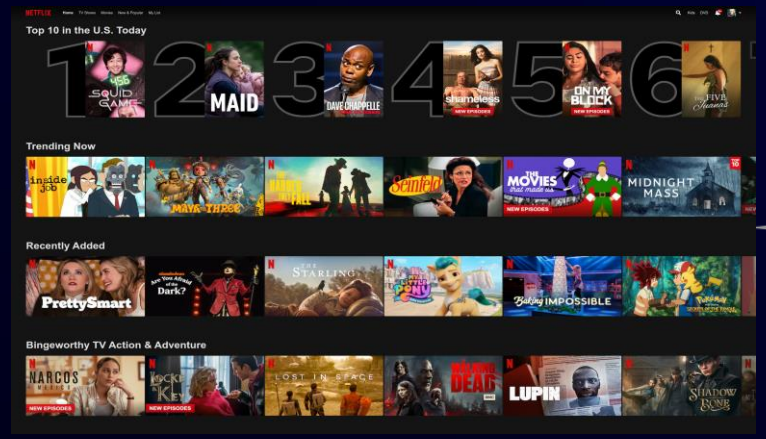
Learnings from production deployments

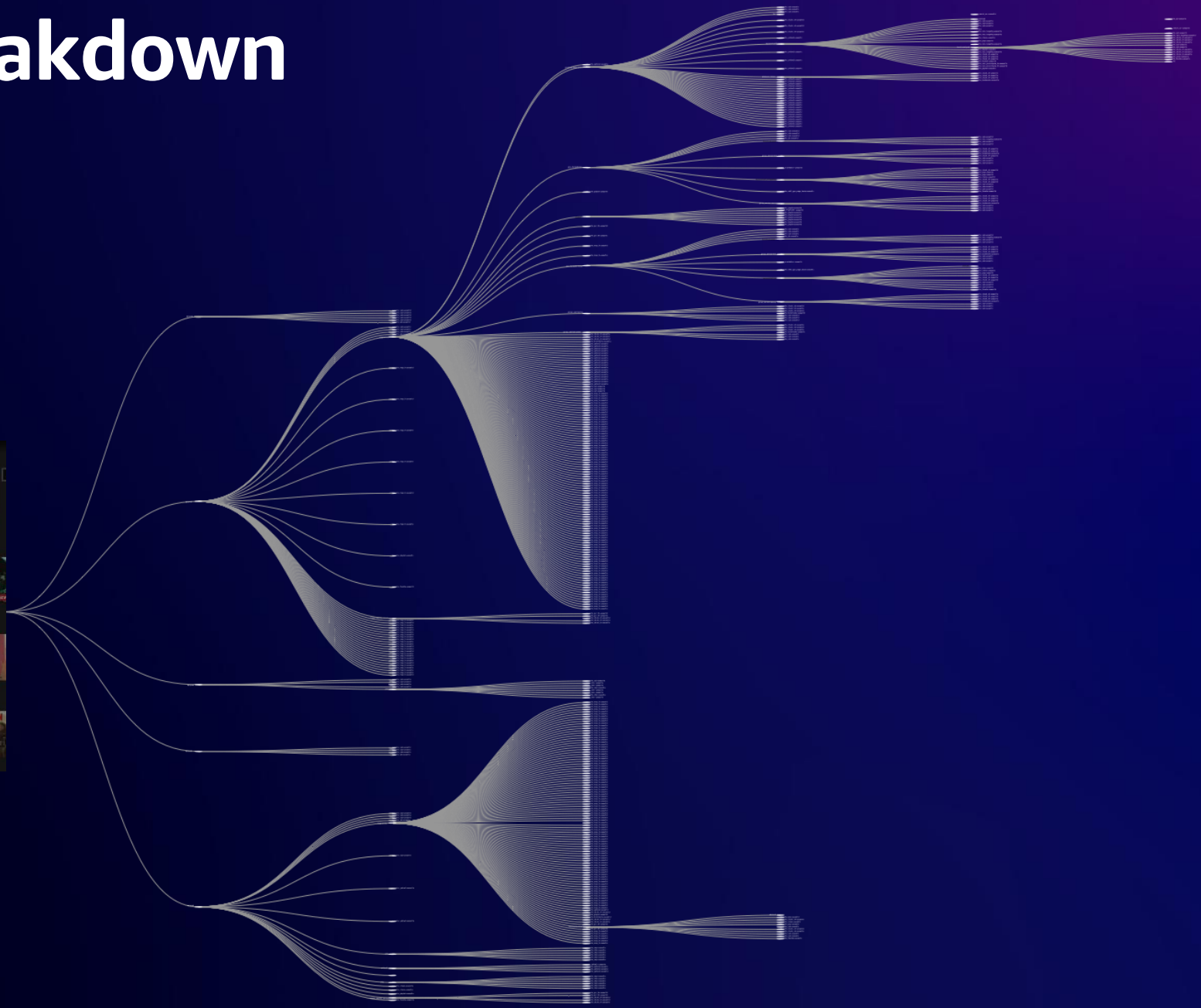
Future

Agenda

Introduction

Homepage breakdown





EVCache

Distributed, sharded, replicated key-value store

Based on Memcached

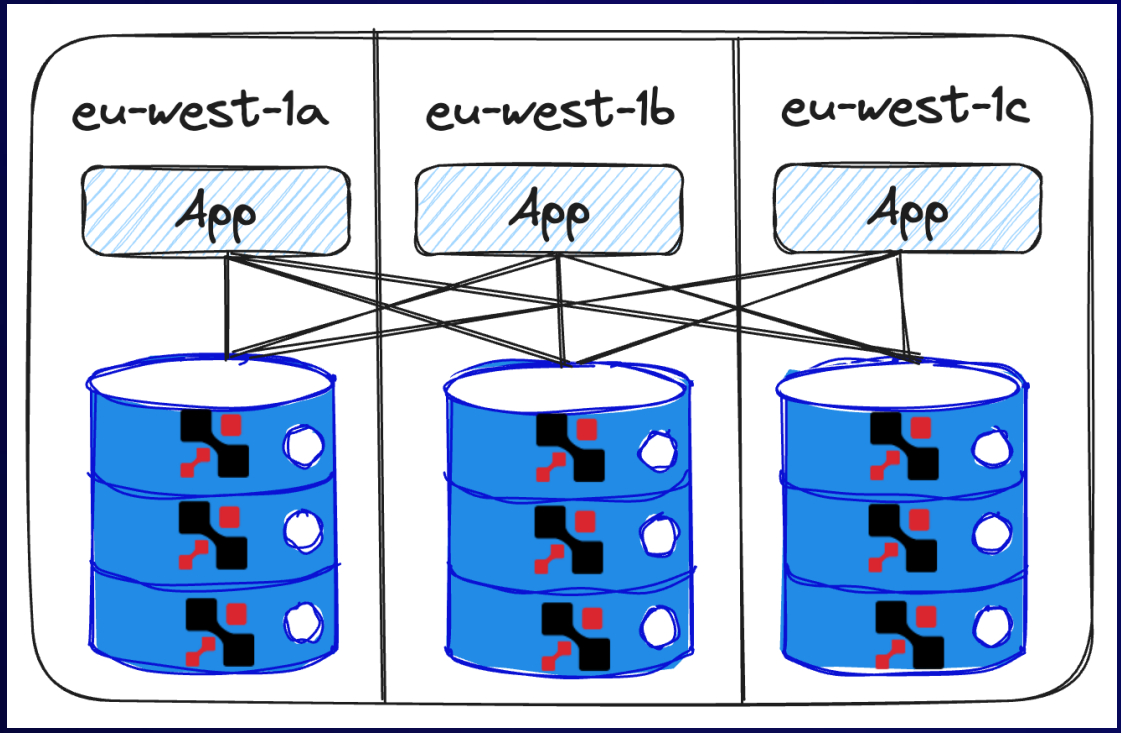
Tunable in-Region and global replication

Resilient to failures

Topology aware

Linearly scalable

Seamless deployments



EVCache

4 regions

~194 clusters

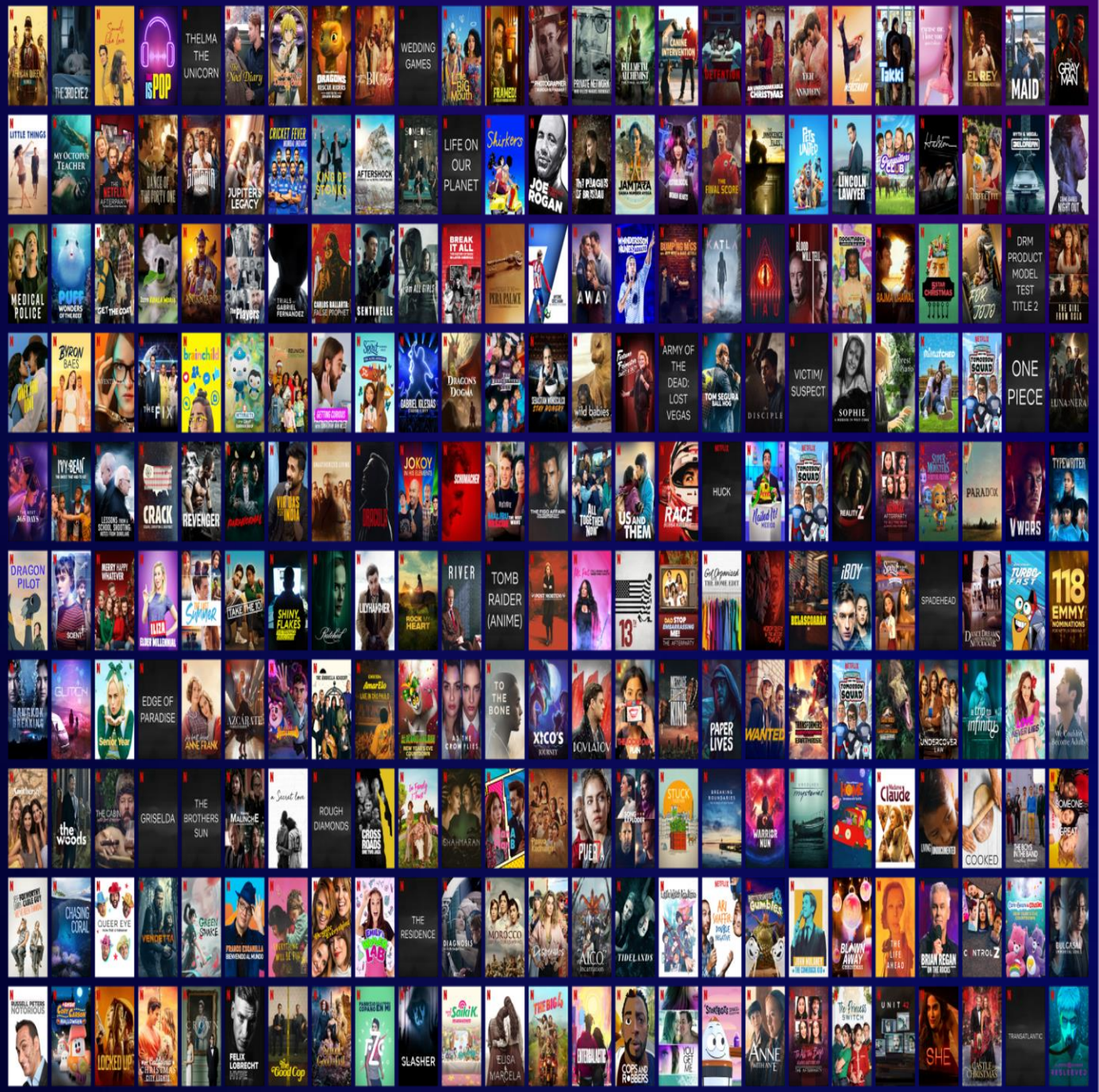
~22,000 servers

~30,000,000 replication

~400,000,000 ops/sec

~2,000,000,000,000 items

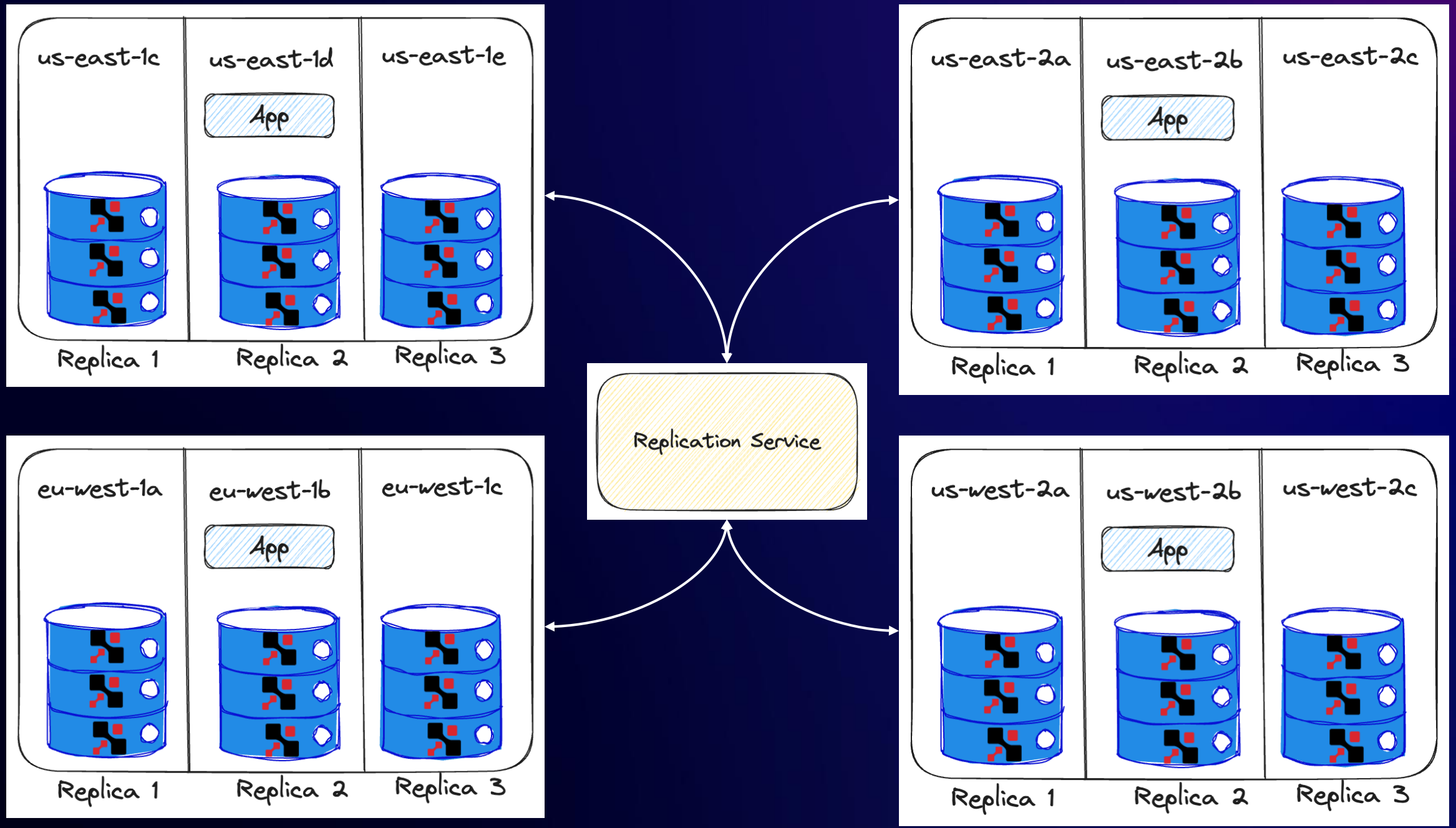
~14,300,000,000,000,000 bytes



Agenda

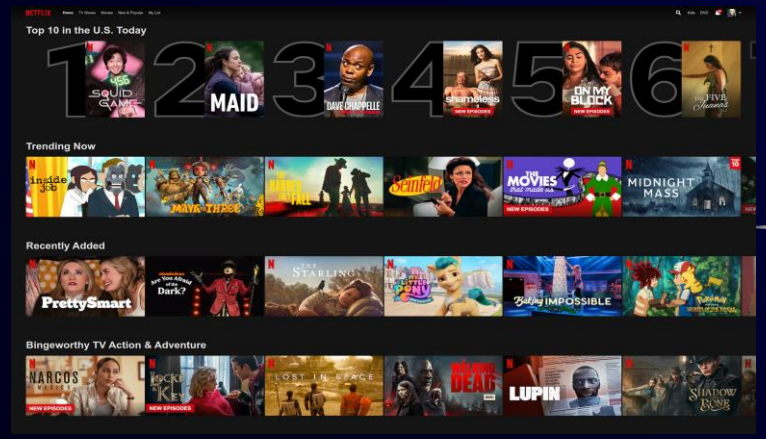
Introduction

Global Replication

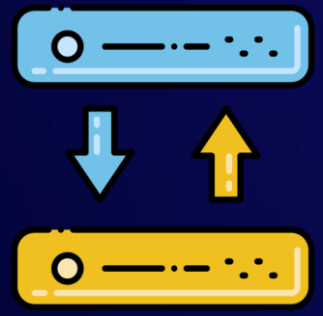


Why Global Replication ?

Why Global Replication ?



Why Global Replication ?

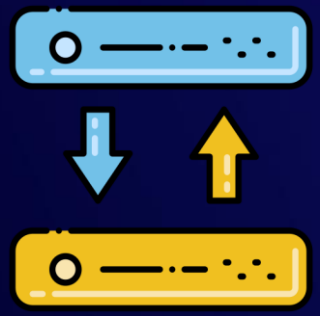


Region Failover

Empty Cache

Database

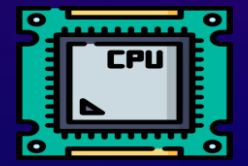
Why Global Replication ?



Region Failover

Empty Cache

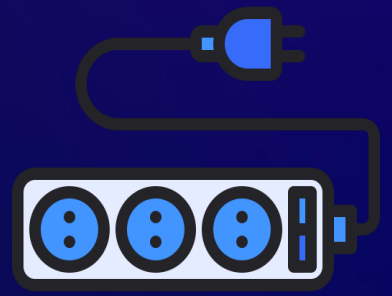
Compute Intensive



Replication Design Goals

Latency

Tunables



Extensibility

Replication Design Requirements

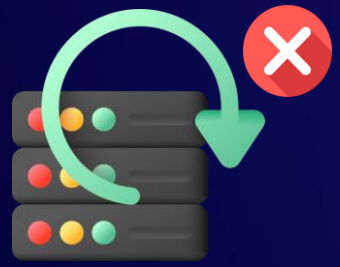
High availability

>

Best Effort Consistency

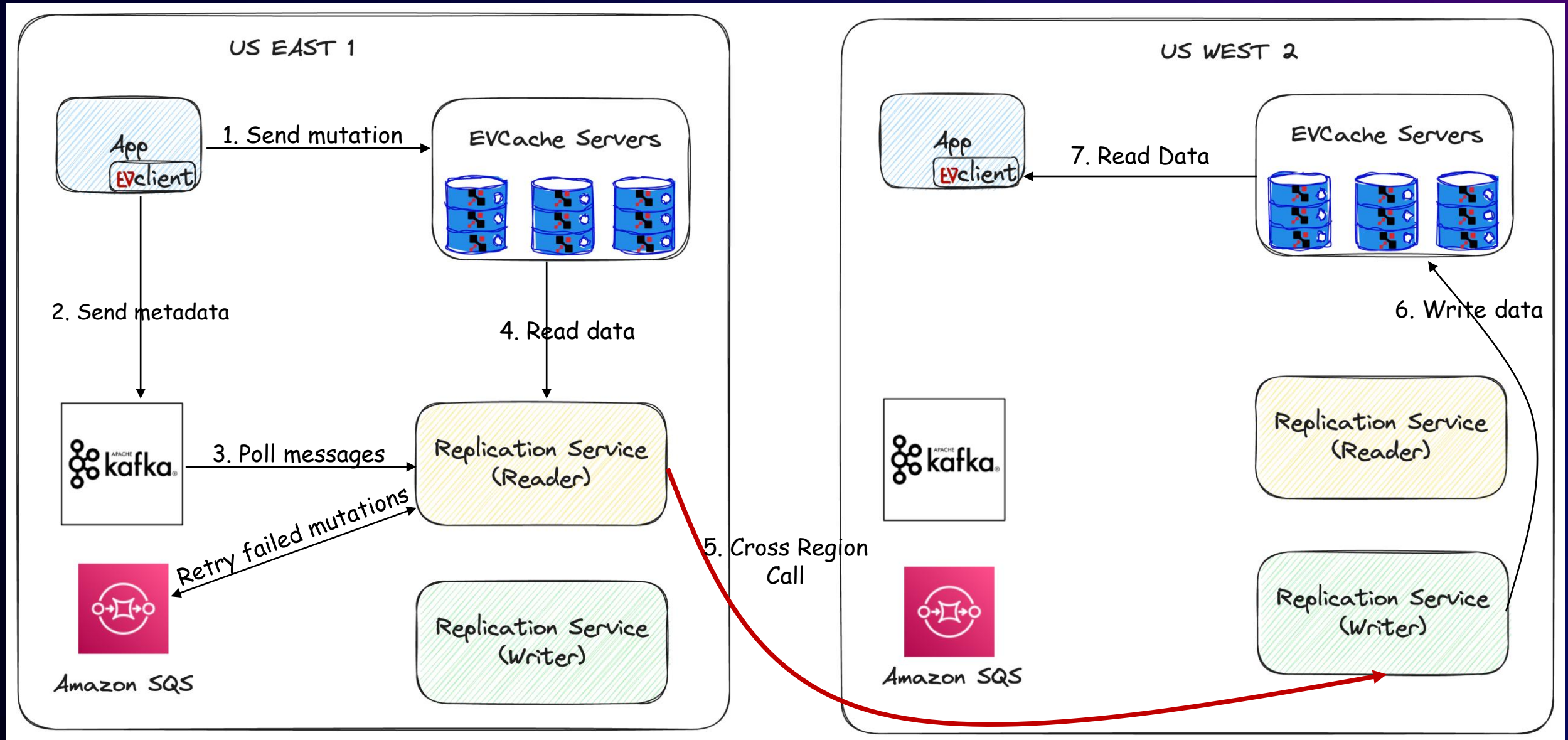
Design Non Requirements

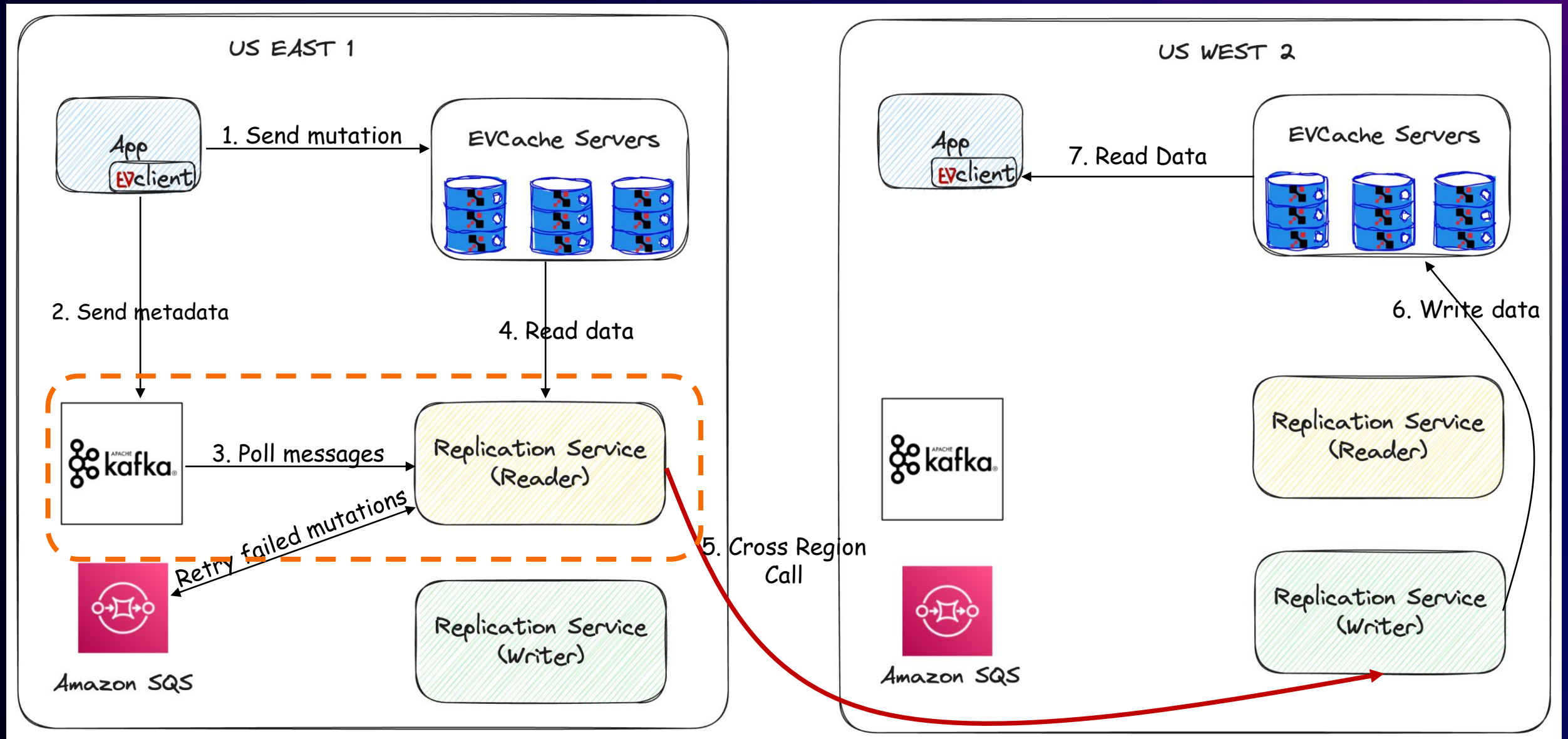
Global locking



Partial-commit rollbacks

Transactional updates





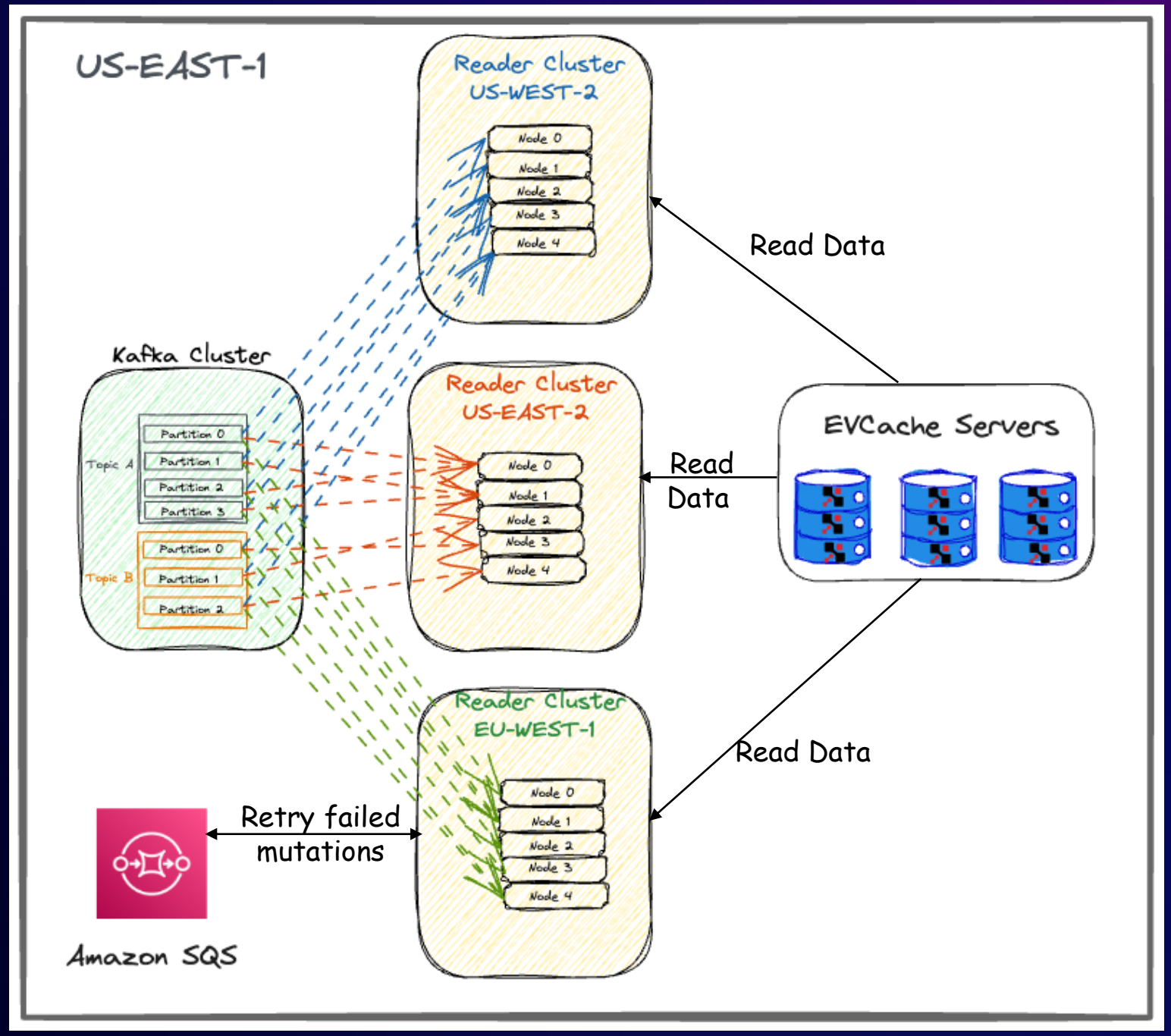
Readers

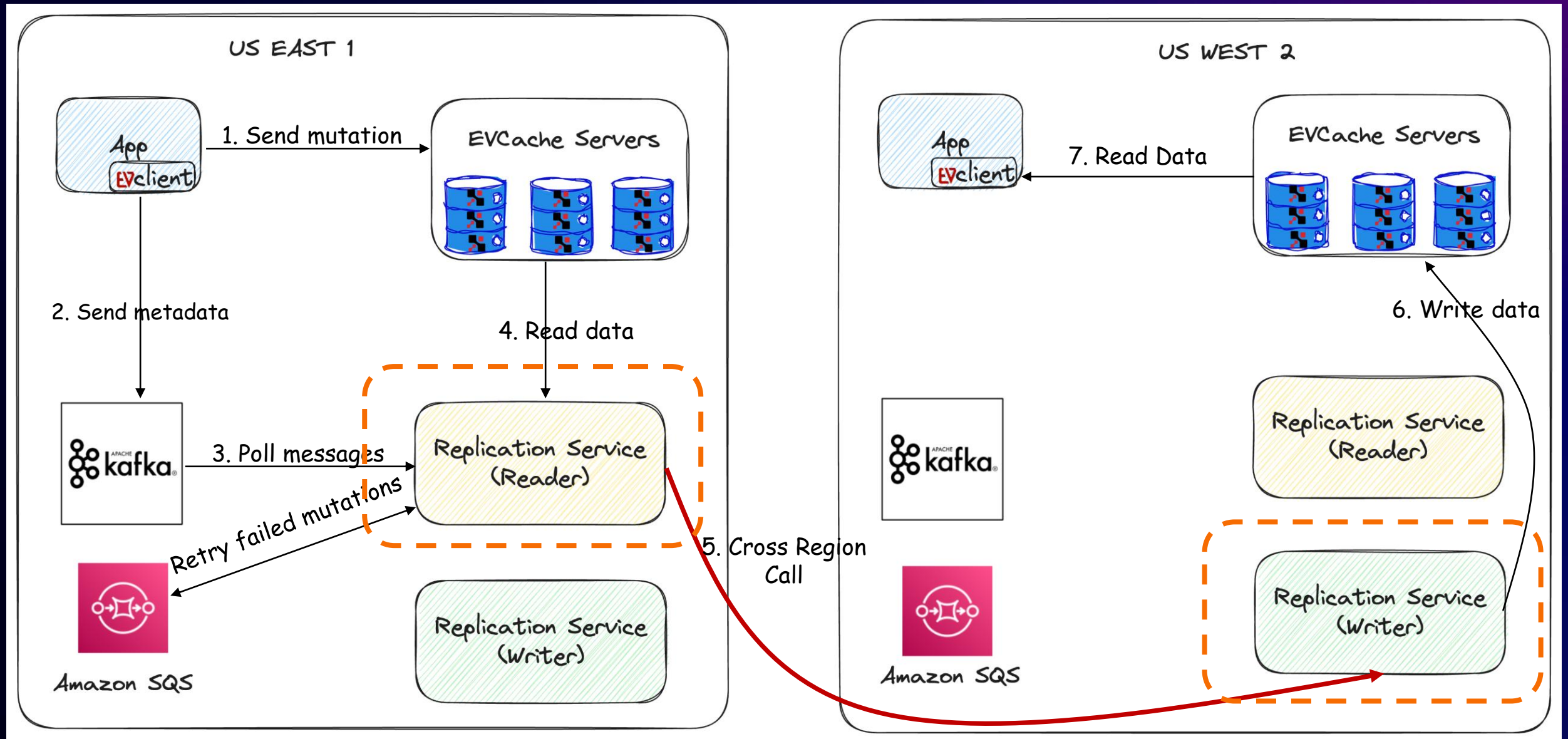
Different consumer groups reading from the
Kafka cluster

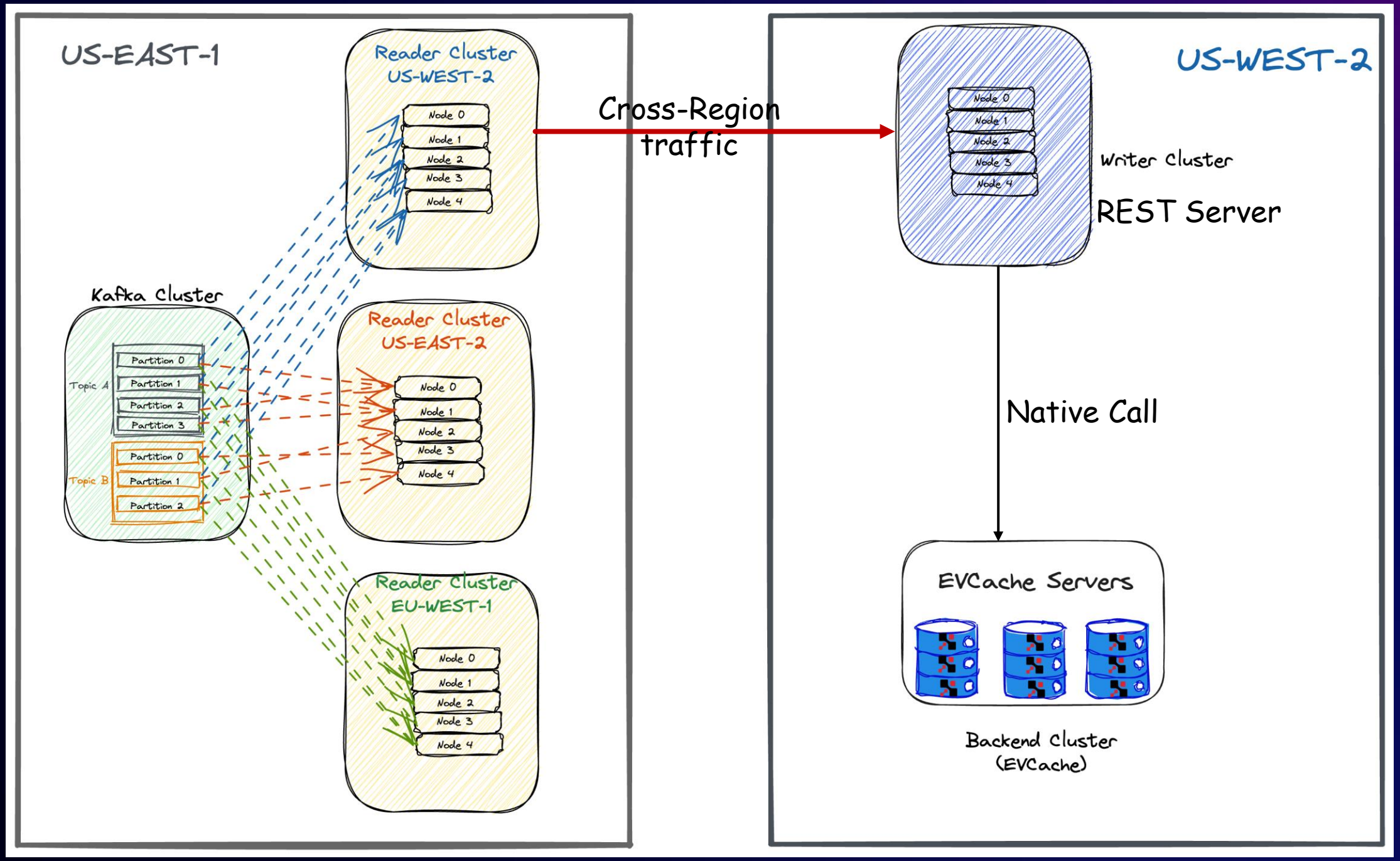
Apply transformations of the incoming mutations

Perform local read

SQS






US-EAST-1
Reader Cluster
US-WEST-2
Node 0
Node 1
Node 2
Node 3
Node 4
Cross-Region
traffic
US-WEST-2
Node 0
Node 1
Node 2
Node 3
Node 4
Writer Cluster
REST Server
Kafka Cluster
Topic A
Partition 0
Partition 1
Partition 2
Partition 3
Topic B
Partition 0
Partition 1
Partition 2
Reader Cluster
US-EAST-2
Node 0
Node 1
Node 2
Node 3
Node 4
Native Call
Reader Cluster
EU-WEST-1
Node 0
Node 1
Node 2
Node 3
Node 4
EVCache Servers
Backend Cluster
(EVCache)

Extensibility

Generic Replication Service

Write Ahead Log

Delay Queues

Extensibility

	Caching	Write Ahead Log
Kafka Configurations	High Availability	High Durability
unclean.leader.election.enabled	true	false
default.replication.factor	2	3
min.insync.replicas	1	2
acks	1	all

Design Advantages

Independent scalability

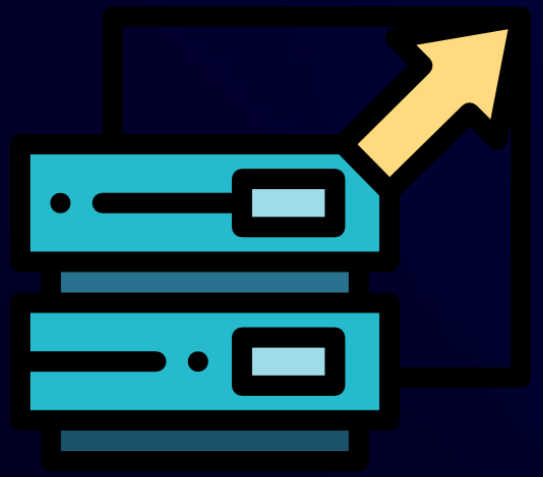
Predictable end-to-end latency

Tunables

Cost efficient

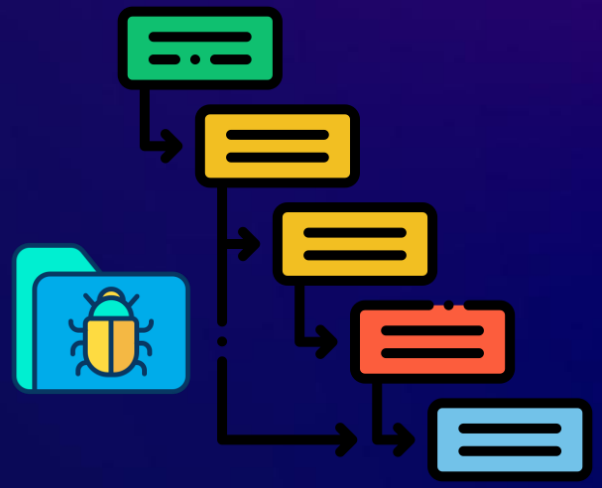
Back pressure

Design Pitfalls



Scaling policies

Message duplication



Cascading failures

Agenda

Introduction

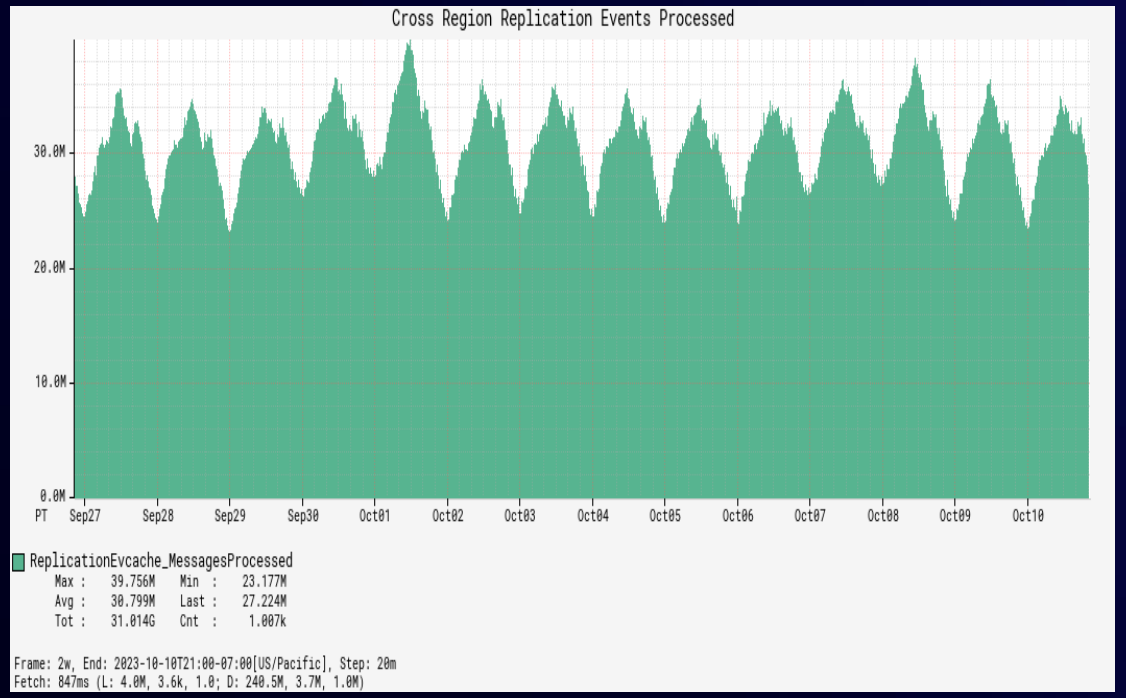
Global Replication

Design and Architecture

Observability

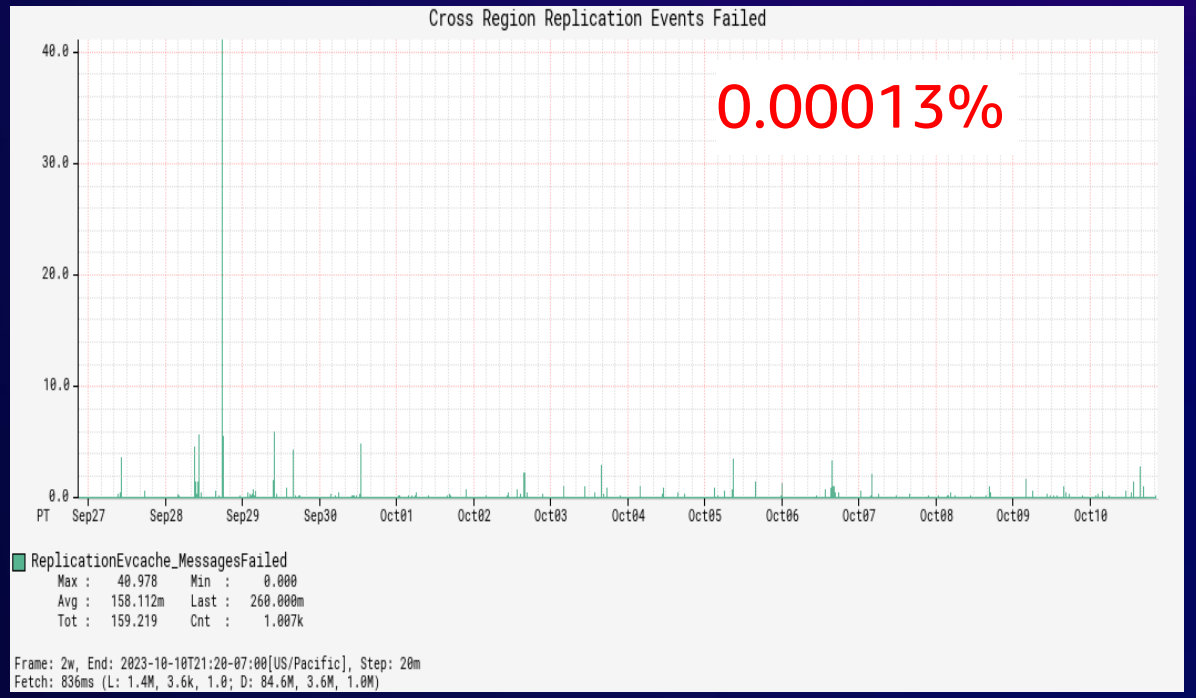
Observability & Alerts

Events Processed



Max : ~40 Million Replication Events
Avg: ~31 Million Replication Events

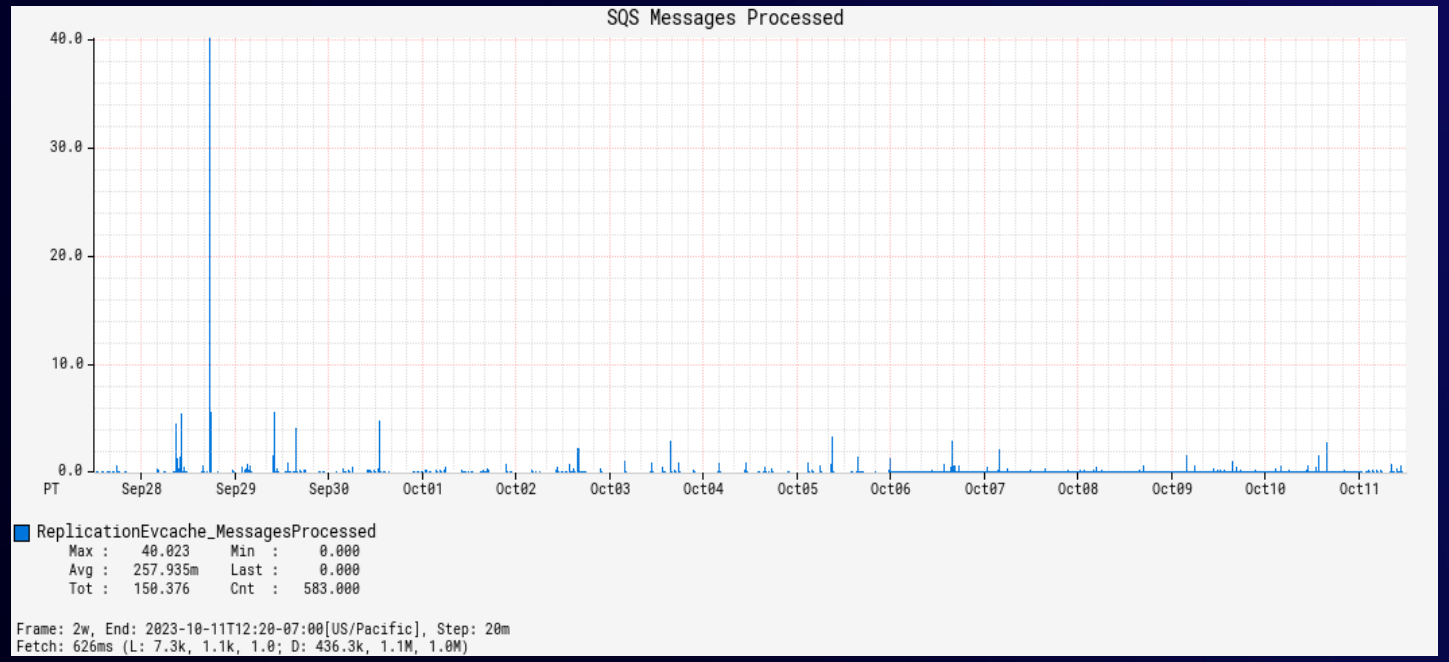
Events Failed



Max : ~40 Failure

Alerts & Observability

Events Retried with SQS



Number of events

Time

Max : ~40 Failure

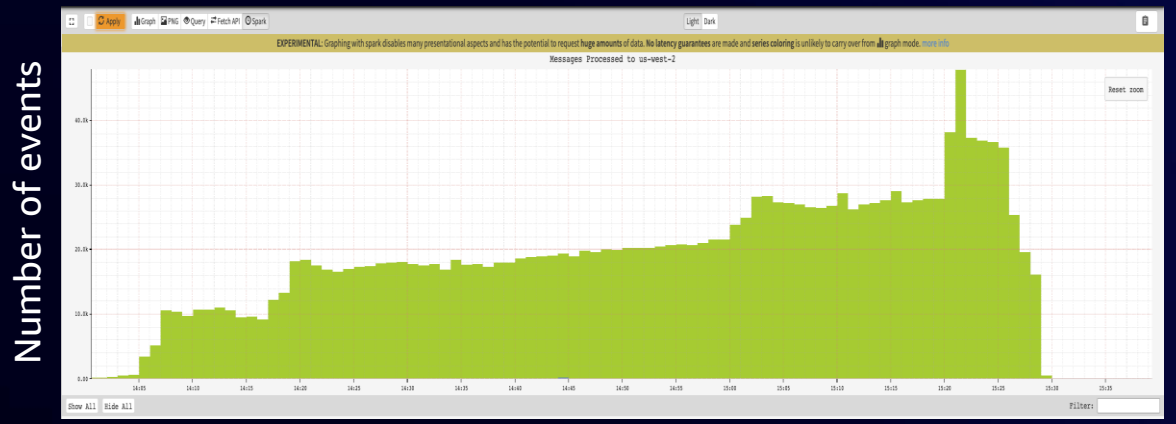
~ 0% Failure

Observability & Alerts

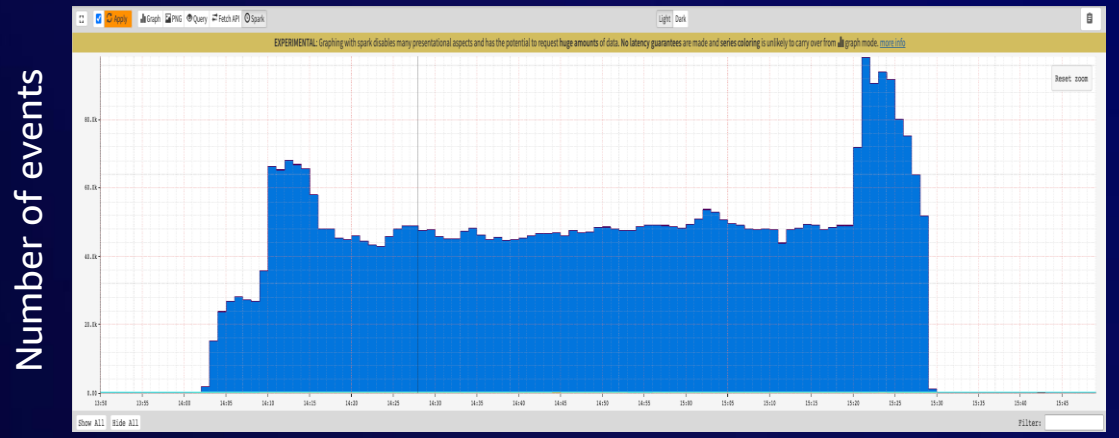
CRR Event Failure

Number of failures > 1000 per minute
5 minutes in a 10 min window

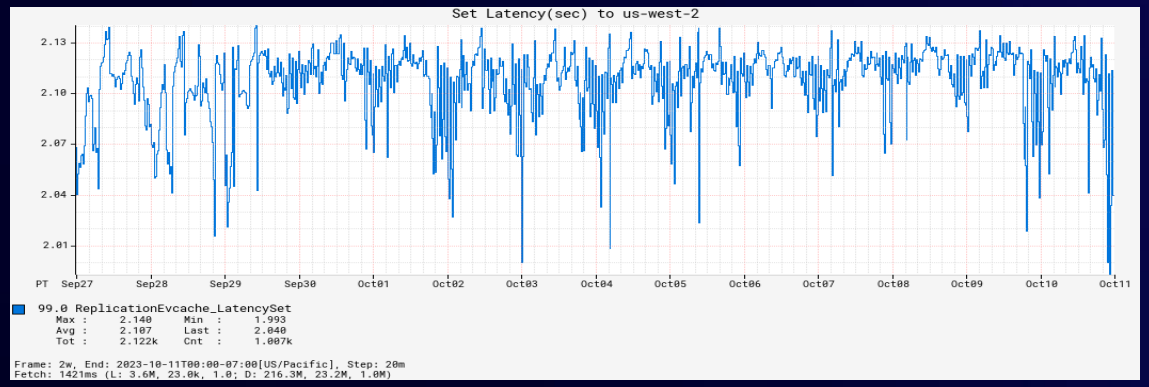
Events Retried with SQS



Failed Replication events



Observability – Latencies

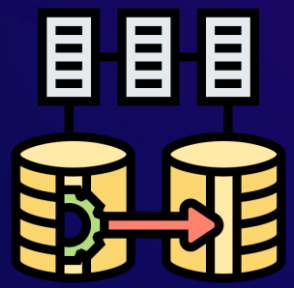


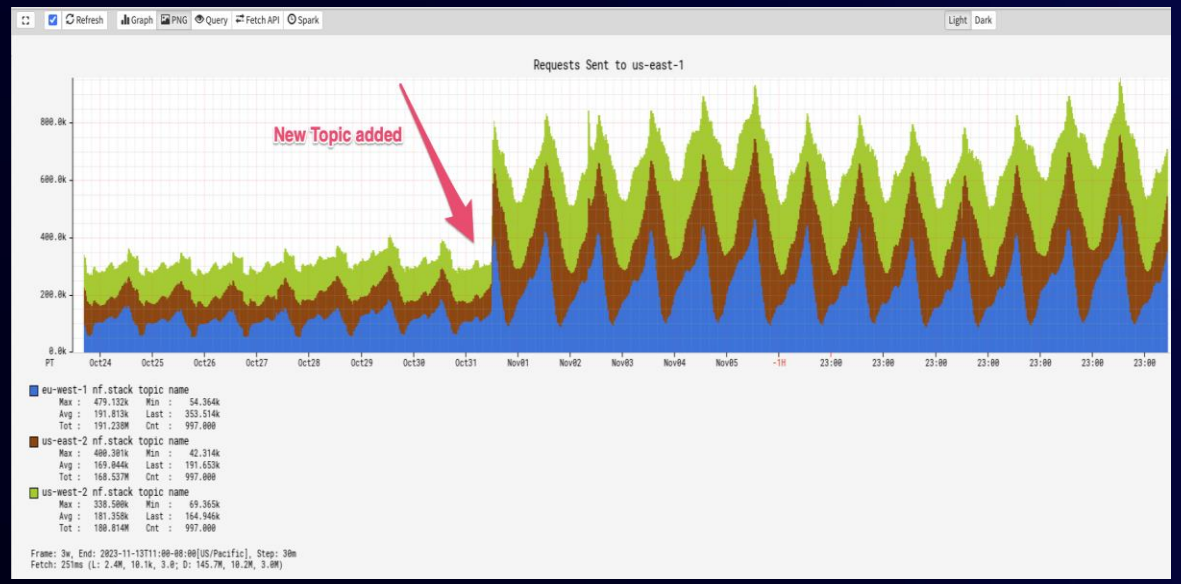
Set Latency
Avg: ~2.10 seconds
Max: ~2.14 seconds

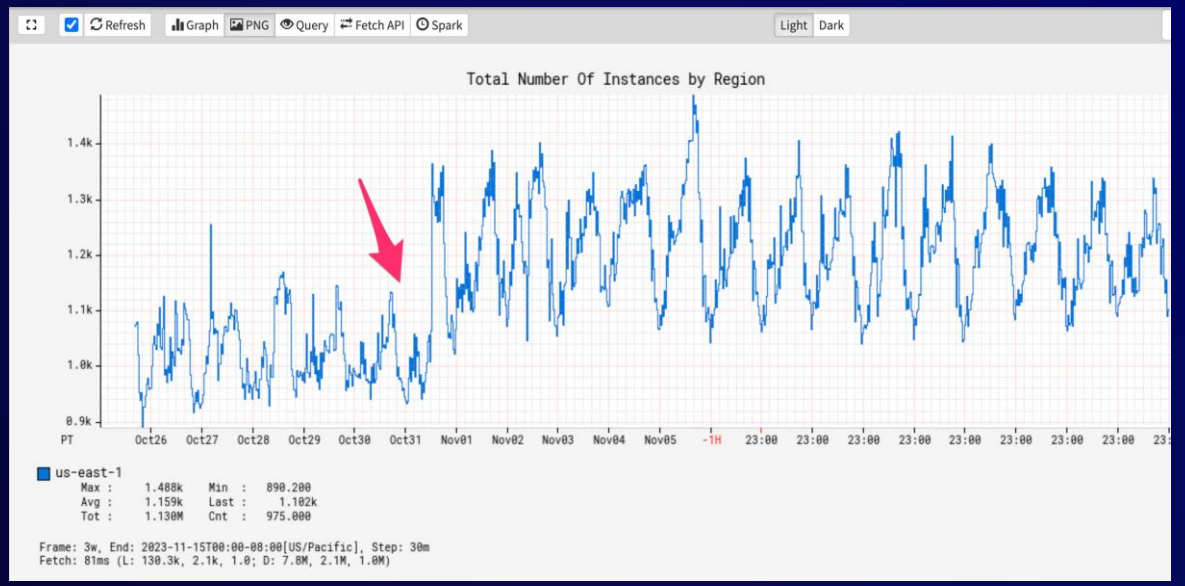
Max: ~500 seconds

Latency Alert

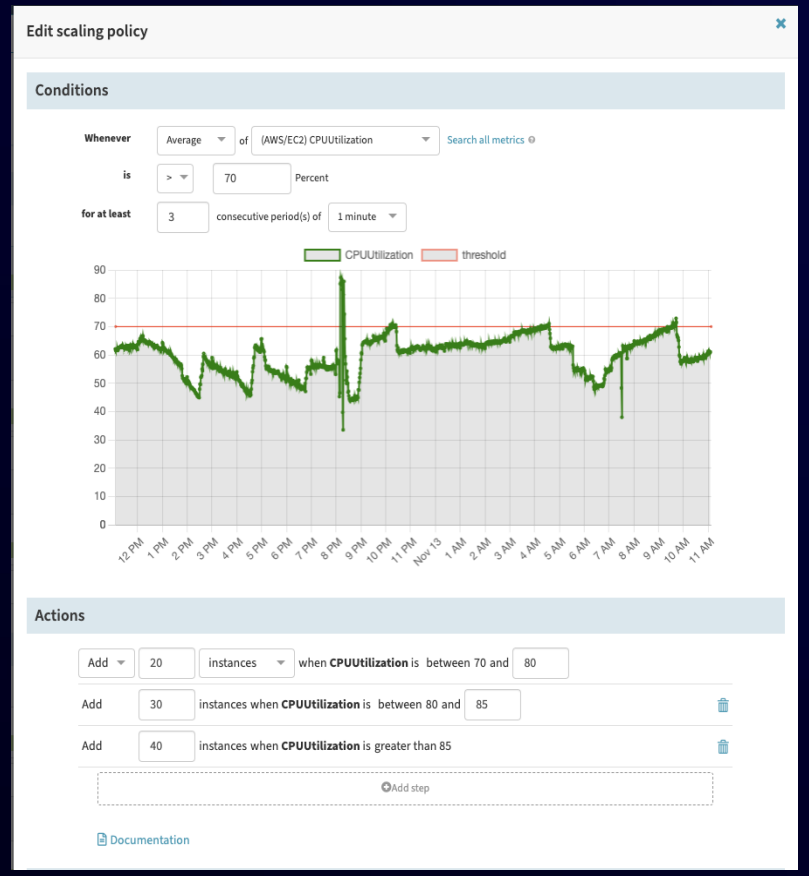
Observability – Autoscaling



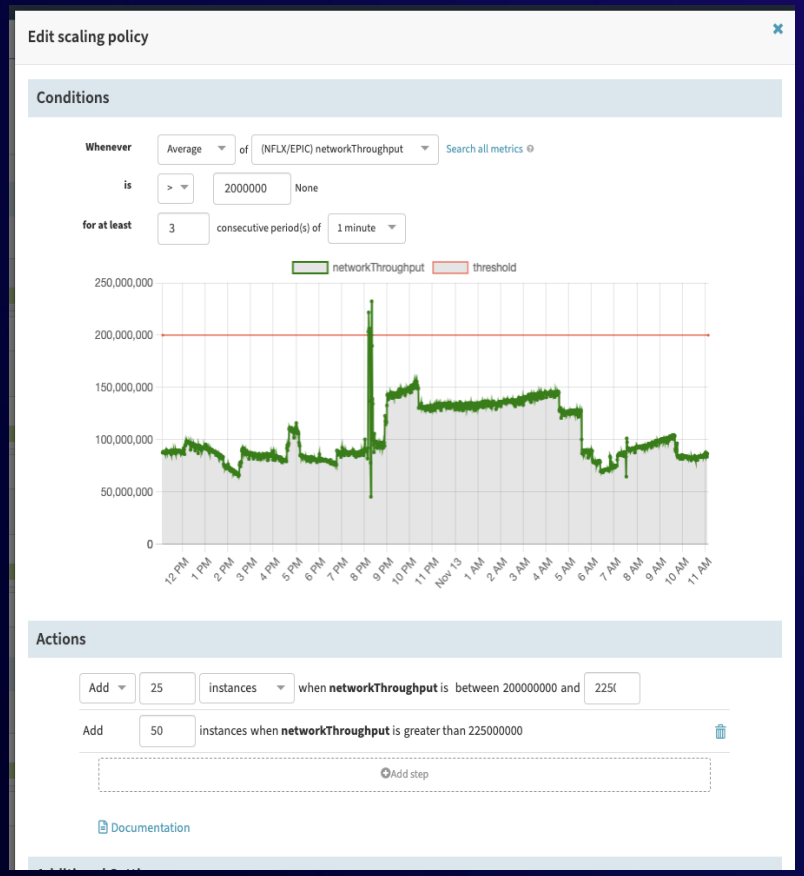




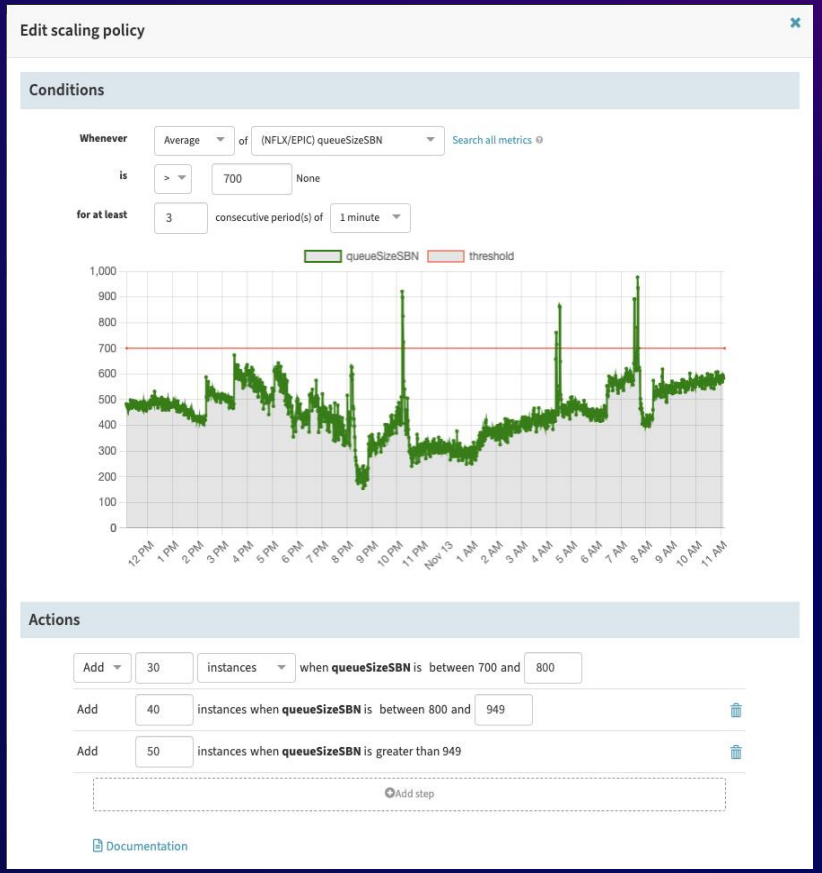
Observability – Autoscaling



CPU > 70%



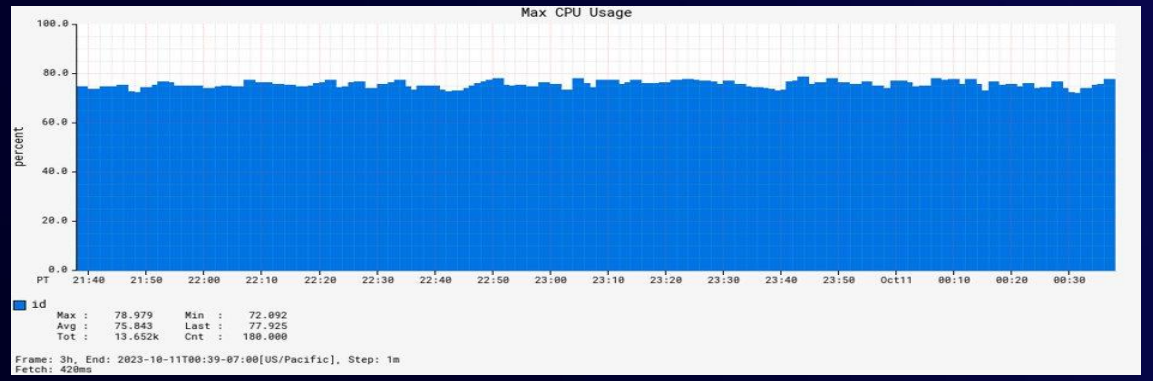
Network
> 200000000 bytes



Queue Size > 700

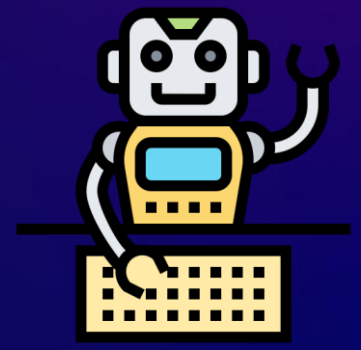
Observability – Resource Utilization

Max CPU Utilization ~ 70%

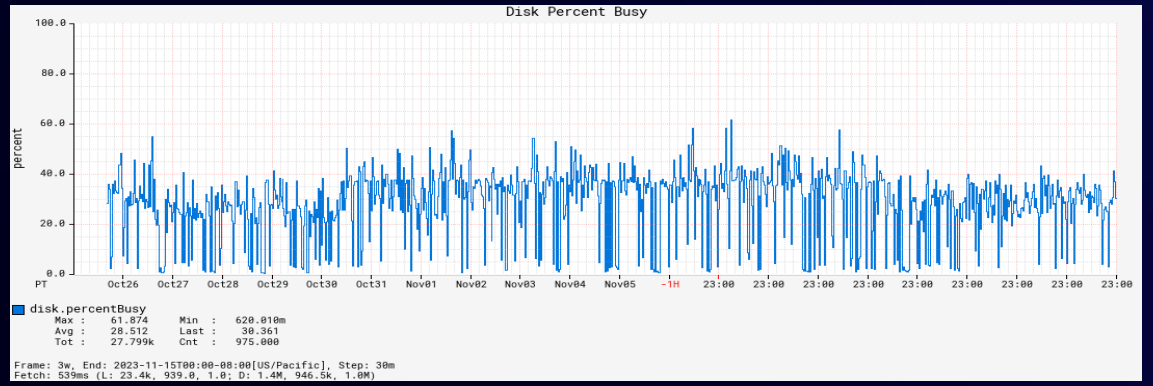


Percentage

Time



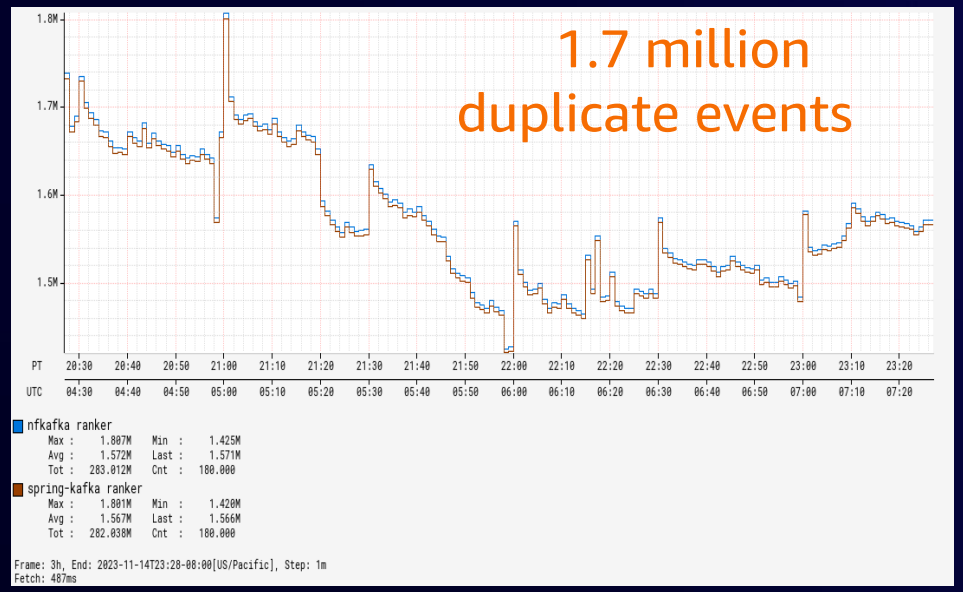
Max Disk Utilization ~ 40%

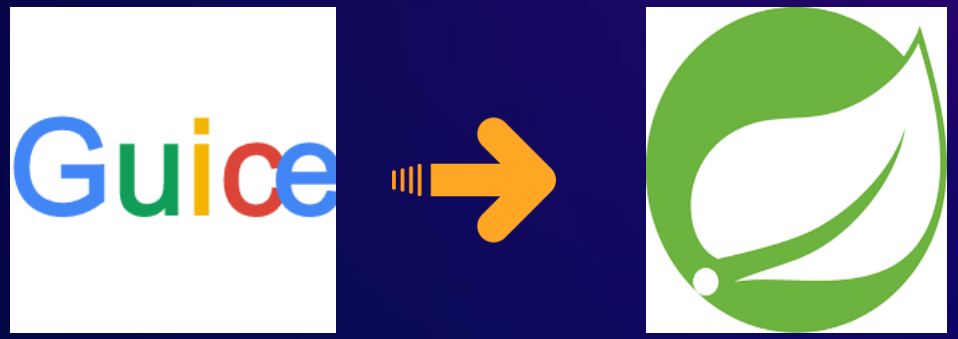


Percentage

Time

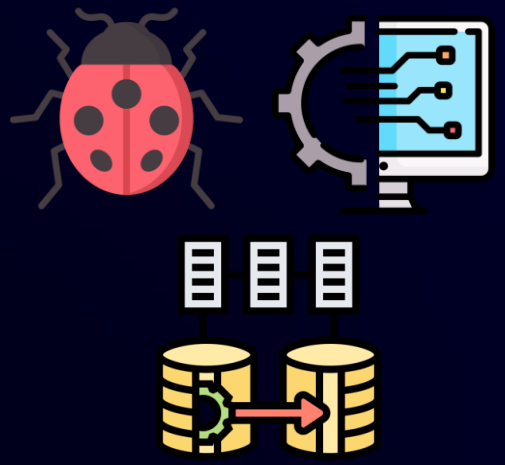
Observability & Alerts – Double Replication

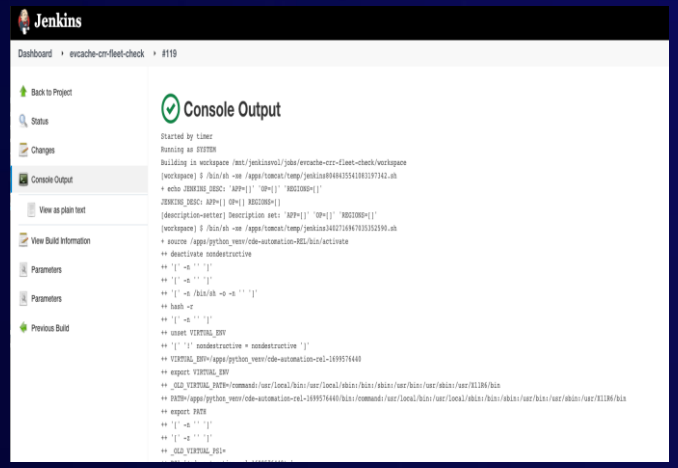




Nfkafka Client

Spring Kafka





Agenda

Introduction

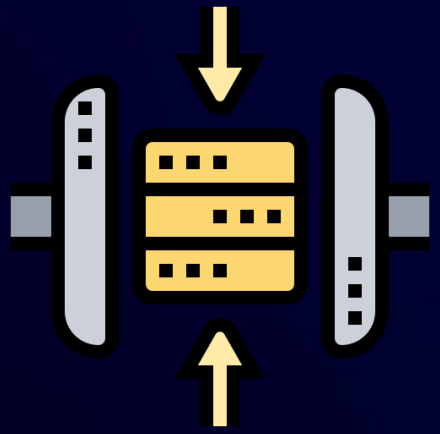
Global Replication

Design and Architecture

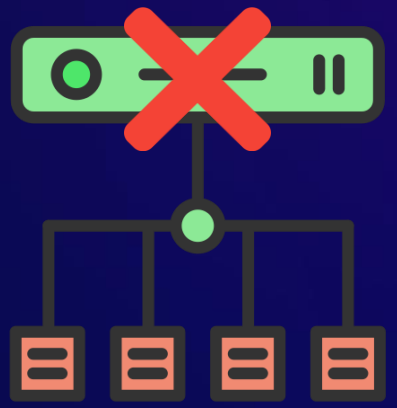
Observability

Efficiency Improvements

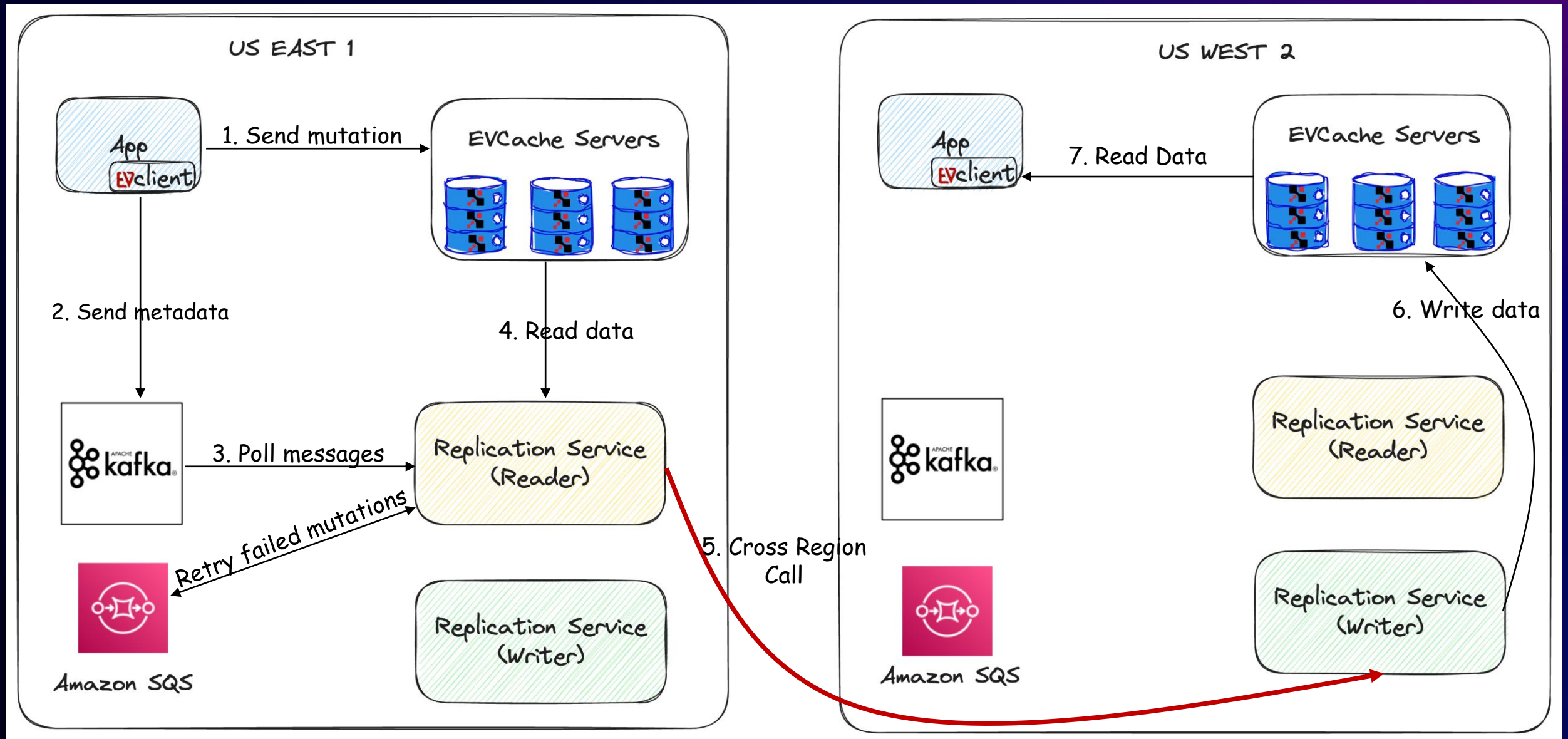
Efficiency Improvements



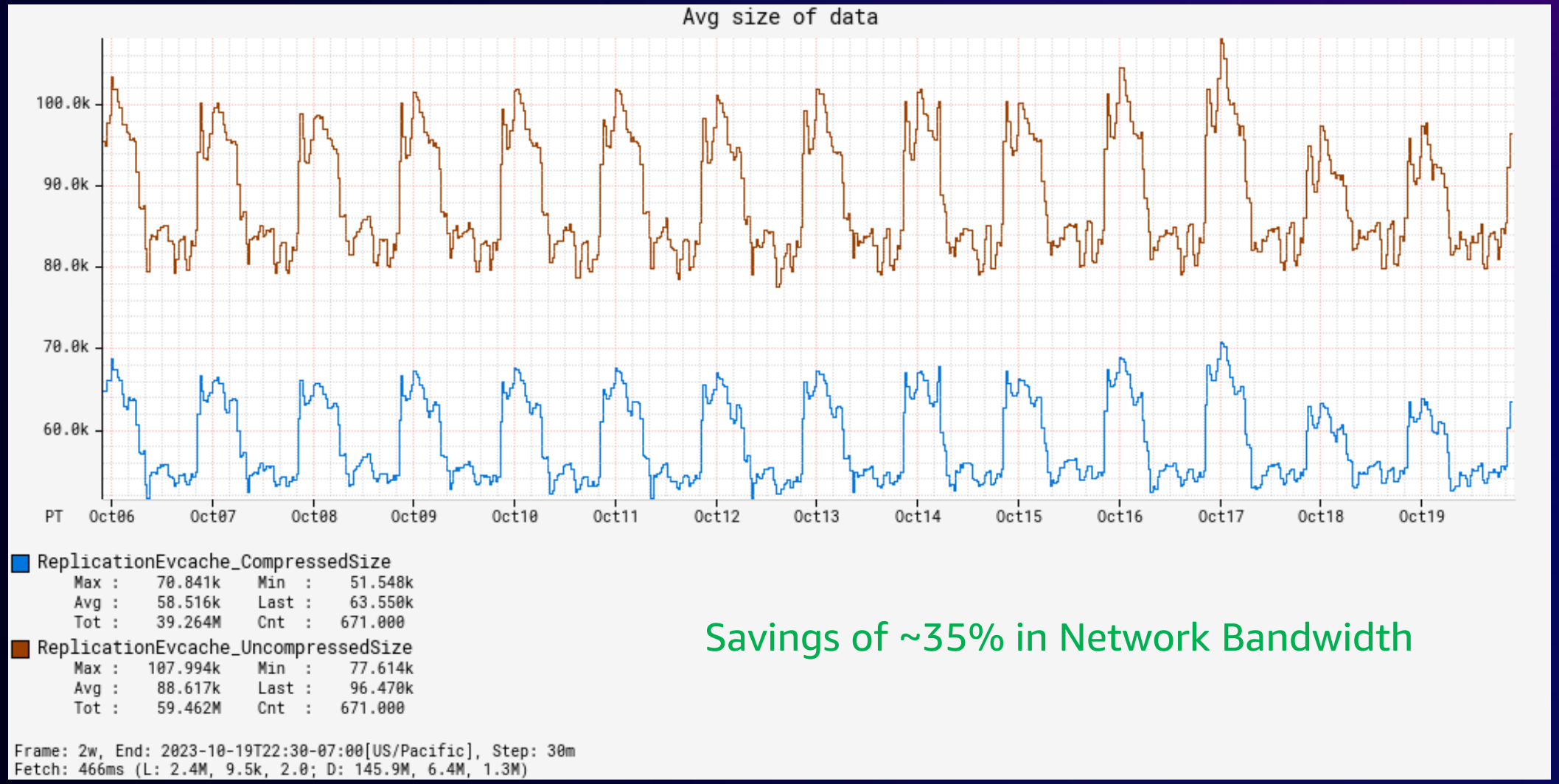
Batch compression



Removing NLBs



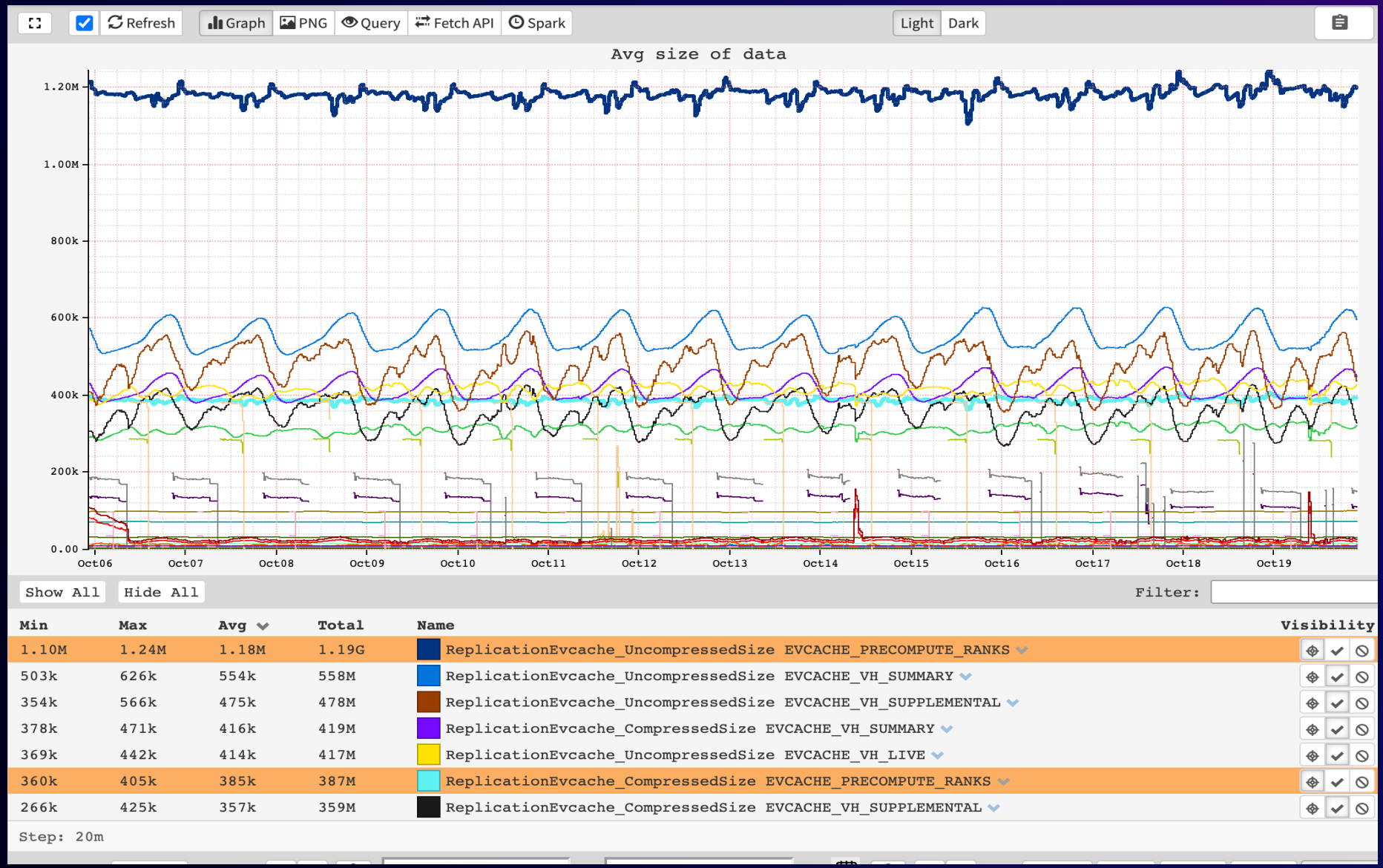
Batch compression – Across all the use cases



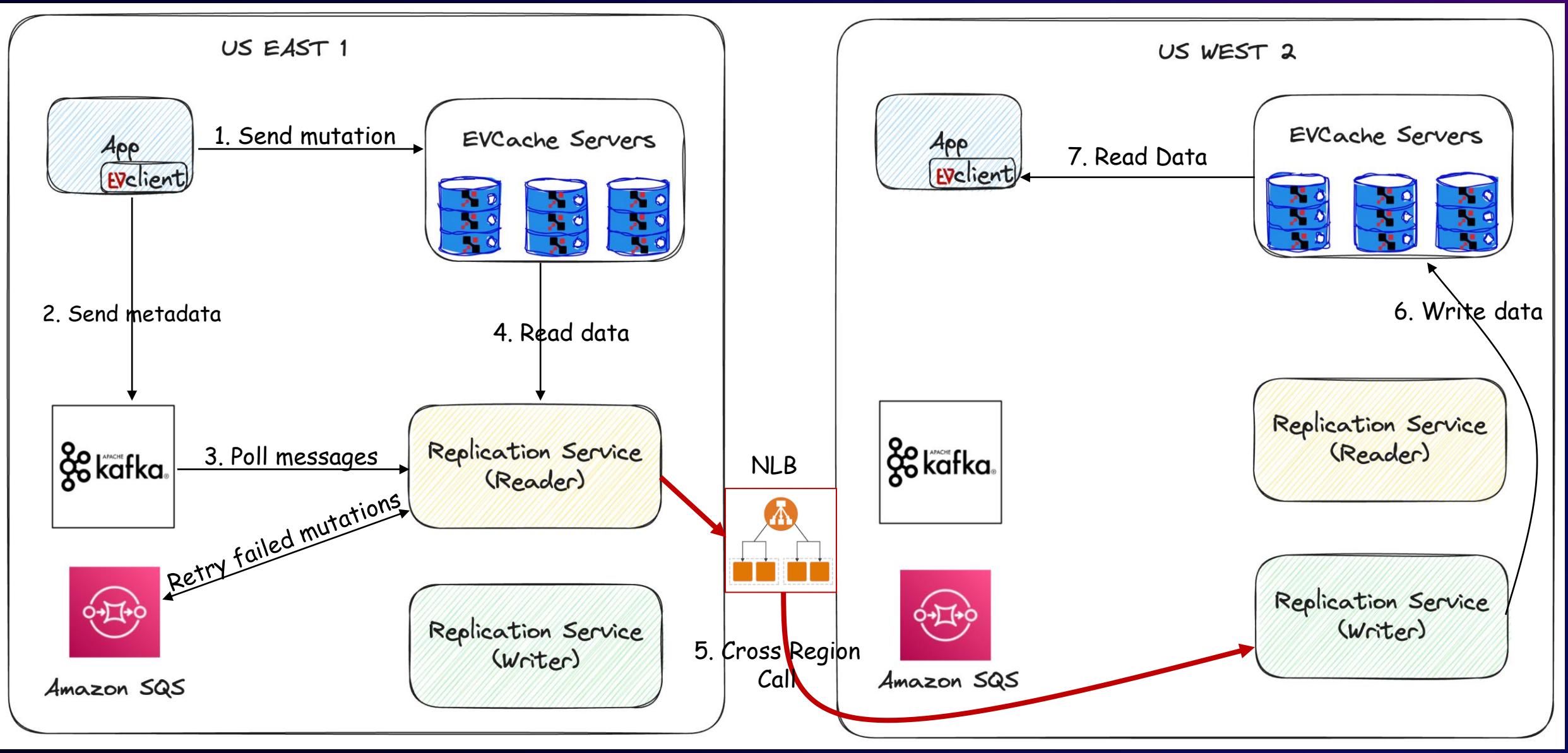
Savings of ~35% in Network Bandwidth

Batch compression – Per use case distribution

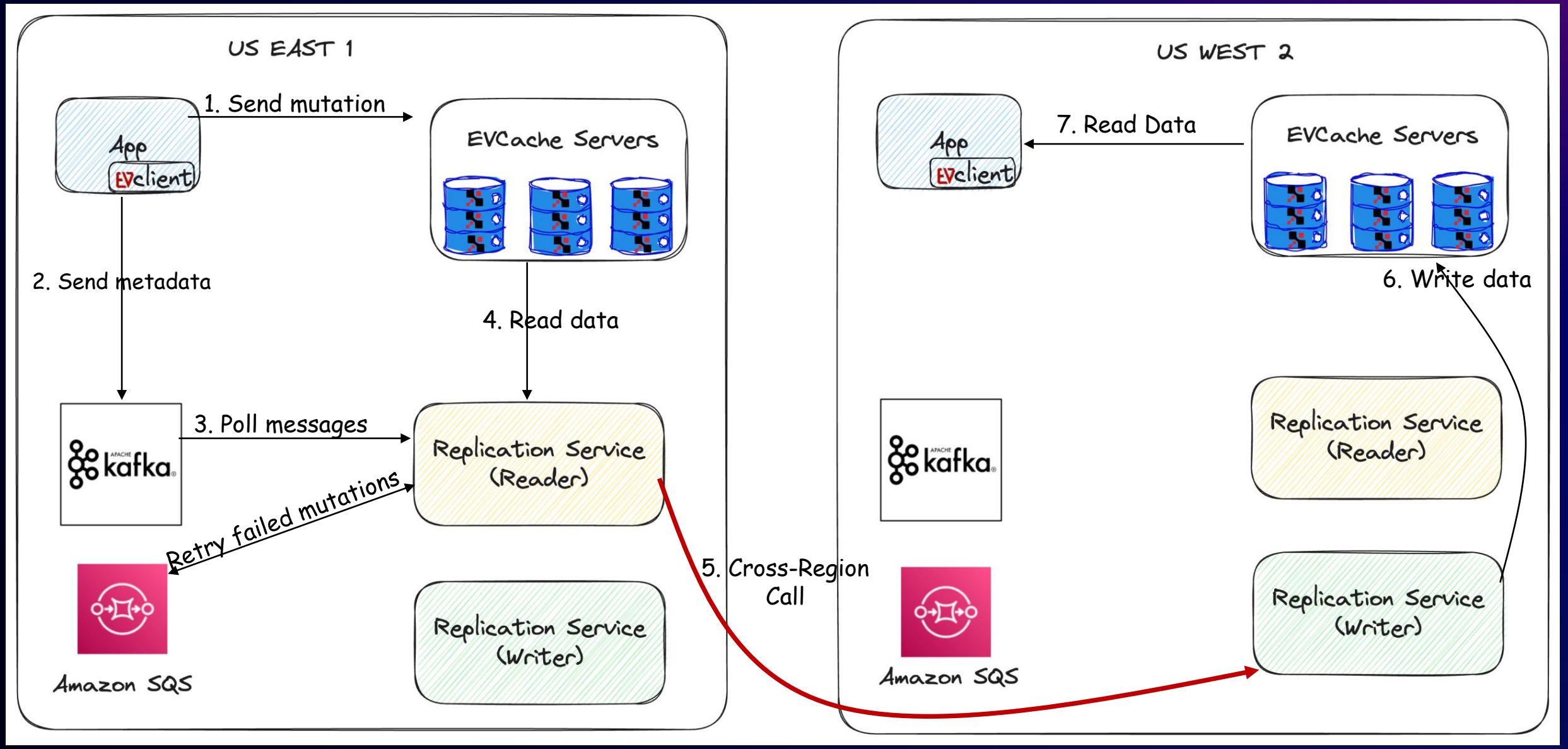
Size in KB



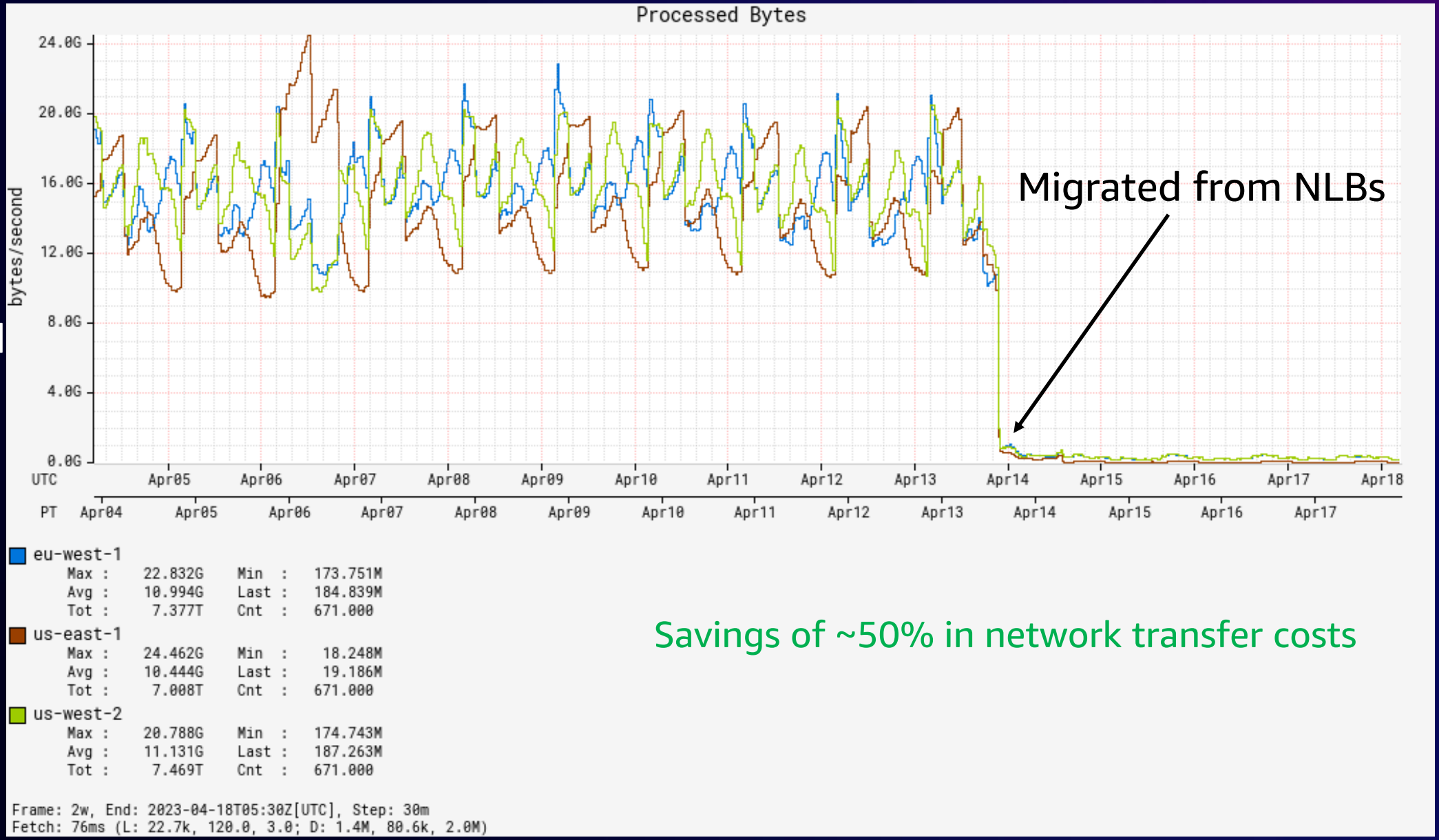
Network Load Balancer



Client Side Load Balancing



NLB Processed Bytes

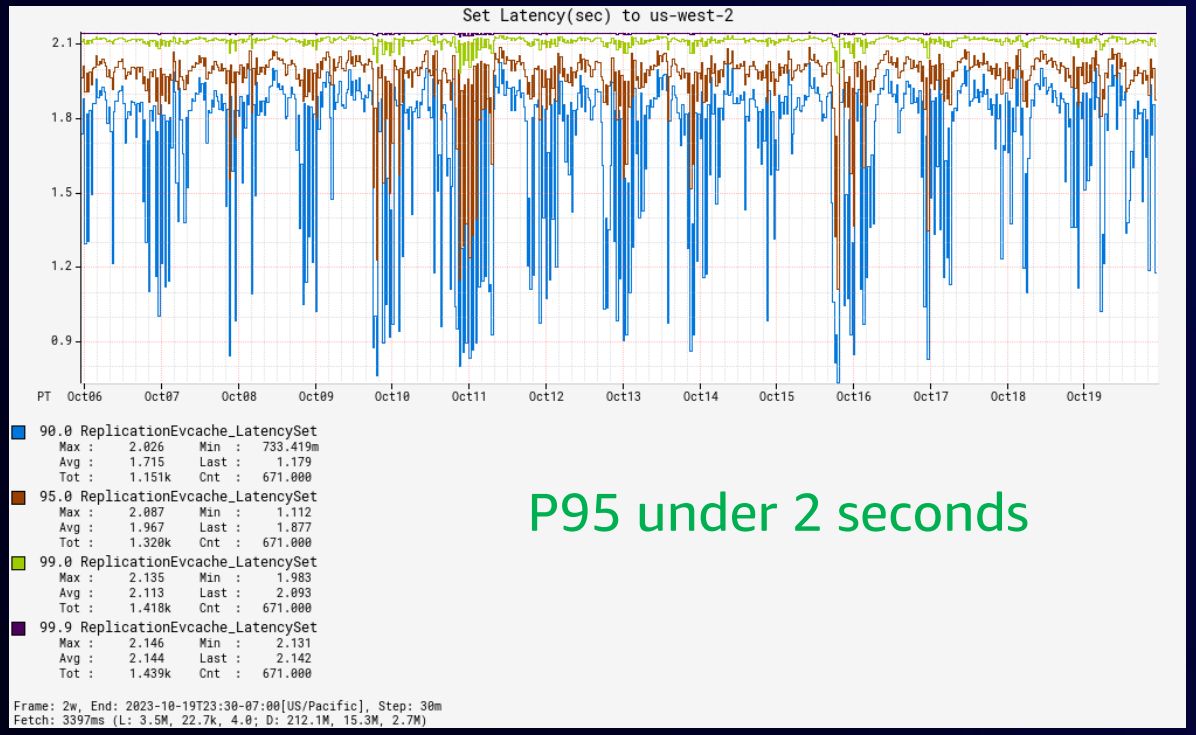


Savings of ~50% in network transfer costs

Did it work?

RPS
On
Writer Side

Even distribution of load



P95 under 2 seconds

End to End
latencies
in seconds

Agenda

Introduction

Global Replication

Design and Architecture

Observability

Efficiency Improvements

Life of an EVCache Engineer

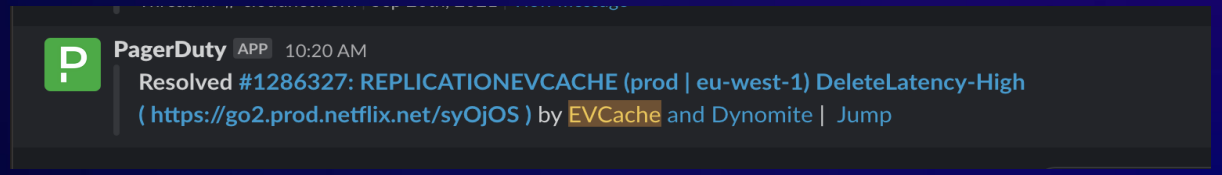
Life of an EVCache Engineer

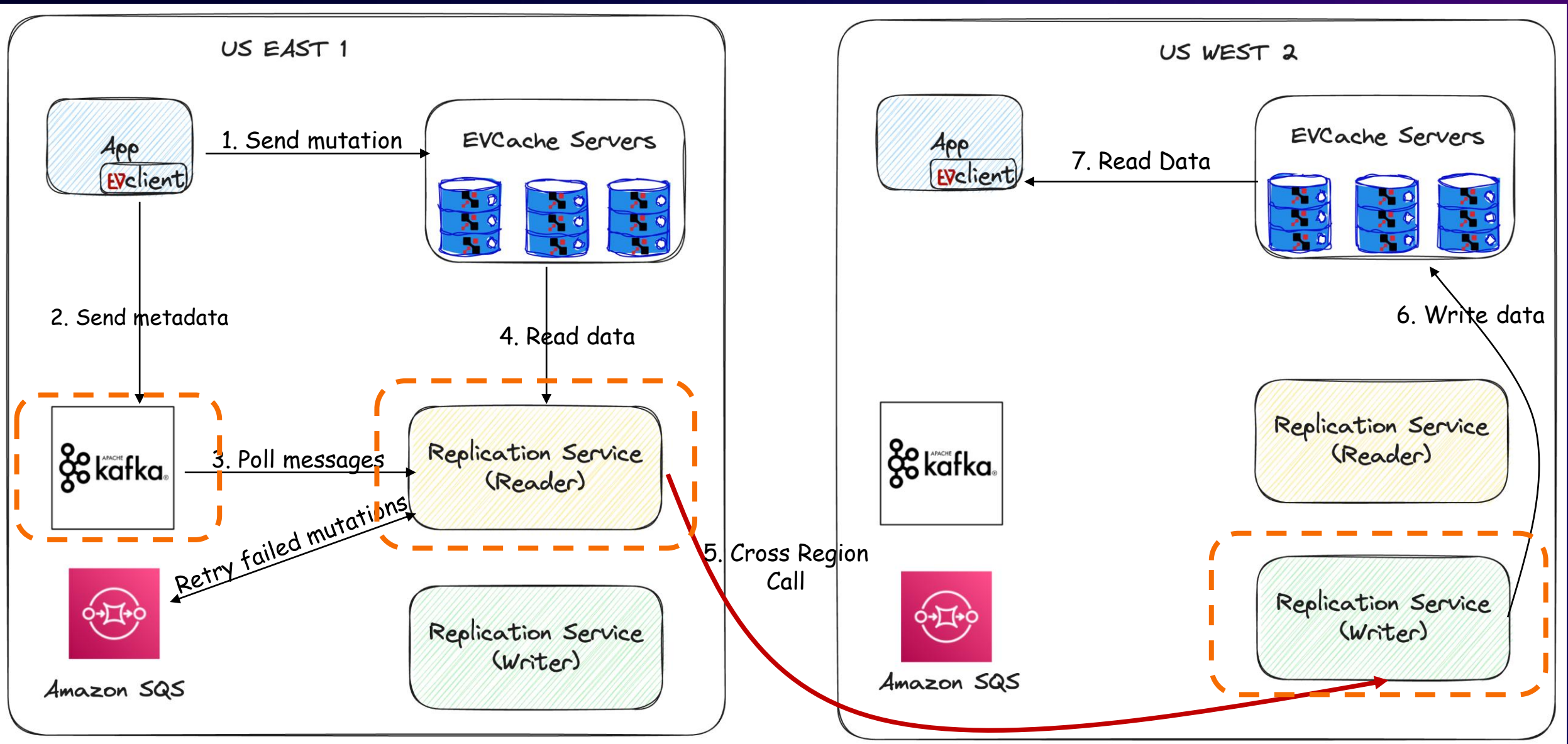
Latency Page

Instances going out of discovery

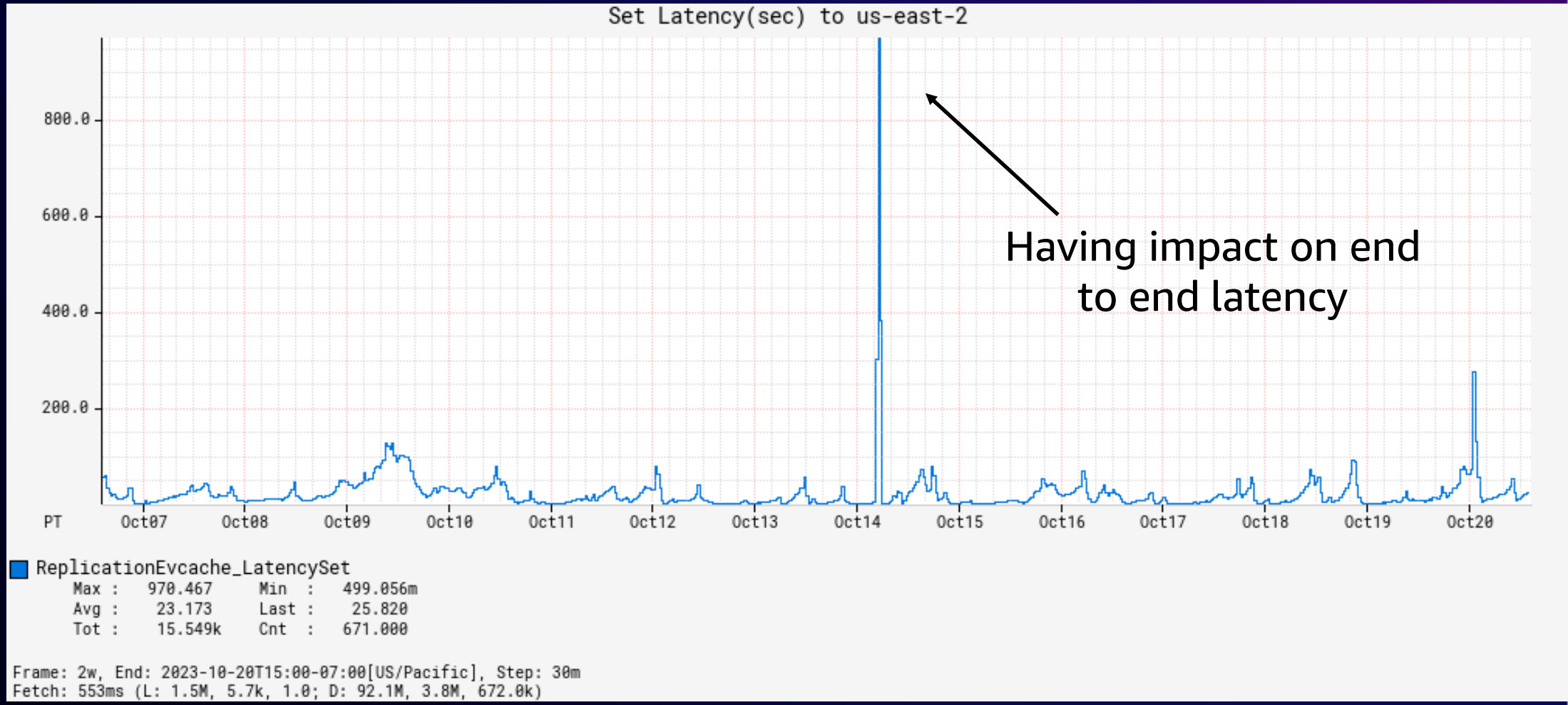
Latency Page

- Kafka
  - Consumer lag
  - # of partitions per topic
  - Join rate
- Reader
  - Bottlenecked by thread pool
  - Control message polling
- Writer
  - Slow backend

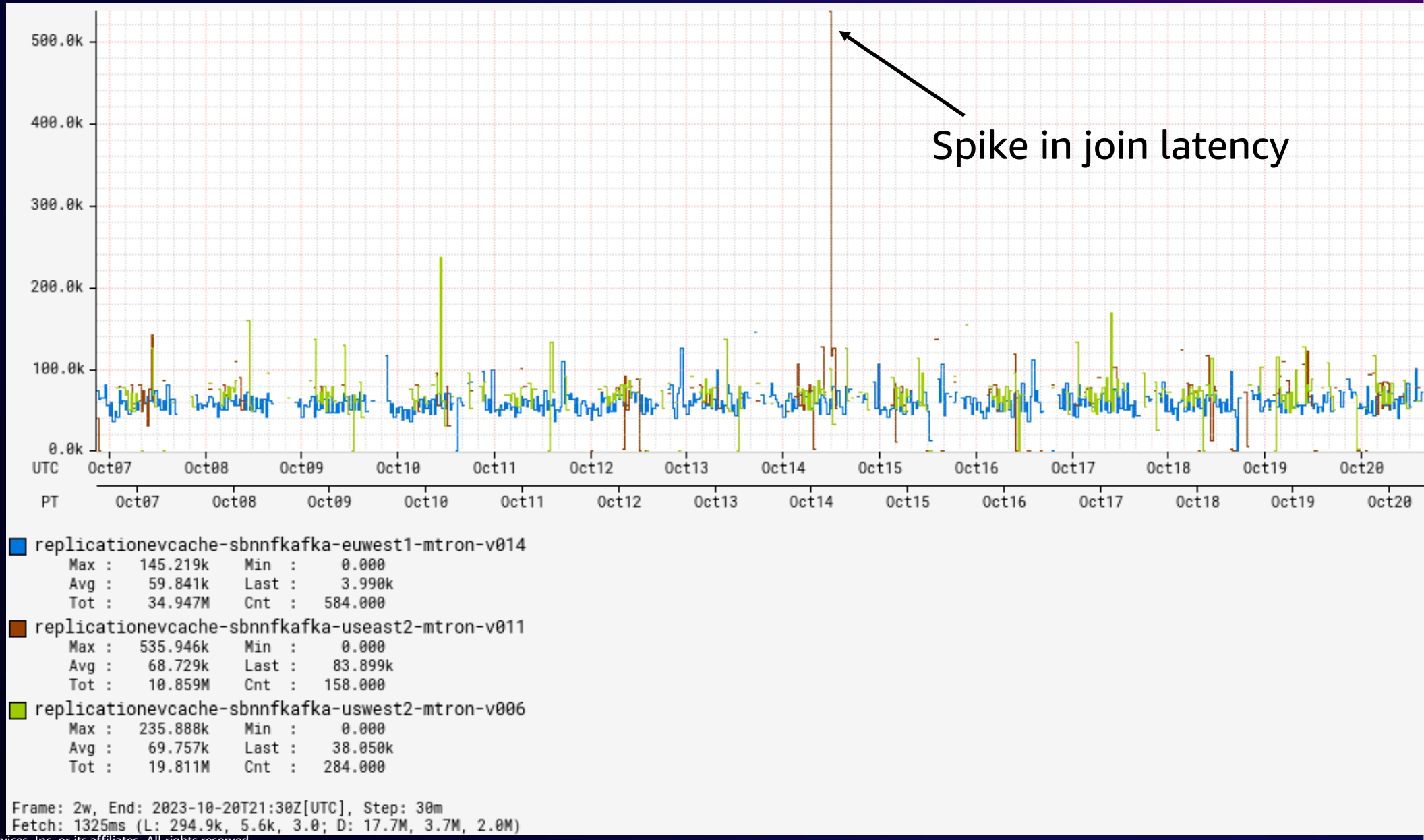




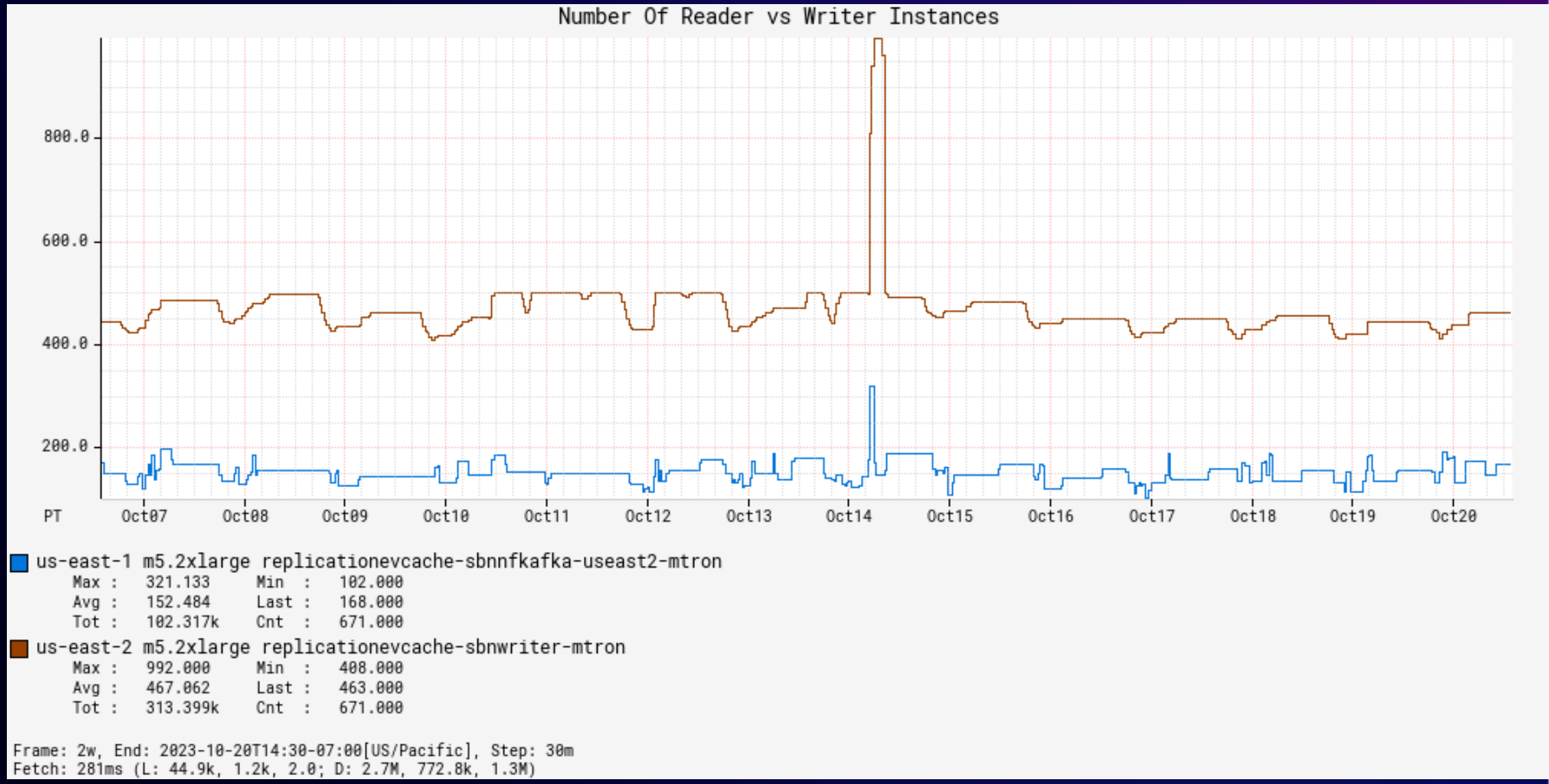
SET latency in Seconds

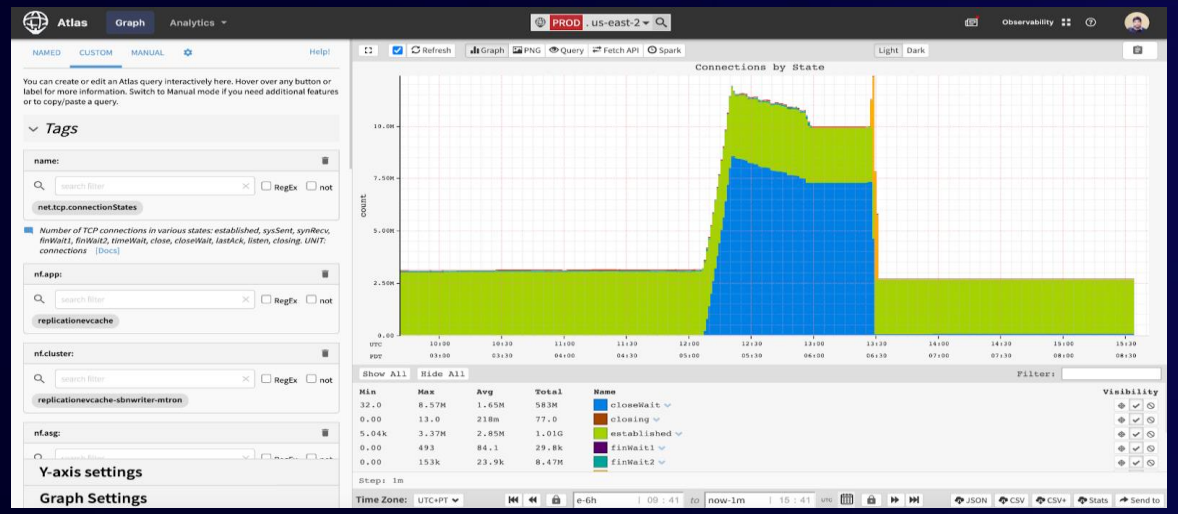


Join latency in milliseconds

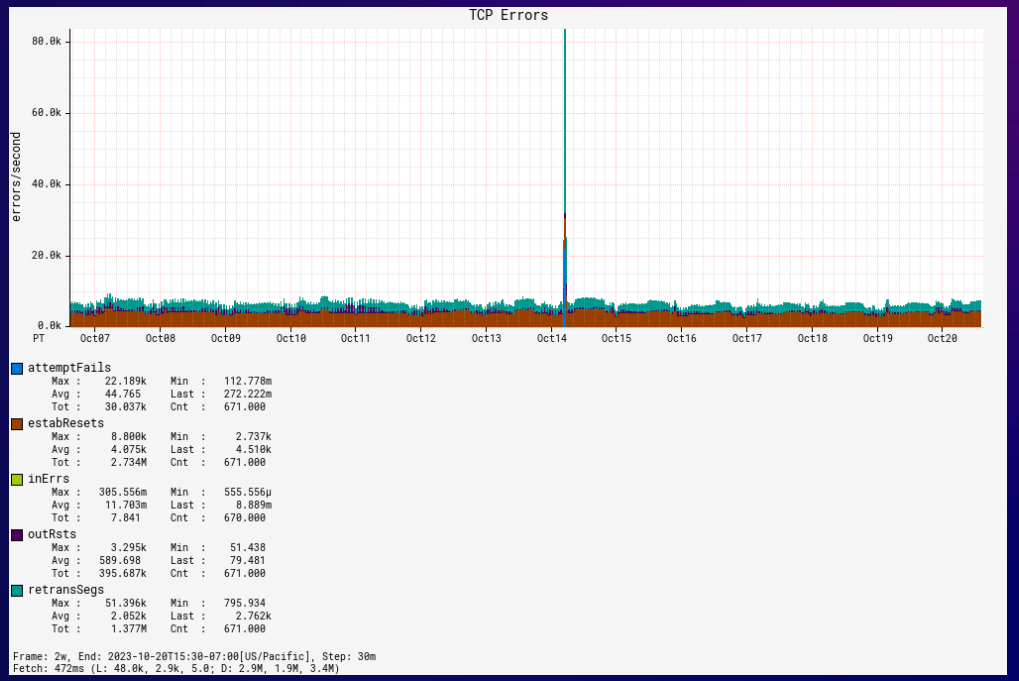


Surge in instances





TCP Errors Writer



TCP Errors Reader

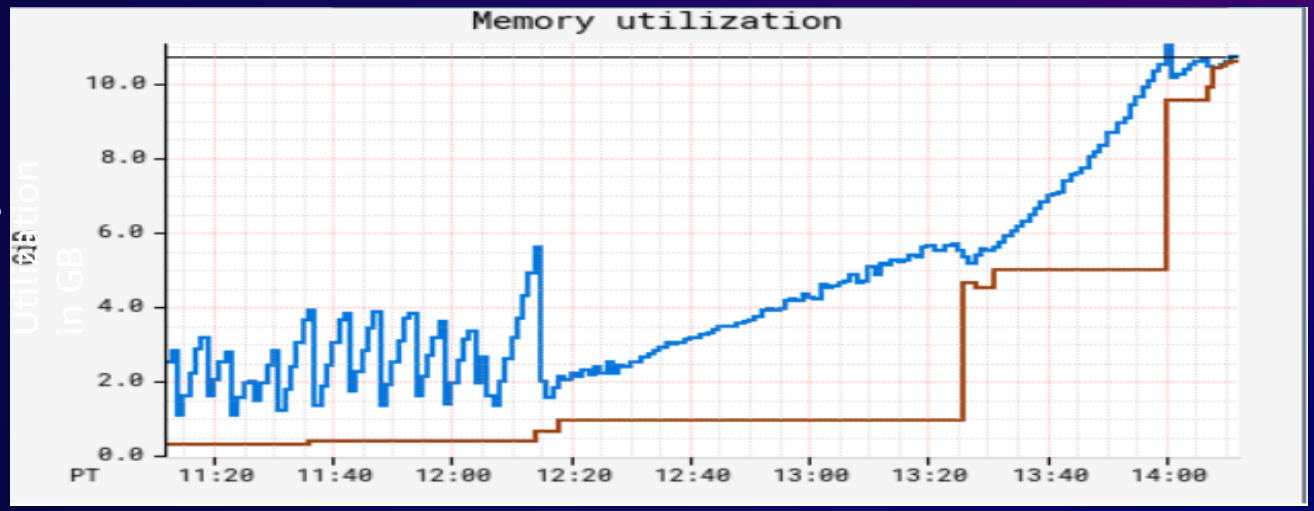
```
tcp6    6017      0 10.168.214.104:7004      52.71.231.25:13048      CLOSE_WAIT
tcp6    6016      0 10.168.214.104:7004      34.220.255.118:19838    CLOSE_WAIT
tcp6    6017      0 10.168.214.104:7004      34.222.159.118:11690    CLOSE_WAIT
tcp6    6015      0 10.168.214.104:7004      54.229.224.111:27374    CLOSE_WAIT
tcp6    6015      0 10.168.214.104:7004      52.12.163.46:51954      CLOSE_WAIT
tcp6     289      0 10.168.214.104:7004      3.253.46.21:17584       CLOSE_WAIT
tcp6    6015      0 10.168.214.104:7004      54.77.132.0:52174       CLOSE_WAIT
tcp6    6017      0 10.168.214.104:7004      3.249.35.102:33114      CLOSE_WAIT
tcp6    6016      0 10.168.214.104:7004      54.78.216.110:17370     CLOSE_WAIT
tcp6    6015      0 10.168.214.104:7004      35.165.82.30:42180      CLOSE_WAIT
tcp6    6016      0 10.168.214.104:7004      3.81.224.186:16600      CLOSE_WAIT
tcp6    6015      0 10.168.214.104:7004      35.175.244.246:26752    CLOSE_WAIT
tcp6    6016      0 10.168.214.104:7004      54.152.147.6:62618      CLOSE_WAIT
tcp6    6015      0 10.168.214.104:7004      44.210.123.30:63946     CLOSE_WAIT
tcp6    6016      0 10.168.214.104:7004      34.242.221.23:17468     CLOSE_WAIT
tcp6    6015      0 10.168.214.104:7004      54.224.241.199:38580    CLOSE_WAIT
tcp6    6016      0 10.168.214.104:7004      3.249.123.94:56310      CLOSE_WAIT
tcp6     289      0 10.168.214.104:7004      3.83.41.125:14794       CLOSE_WAIT
tcp6    6016      0 10.168.214.104:7004      35.88.109.167:19962     CLOSE_WAIT

prod replicationevcache-sbnwriter-mtron-v008 us-east-2 i-05d6b6ad38d419498
(nfsuper) ~ $ netstat -ltan | grep 7004 | wc -l
94149

prod replicationevcache-sbnwriter-mtron-v008 us-east-2 i-05d6b6ad38d419498
(nfsuper) ~ $
```

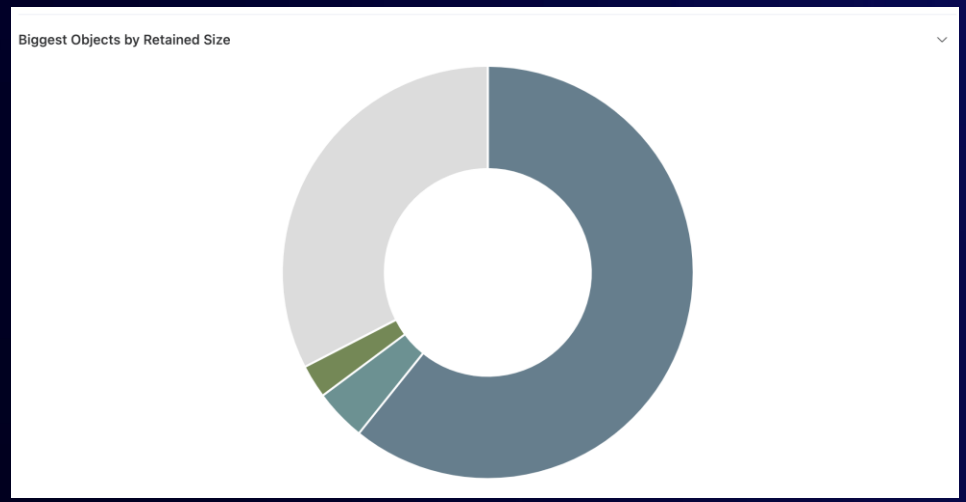
```
2023-10-14 12:19:27.679 ERROR 4659 --- [replication-369] c.n.e.r.r.RemoteHttpRequestEventListener : Connection Failed to : /18.191.27.128:7004; Protocol : null
java.net.SocketTimeoutException: Read timed out
        at java.base/sun.nio.ch.NioSocketImpl.timedRead(NioSocketImpl.java:288)
        at java.base/sun.nio.ch.NioSocketImpl.implRead(NioSocketImpl.java:314)
        at java.base/sun.nio.ch.NioSocketImpl.read(NioSocketImpl.java:355)
        at java.base/sun.nio.ch.NioSocketImpl$1.read(NioSocketImpl.java:808)
        at java.base/java.net.Socket$SocketInputStream.read(Socket.java:976)
        at java.base/sun.security.ssl.SSLSocketInputRecord.read(SSLSocketInputRecord.java:484)
        at java.base/sun.security.ssl.SSLSocketInputRecord.readHeader(SSLSocketInputRecord.java:478)
        at java.base/sun.security.ssl.SSLSocketInputRecord.decode(SSLSocketInputRecord.java:160)
        at java.base/sun.security.ssl.SSLTransport.decode(SSLTransport.java:111)
        at java.base/sun.security.ssl.SSLSocketImpl.decode(SSLSocketImpl.java:1510)
        at java.base/sun.security.ssl.SSLSocketImpl.readHandshakeRecord(SSLSocketImpl.java:1425)
        at java.base/sun.security.ssl.SSLSocketImpl.startHandshake(SSLSocketImpl.java:455)
        at java.base/sun.security.ssl.SSLSocketImpl.startHandshake(SSLSocketImpl.java:426)
        at com.netflix.metatron.ipc.security.DelegatingSSLSocket.startHandshake(DelegatingSSLSocket.java:88)
        at okhttp3.internal.connection.RealConnection.connectTls(RealConnection.kt:379)
        at okhttp3.internal.connection.RealConnection.establishProtocol(RealConnection.kt:337)
        at okhttp3.internal.connection.RealConnection.connect(RealConnection.kt:209)
        at okhttp3.internal.connection.ExchangeFinder.findConnection(ExchangeFinder.kt:226)
```

Instances going out of discovery

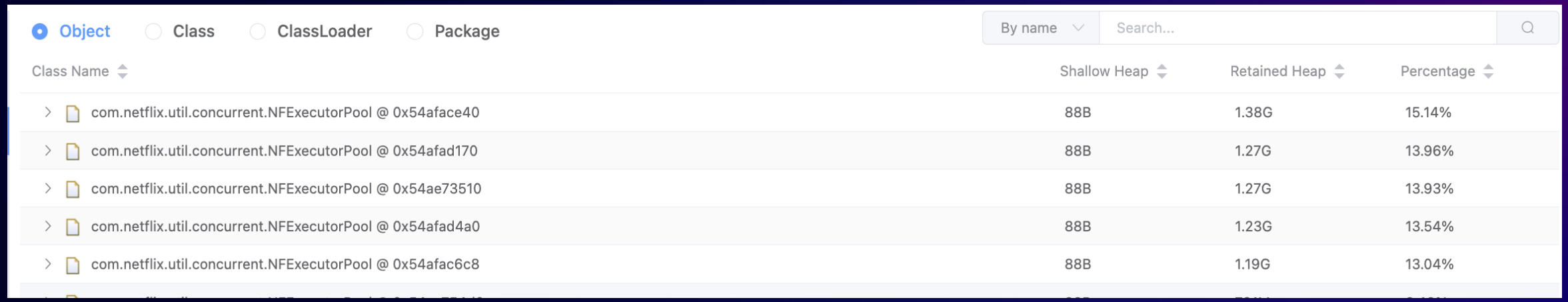
```
2023-11-01 22:41:07.268 ERROR 3989 --- [tor-scheduled-2] c.n.e.p.EVCacheClientPool                : Exception while
refreshing the server list
java.lang.OutOfMemoryError: Java heap space
2023-11-01 22:41:25.951  WARN 3989 --- [nfoReplicator-0] c.n.s.c.ArchaiusPollerHealthIndicator    : Archaius poller failed
to load remote layer for 520 cycles
2023-11-01 22:41:27.250  WARN 3989 --- [nfoReplicator-0] c.n.d.InstanceInfoReplicator             : There was a problem with
the instance info replicator
java.lang.OutOfMemoryError: Java heap space
2023-11-01 22:41:57.890  WARN 3989 --- [nfoReplicator-0] c.n.s.c.ArchaiusPollerHealthIndicator    : Archaius poller failed
to load remote layer for 521 cycles
```

Out of memory (OOM)

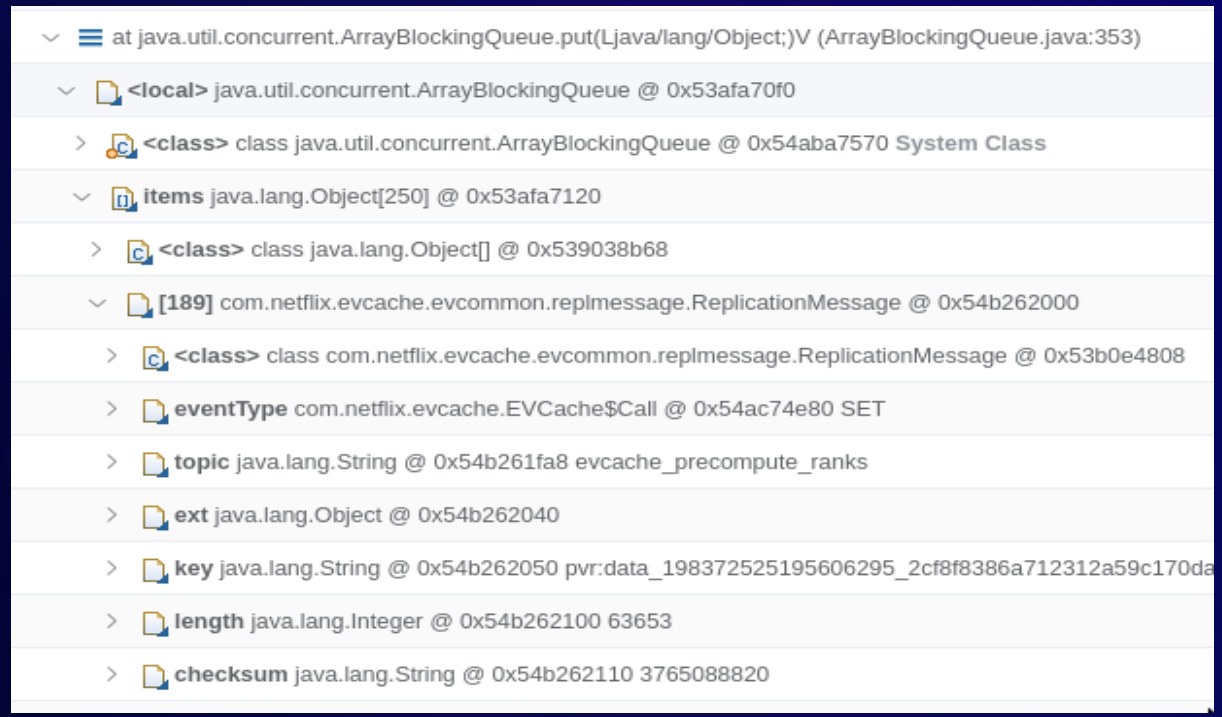


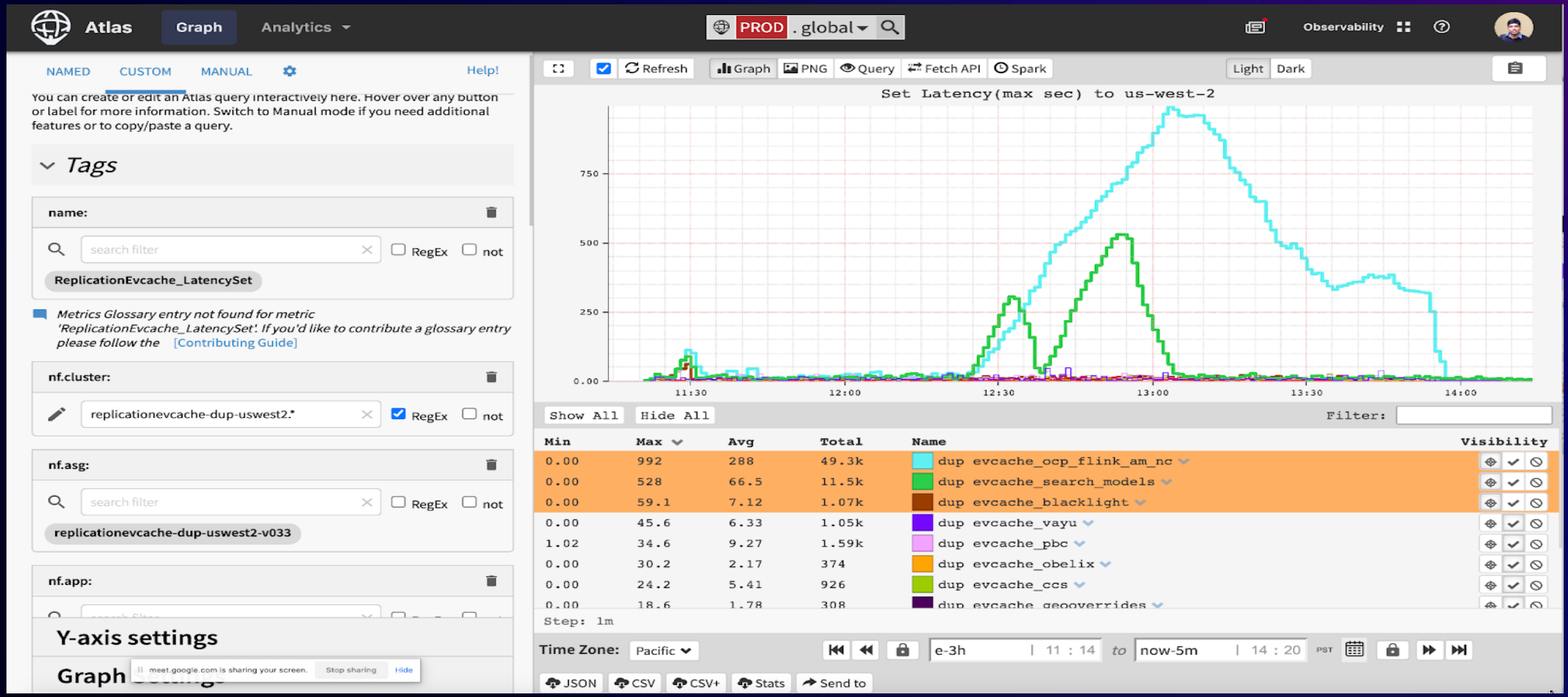


Heap Dump



700K items in some of the queues, and items are ~KBs in size, so that means ~GBs of memory taken per queue

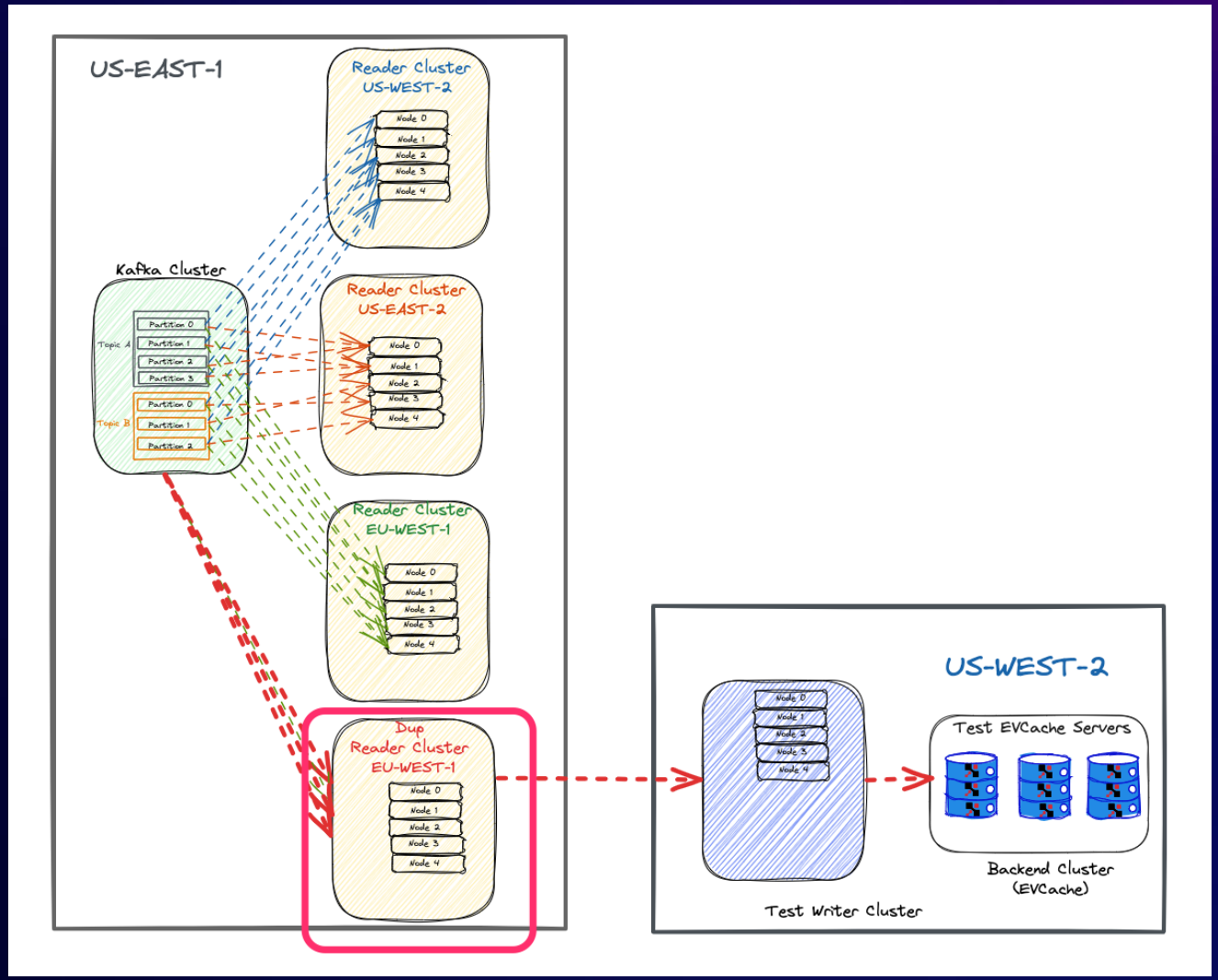




Apply backpressure

Shadow Testing

- Route X % of traffic via staging cluster
- Build confidence, faster development cycle



Agenda

Introduction

Global Replication

Design and Architecture

Observability

Efficiency Improvements

Life of an EVCache Engineer

Future

Future

IPv6

Container Migration

Generic Service

Thank you!

Please complete the session survey in the mobile app

Prudhviraj Karumanchi

https://www.linkedin.com/in/prudhviraj9/

Sriram Rangarajan

https://www.linkedin.com/in/sriram-rangarajan-35169715/

Previous Re:Invent Talk - AWS re:Invent 2021 - How Netflix operates mission-critical data stores on AWS
https://www.youtube.com/watch?v=MtGYgilaPgo