AWS re:Invent

NOV. 29 – DEC. 3, 2021 | LAS VEGAS, NV

NFX203

How Netflix operates mission-critical data stores on AWS

Tharanga Gamaethige (he/him)
Sr. Software Engineer
Netflix

Prudhviraj Karumanchi (he/him)
Sr. Software Engineer
Netflix

Data stores at Netflix

cassandra

EVCache

CockroachDB

DYNOMITE

EVCache

Agenda

Introduction

High-level architecture

Workload characterization

Observability

A day in the life of EVCache

Incidents and takeaways

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

Two variants of Memcached

- In-memory
- NVMe storage*

* https://netflixtechblog.com/application-data-caching-using-ssds-5bf25df851ef

EVCache footprint at Netflix

3 Regions
4 engineers
~160 clusters
~18,000 servers
~15,000,000 replications/sec
~350,000,000 ops/sec
~1,400,000,000,000 items
~14,000,000,000,000,000 bytes

Agenda

Introduction

High-level architecture

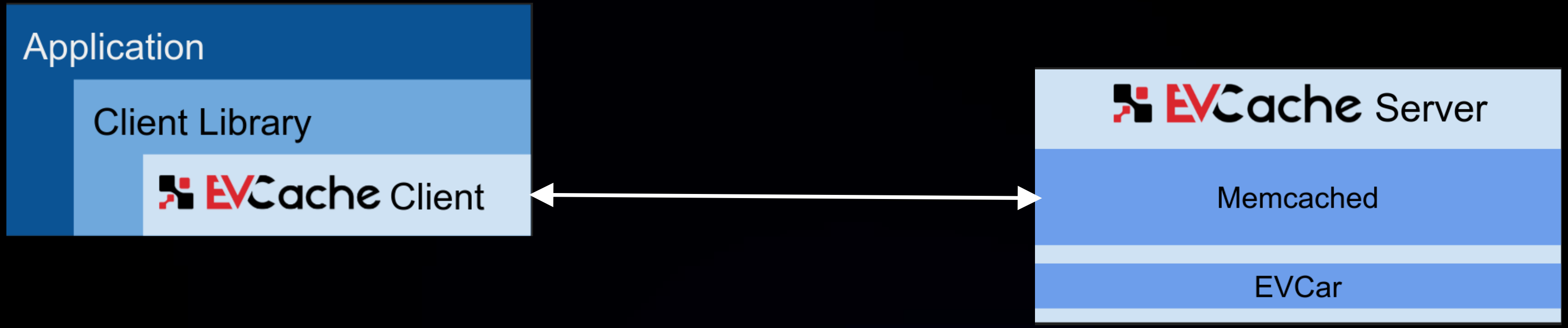
Workload characterization

Observability

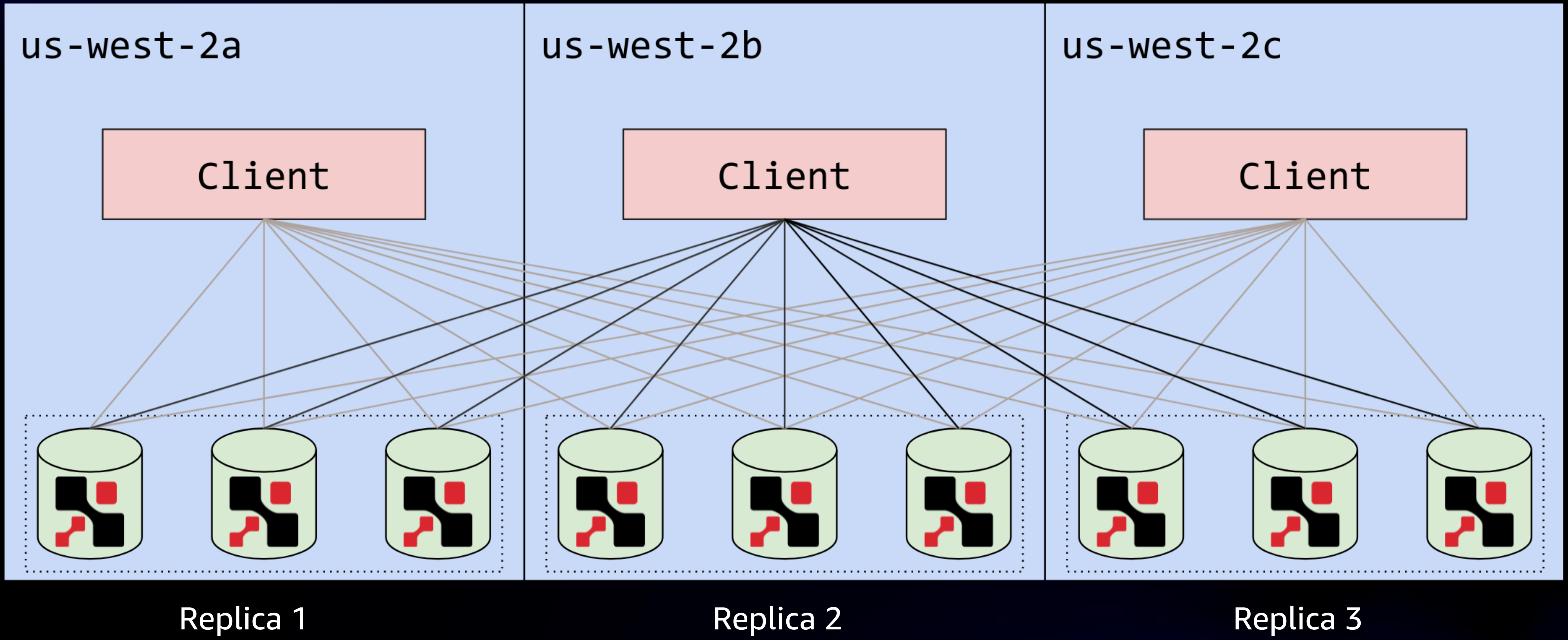
A day in the life of EVCache

Incidents and takeaways

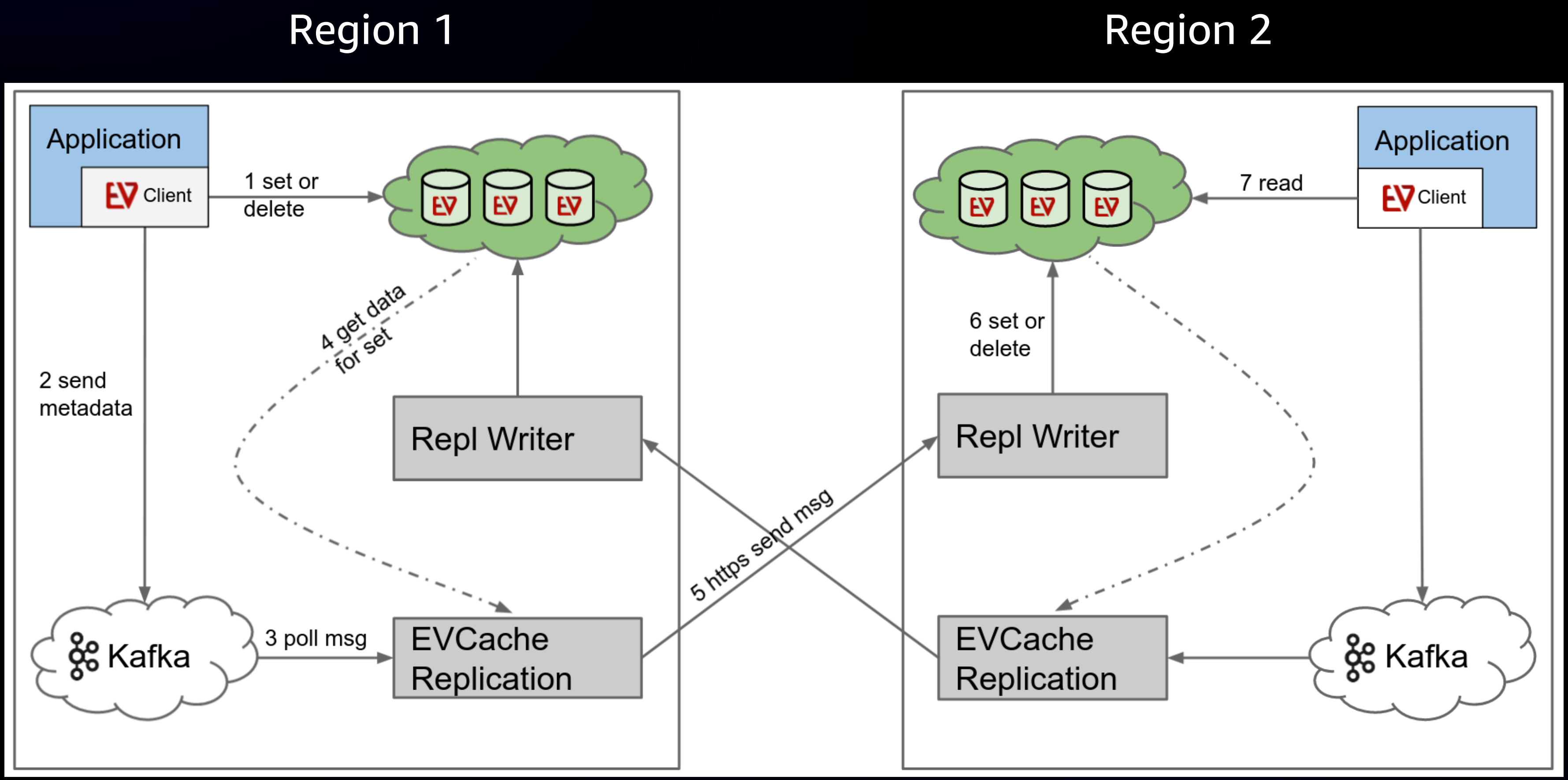
High-level architecture



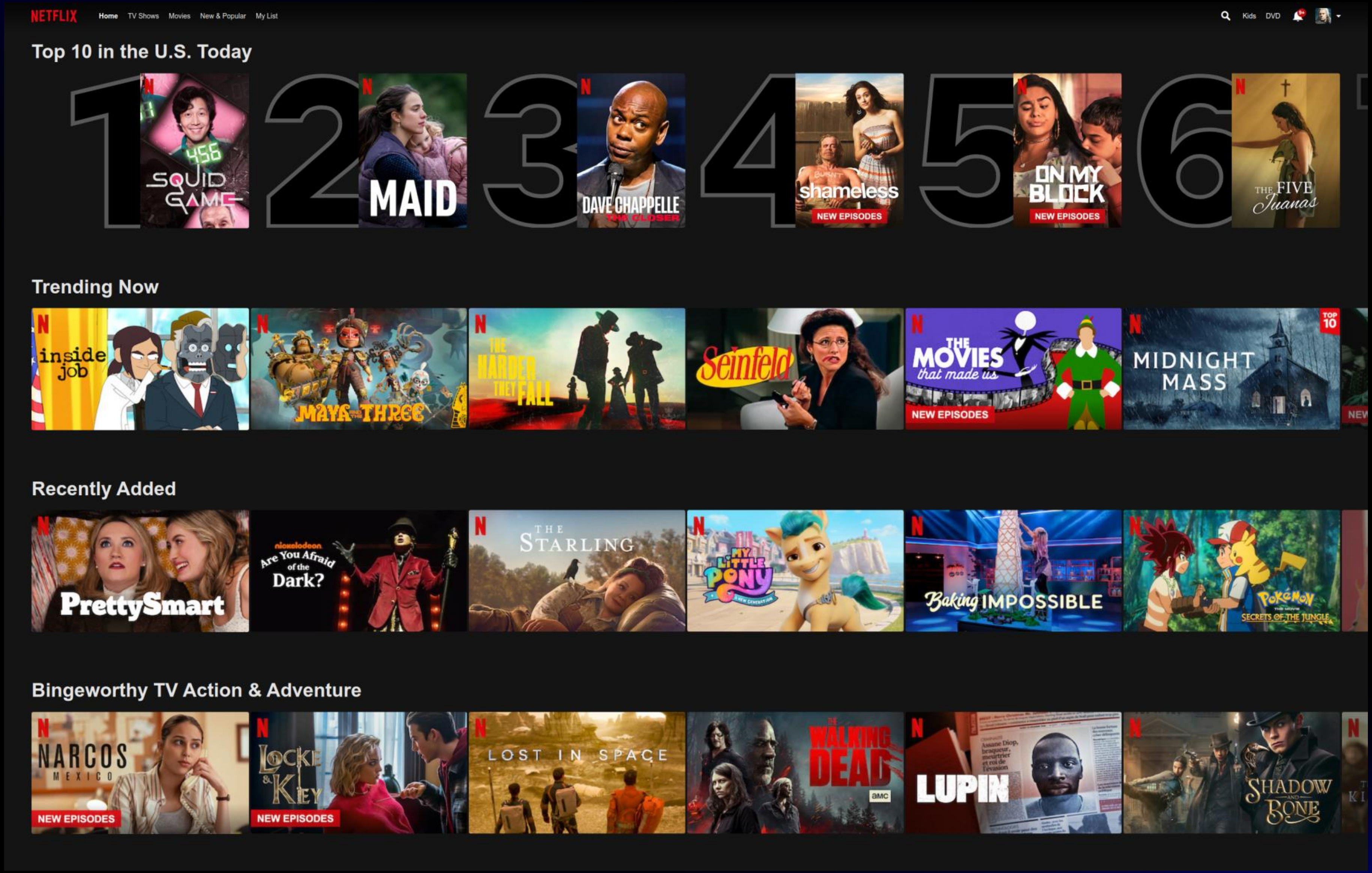
High-level architecture



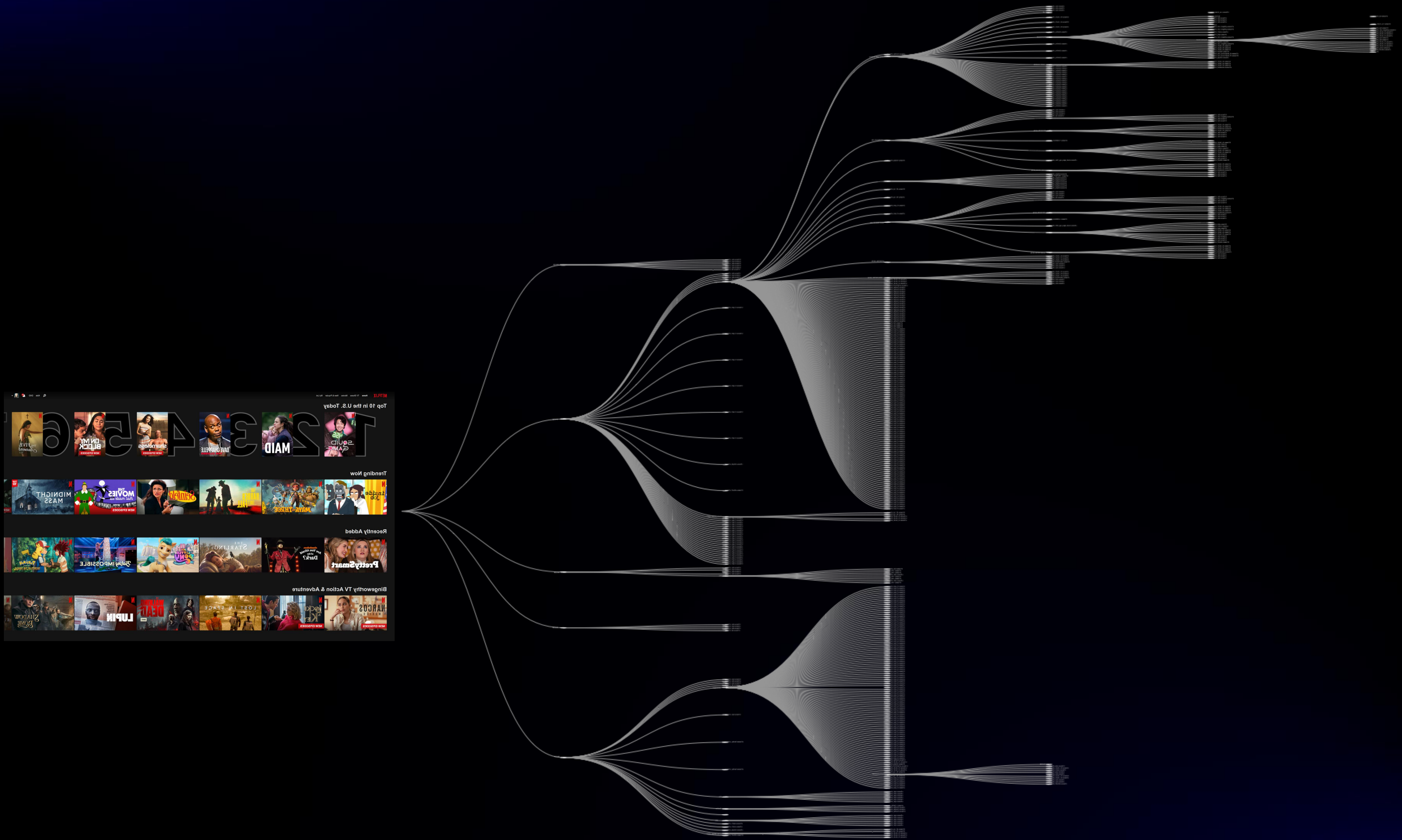
High-level architecture



Netflix home page



Request breakdown



Agenda

Introduction

High-level architecture

Workload characterization

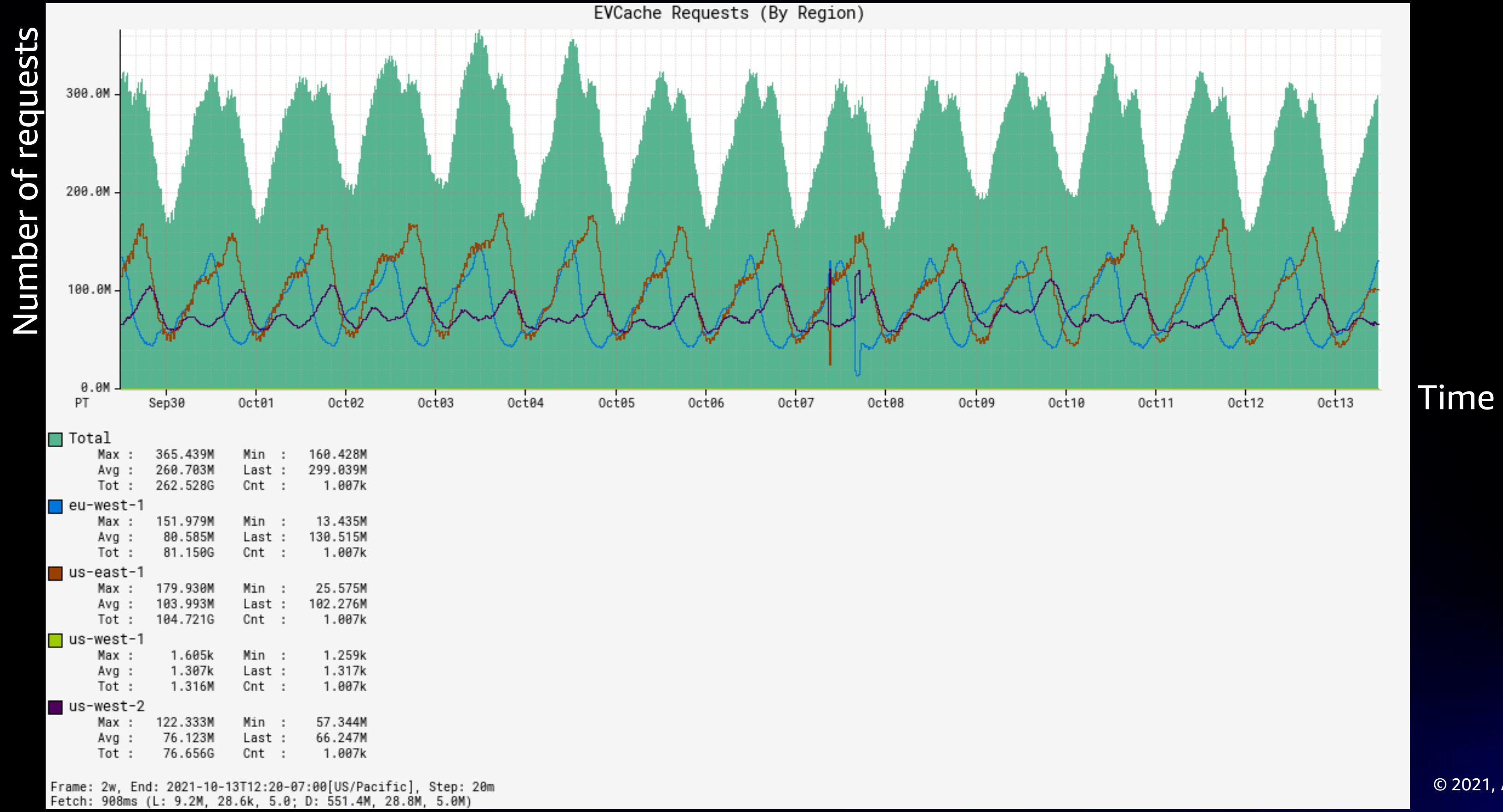
Observability

A day in the life of EVCache

Incidents and takeaways

Memcached workload characteristics at Netflix

We have 100s of clusters operating in production, out of which 45% are SSD based and the other 55% use RAM-based Memcached only

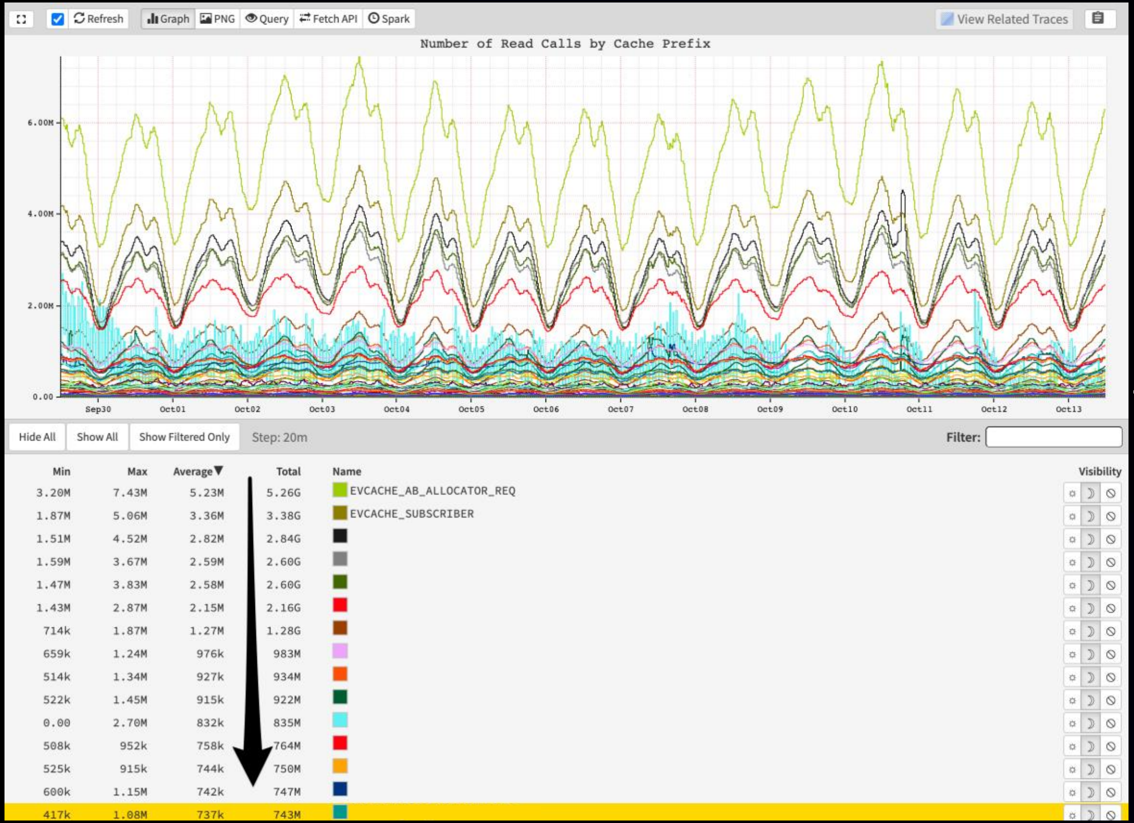


```
🟩 Total
      Max :   365.439M    Min  :   160.428M
      Avg :   260.703M    Last :   299.039M
      Tot :   262.528G    Cnt  :     1.007k
🟦 eu-west-1
      Max :   151.979M    Min  :    13.435M
      Avg :    80.585M    Last :   130.515M
      Tot :    81.150G    Cnt  :     1.007k
🟫 us-east-1
      Max :   179.930M    Min  :    25.575M
      Avg :   103.993M    Last :   102.276M
      Tot :   104.721G    Cnt  :     1.007k
🟩 us-west-1
      Max :     1.605k    Min  :     1.259k
      Avg :     1.307k    Last :     1.317k
      Tot :     1.316M    Cnt  :     1.007k
🟪 us-west-2
      Max :   122.333M    Min  :    57.344M
      Avg :    76.123M    Last :    66.247M
      Tot :    76.656G    Cnt  :     1.007k

Frame: 2w, End: 2021-10-13T12:20-07:00[US/Pacific], Step: 20m
Fetch: 908ms (L: 9.2M, 28.6k, 5.0; D: 551.4M, 28.8M, 5.0M)
```

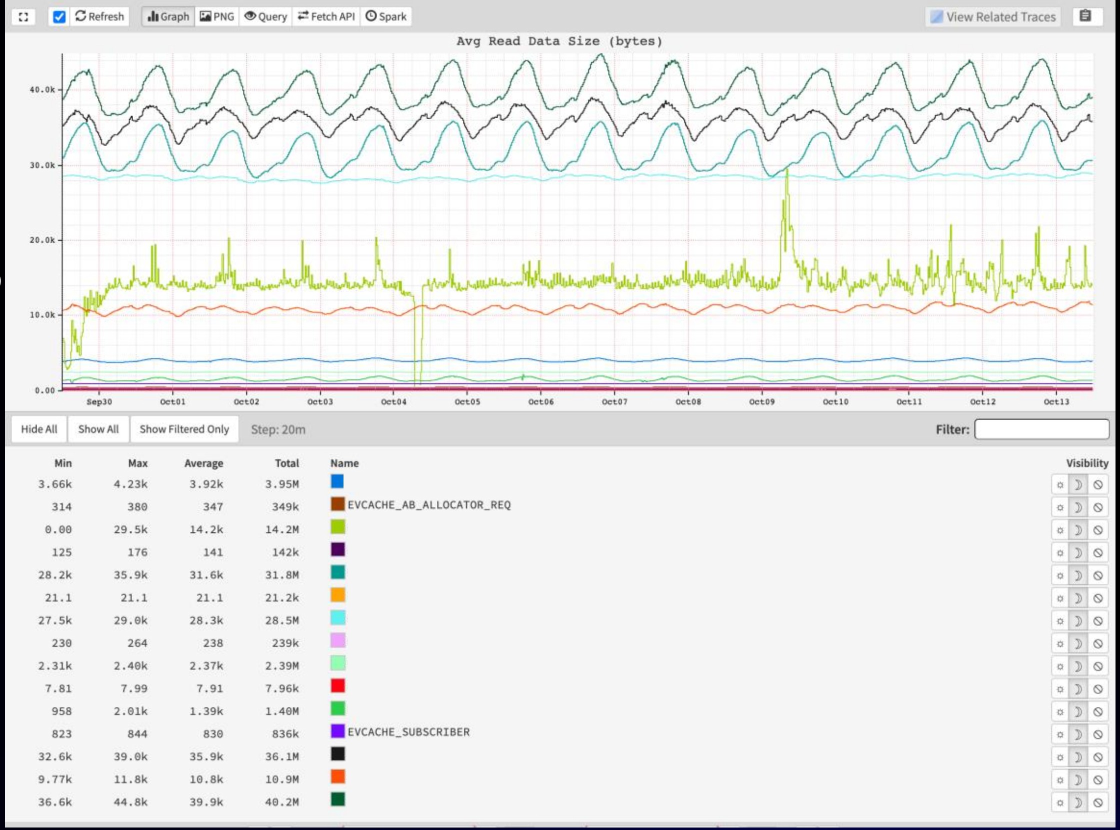
Memcached workload characteristics at Netflix

Clusters with high RPS and their average payload sizes



Number of Read Calls by Cache Prefix

Min	Max	Average▼	Total	Name
3.20M	7.43M	5.23M	5.26G	🟩 EVCACHE_AB_ALLOCATOR_REQ
1.87M	5.06M	3.36M	3.38G	🟫 EVCACHE_SUBSCRIBER
1.51M	4.52M	2.82M	2.84G	⬛
1.59M	3.67M	2.59M	2.60G	⬜
1.47M	3.83M	2.58M	2.60G	🟩
1.43M	2.87M	2.15M	2.16G	🟥
714k	1.87M	1.27M	1.28G	🟫
659k	1.24M	976k	983M	🟪
514k	1.34M	927k	934M	🟧
522k	1.45M	915k	922M	🟩
0.00	2.70M	832k	835M	🟦
508k	952k	758k	764M	🟥
525k	915k	744k	750M	🟧
600k	1.15M	742k	747M	🟦
417k	1.08M	737k	743M	🟦



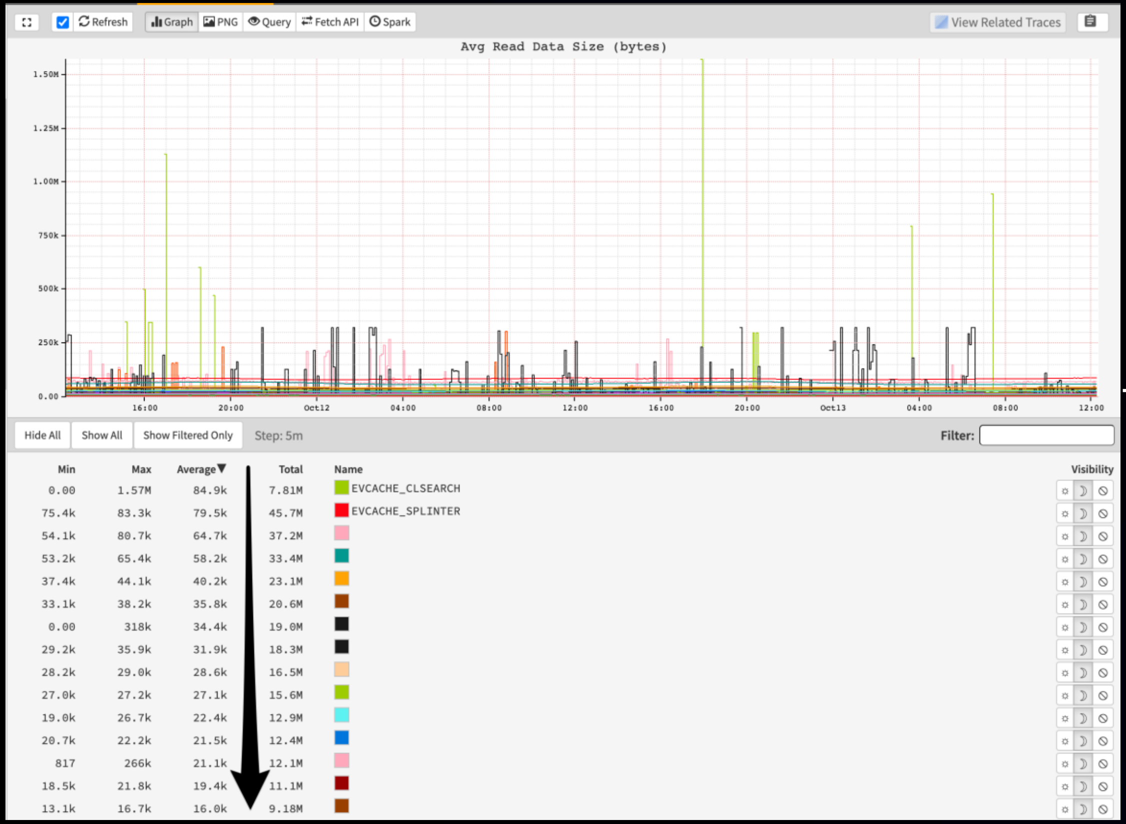
Avg Read Data Size (bytes)

Min	Max	Average	Total	Name
3.66k	4.23k	3.92k	3.95M	🟦
314	380	347	349k	🟫 EVCACHE_AB_ALLOCATOR_REQ
0.00	29.5k	14.2k	14.2M	🟩
125	176	141	142k	🟪
28.2k	35.9k	31.6k	31.8M	🟦
21.1	21.1	21.1	21.2k	🟧
27.5k	29.0k	28.3k	28.5M	🟦
230	264	238	239k	🟪
2.31k	2.40k	2.37k	2.39M	🟩
7.81	7.99	7.91	7.96k	🟥
958	2.01k	1.39k	1.40M	🟩
823	844	830	836k	🟪 EVCACHE_SUBSCRIBER
32.6k	39.0k	35.9k	36.1M	⬛
9.77k	11.8k	10.8k	10.9M	🟧
36.6k	44.8k	39.9k	40.2M	🟩

Memcached workload characteristics at Netflix

Clusters with large payload sizes and their RPS

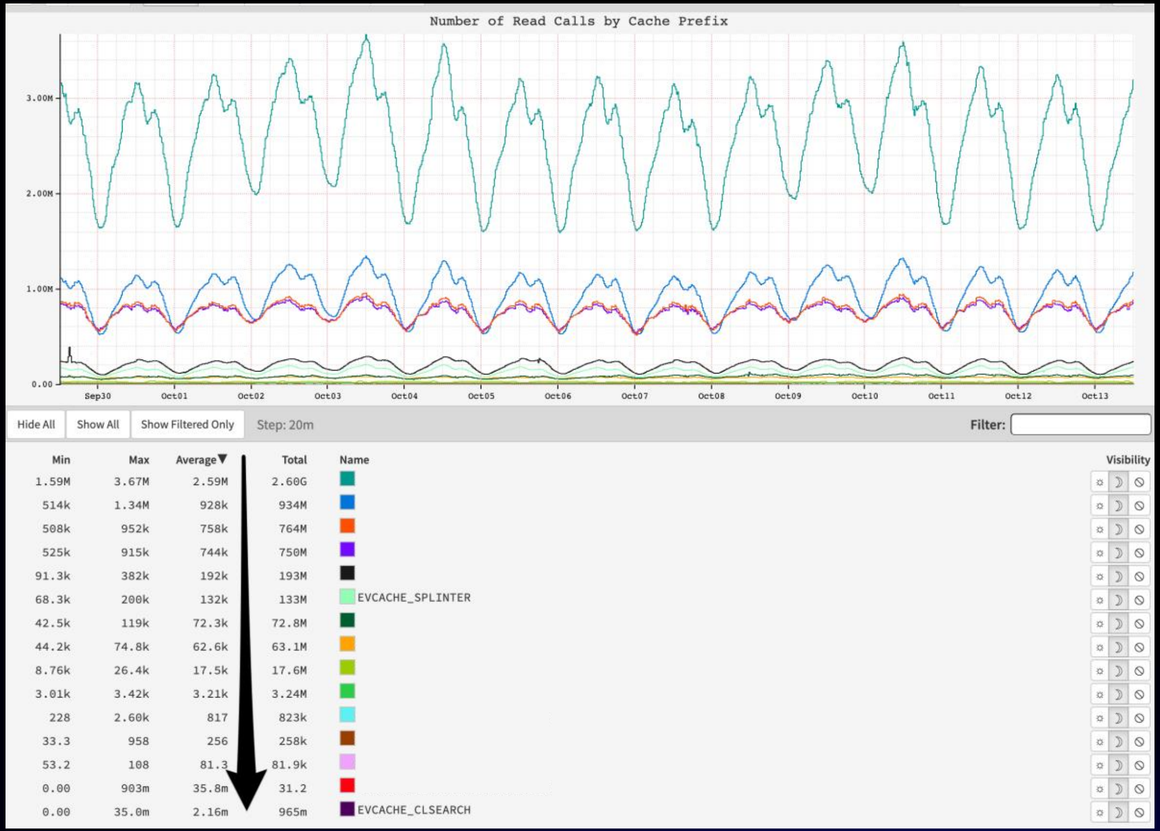
Avg read size (KB)



Min	Max	Average▼	Total	Name	Visibility
0.00	1.57M	84.9k	7.81M	■ EVCACHE_CLSEARCH	
75.4k	83.3k	79.5k	45.7M	■ EVCACHE_SPLINTER	
54.1k	80.7k	64.7k	37.2M	■	
53.2k	65.4k	58.2k	33.4M	■	
37.4k	44.1k	40.2k	23.1M	■	
33.1k	38.2k	35.8k	20.6M	■	
0.00	318k	34.4k	19.0M	■	
29.2k	35.9k	31.9k	18.3M	■	
28.2k	29.0k	28.6k	16.5M	■	
27.0k	27.2k	27.1k	15.6M	■	
19.0k	26.7k	22.4k	12.9M	■	
20.7k	22.2k	21.5k	12.4M	■	
817	266k	21.1k	12.1M	■	
18.5k	21.8k	19.4k	11.1M	■	
13.1k	16.7k	16.0k	9.18M	■	

Time

Number of read calls



Min	Max	Average▼	Total	Name	Visibility
1.59M	3.67M	2.59M	2.60G	■	
514k	1.34M	928k	934M	■	
508k	952k	758k	764M	■	
525k	915k	744k	750M	■	
91.3k	382k	192k	193M	■	
68.3k	200k	132k	133M	■ EVCACHE_SPLINTER	
42.5k	119k	72.3k	72.8M	■	
44.2k	74.8k	62.6k	63.1M	■	
8.76k	26.4k	17.5k	17.6M	■	
3.01k	3.42k	3.21k	3.24M	■	
228	2.60k	817	823k	■	
33.3	958	256	258k	■	
53.2	108	81.3	81.9k	■	
0.00	903m	35.8m	31.2	■	
0.00	35.0m	2.16m	965m	■ EVCACHE_CLSEARCH	

Time

Agenda

Introduction

High-level architecture

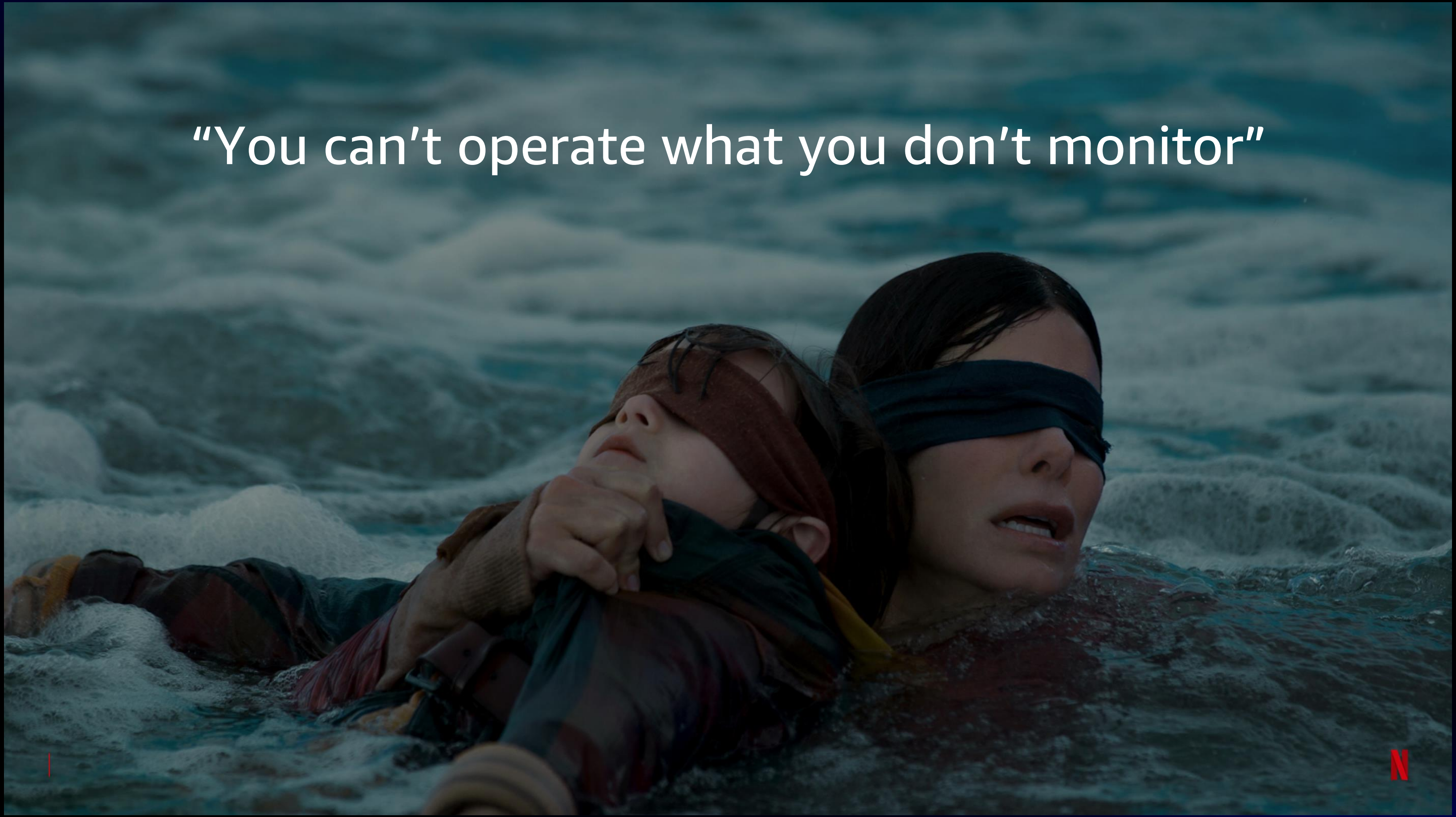
Workload characterization

Observability

A day in the life of EVCache

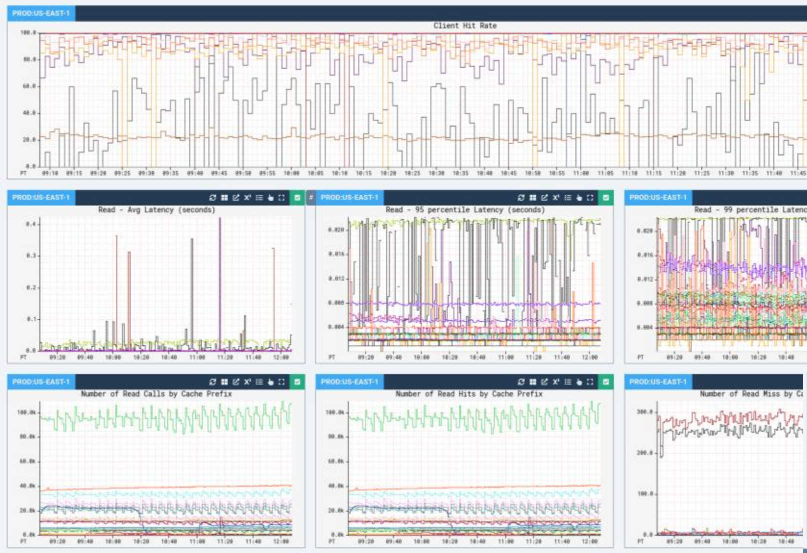
Incidents and takeaways

Observability

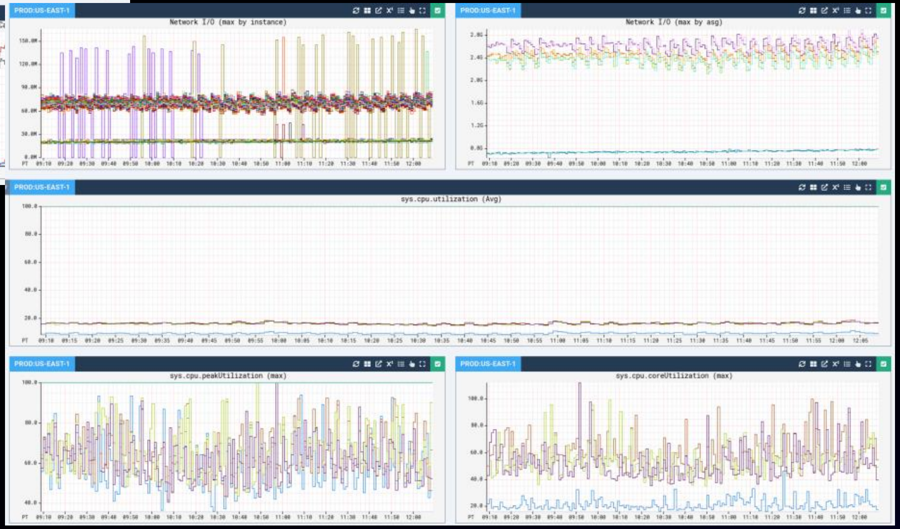


“You can’t operate what you don’t monitor”

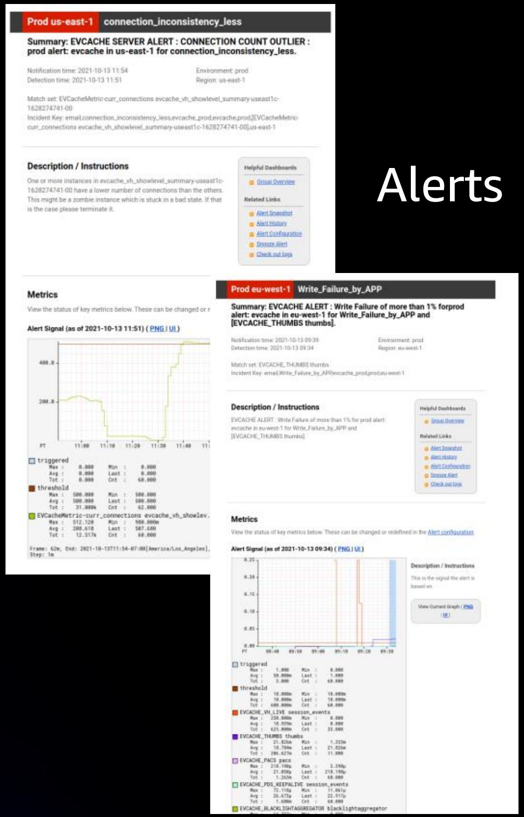
Observability


PROD:US-EAST-1
Client Hit Rate
Read - Avg Latency (seconds)
Read - 95 percentile Latency (seconds)
Read - 99 percentile Latency (seconds)
Number of Read Calls by Cache Prefix
Number of Read Hits by Cache Prefix
Number of Read Miss by Ca

Client dashboard


PROD:US-EAST-1
Network I/O (max by instance)
Network I/O (max by asg)
sys.cpu.utilization (Avg)
sys.cpu.peakUtilization (max)
sys.cpu.coreUtilization (max)

Server dashboard



Prod us-east-1 connection_inconsistency_less

Summary: EVCACHE SERVER ALERT : CONNECTION COUNT OUTLIER : prod alert: evcache in us-east-1 for connection_inconsistency_less.

Notification time: 2021-10-13 11:54
Detection time: 2021-10-13 11:51

Environment: prod
Region: us-east-1

Match set: EVCacheMetric-curr_connections evcache_vh_showlevel_summary-useast1c-1628274741-00
Incident Key: email,connection_inconsistency_less,evcache_prod,evcache,prod,[EVCacheMetric-curr_connections evcache_vh_showlevel_summary-useast1c-1628274741-00],us-east-1

Description / Instructions

One or more instances in evcache_vh_showlevel_summary-useast1c-1628274741-00 have a lower number of connections than the others. This might be a zombie instance which is stuck in a bad state. If that is the case please terminate it.

Helpful Dashboards
- Group Overview

Related Links
- Alert Snapshot
- Alert History
- Alert Configuration
- Snooze Alert
- Check out logs

Metrics

View the status of key metrics below. These can be changed or r

Alert Signal (as of 2021-10-13 11:51) (PNG | UI)

Prod eu-west-1 Write_Failure_by_APP

Summary: EVCACHE ALERT : Write Failure of more than 1% forprod alert: evcache in eu-west-1 for Write_Failure_by_APP and [EVCACHE_THUMBS thumbs].

Notification time: 2021-10-13 09:39
Detection time: 2021-10-13 09:34

Environment: prod
Region: eu-west-1

Match set: EVCACHE_THUMBS thumbs
Incident Key: email,Write_Failure_by_APP,evcache_prod,prod,eu-west-1

Description / Instructions

EVCACHE ALERT : Write Failure of more than 1% for prod alert: evcache in eu-west-1 for Write_Failure_by_APP and [EVCACHE_THUMBS thumbs].

Helpful Dashboards
- Group Overview

Related Links
- Alert Snapshot
- Alert History
- Alert Configuration
- Snooze Alert
- Check out logs

Metrics

View the status of key metrics below. These can be changed or redefined in the Alert configuration.

Alert Signal (as of 2021-10-13 09:34) (PNG | UI)

Description / Instructions

This is the signal the alert is based on.

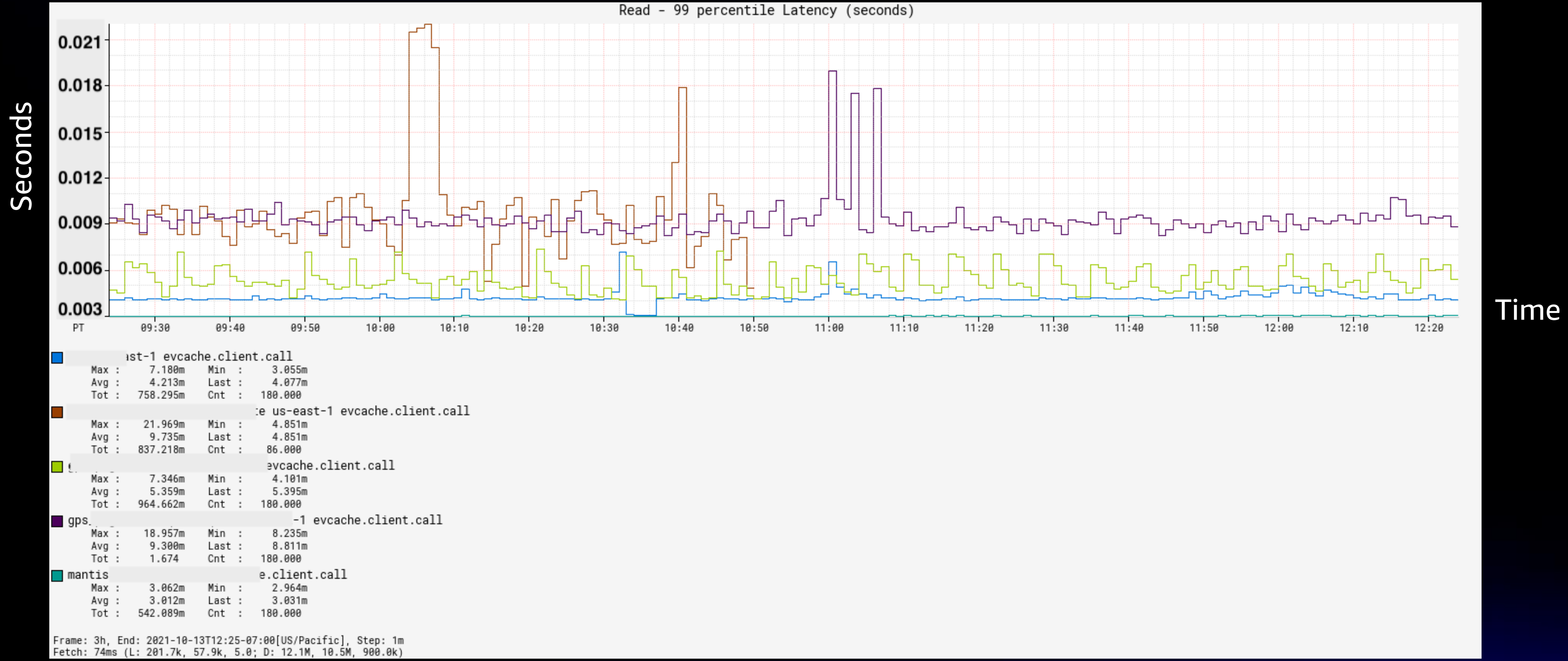
View Current Graph (PNG | UI)

triggered
threshold
EVCacheMetric-curr_connections evcache_vh_showlev.
EVCACHE_VH_LIVE session_events
EVCACHE_THUMBS thumbs
EVCACHE_PACS pacs
EVCACHE_PDS_KEEPALIVE session_events
EVCACHE_BLACKLIGHTAGGREGATOR blacklightaggregator

Alerts

Observability

Read latency of clients

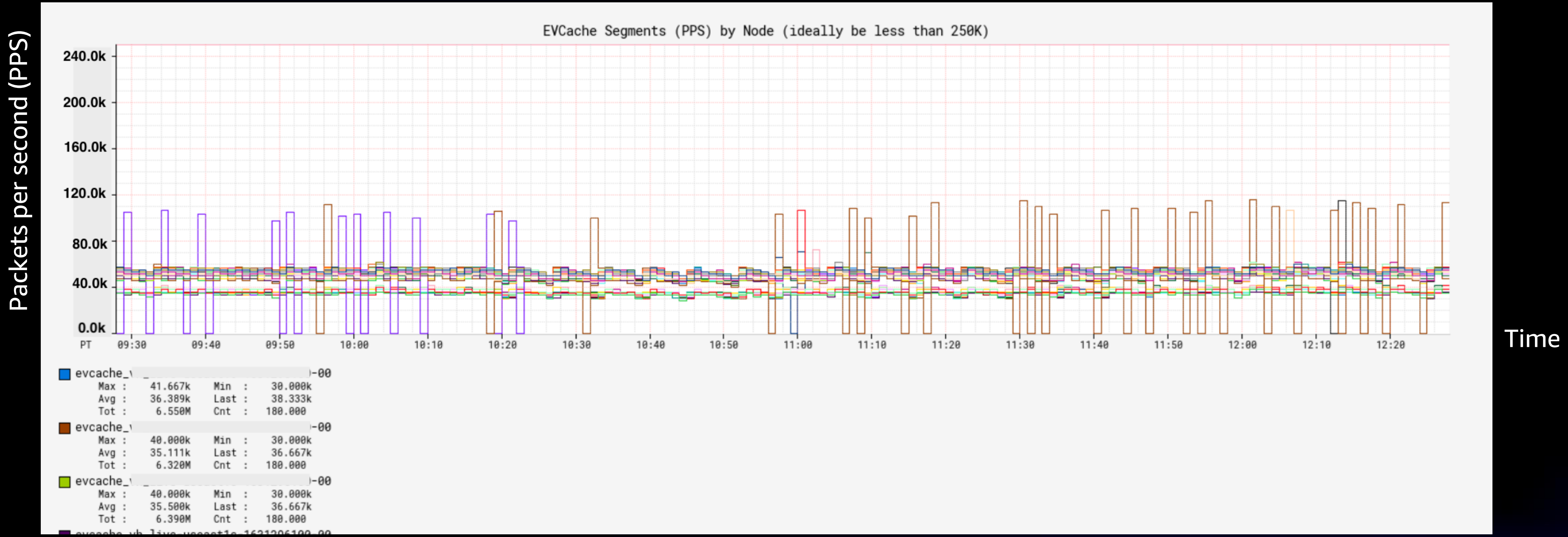


```
🟦 [redacted]ast-1 evcache.client.call
      Max :     7.180m   Min  :     3.055m
      Avg :     4.213m   Last :     4.077m
      Tot :   758.295m   Cnt  :   180.000
🟫 [redacted]te us-east-1 evcache.client.call
      Max :    21.969m   Min  :     4.851m
      Avg :     9.735m   Last :     4.851m
      Tot :   837.218m   Cnt  :    86.000
🟩 [redacted]evcache.client.call
      Max :     7.346m   Min  :     4.101m
      Avg :     5.359m   Last :     5.395m
      Tot :   964.662m   Cnt  :   180.000
🟪 gps[redacted]-1 evcache.client.call
      Max :    18.957m   Min  :     8.235m
      Avg :     9.300m   Last :     8.811m
      Tot :     1.674    Cnt  :   180.000
🟩 mantis[redacted]e.client.call
      Max :     3.062m   Min  :     2.964m
      Avg :     3.012m   Last :     3.031m
      Tot :   542.089m   Cnt  :   180.000

Frame: 3h, End: 2021-10-13T12:25-07:00[US/Pacific], Step: 1m
Fetch: 74ms (L: 201.7k, 57.9k, 5.0; D: 12.1M, 10.5M, 900.0k)
```

Observability

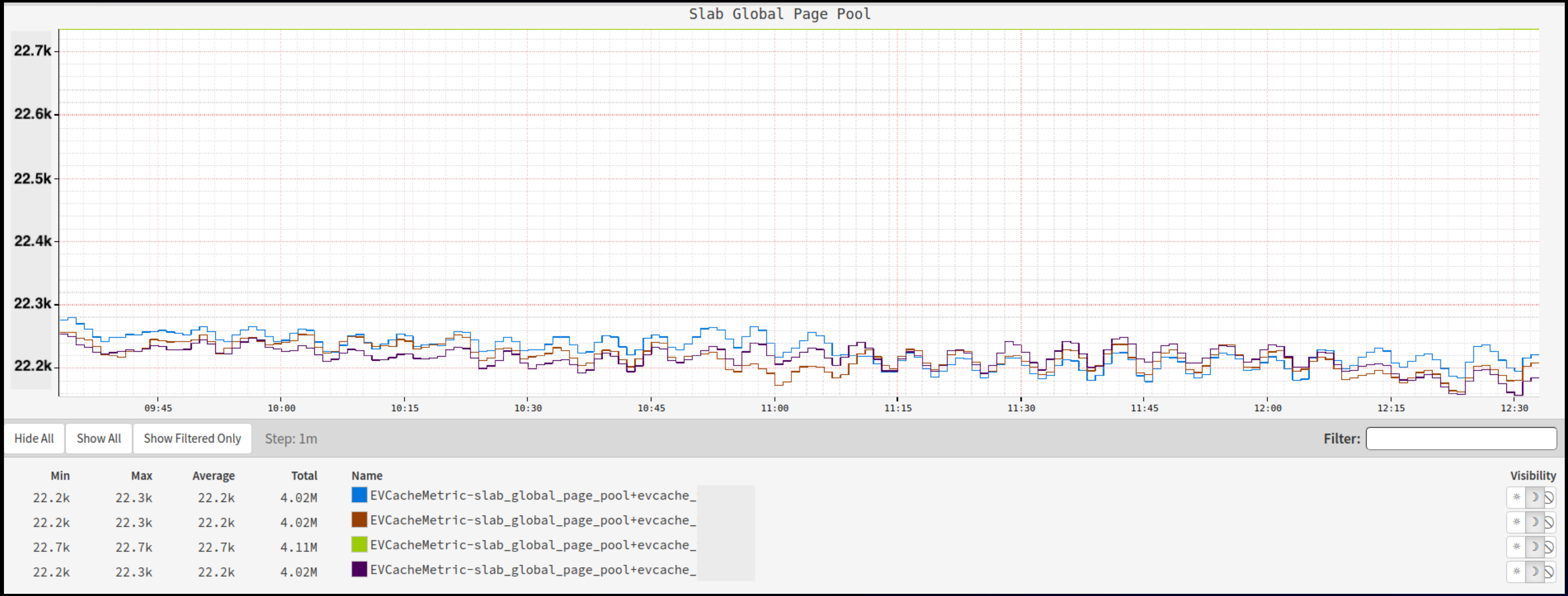
Server network usage



```
■ evcache_[redacted]-00
     Max :    41.667k    Min  :    30.000k
     Avg :    36.389k    Last :    38.333k
     Tot :     6.550M    Cnt  :   180.000
■ evcache_[redacted]-00
     Max :    40.000k    Min  :    30.000k
     Avg :    35.111k    Last :    36.667k
     Tot :     6.320M    Cnt  :   180.000
■ evcache_[redacted]-00
     Max :    40.000k    Min  :    30.000k
     Avg :    35.500k    Last :    36.667k
     Tot :     6.390M    Cnt  :   180.000
```

Observability

Memcached page pools



Min	Max	Average	Total	Name	Visibility
22.2k	22.3k	22.2k	4.02M	🟦 EVCacheMetric-slab_global_page_pool+evcache_	
22.2k	22.3k	22.2k	4.02M	🟫 EVCacheMetric-slab_global_page_pool+evcache_	
22.7k	22.7k	22.7k	4.11M	🟩 EVCacheMetric-slab_global_page_pool+evcache_	
22.2k	22.3k	22.2k	4.02M	🟪 EVCacheMetric-slab_global_page_pool+evcache_	

Number of pages in the global pool

Time

How do we provision the caching servers

3 dimensions to monitor

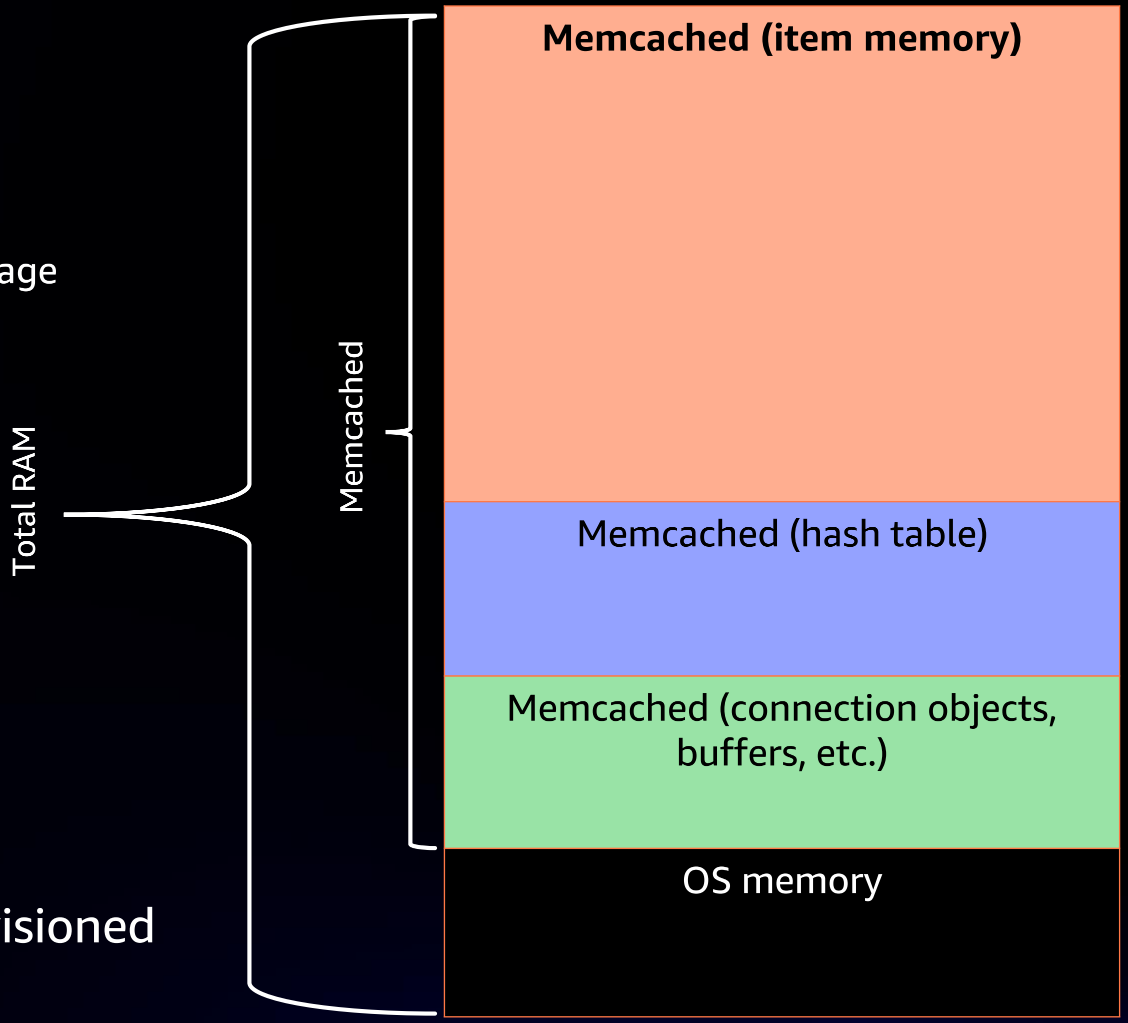
- Compute
- Network
- Memory

How do we provision the caching servers

Let's pick an example and see how we provision memory for Memcached

```
/usr/bin/memcached  -m 27045 -o hashpower=27 <...>
```

- The above instance can hold up to 2^{27} items = 128 million items with average item size of 220 bytes
- Total memory used by Memcached
- Overcommit ratio is disabled



General rule of the thumb = aim for 90% memory provisioned for Memcached

Agenda

Introduction

High-level architecture

Workload characterization

Observability

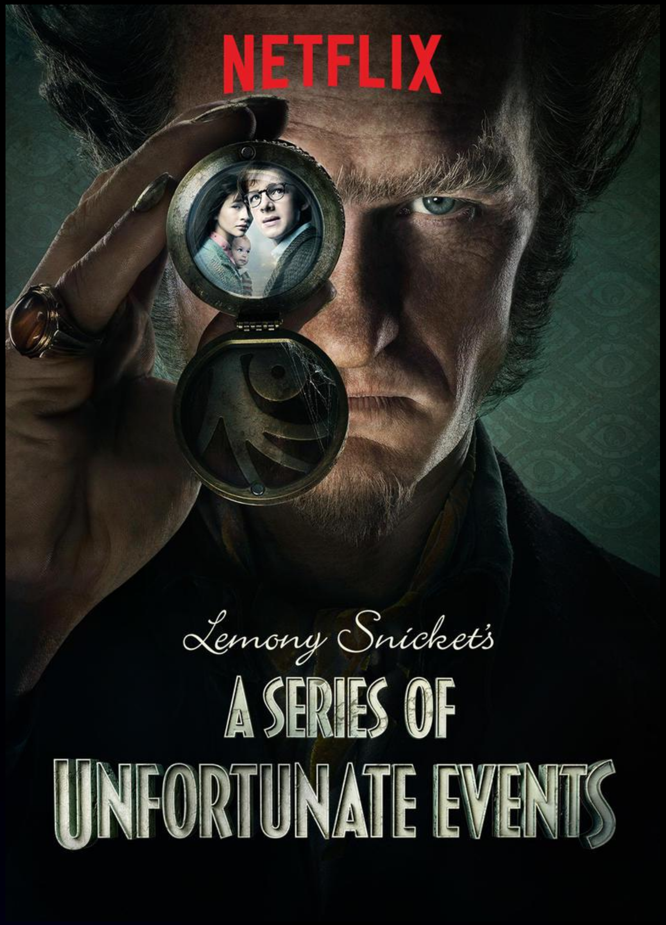
A day in the life of EVCache

Incidents and takeaways

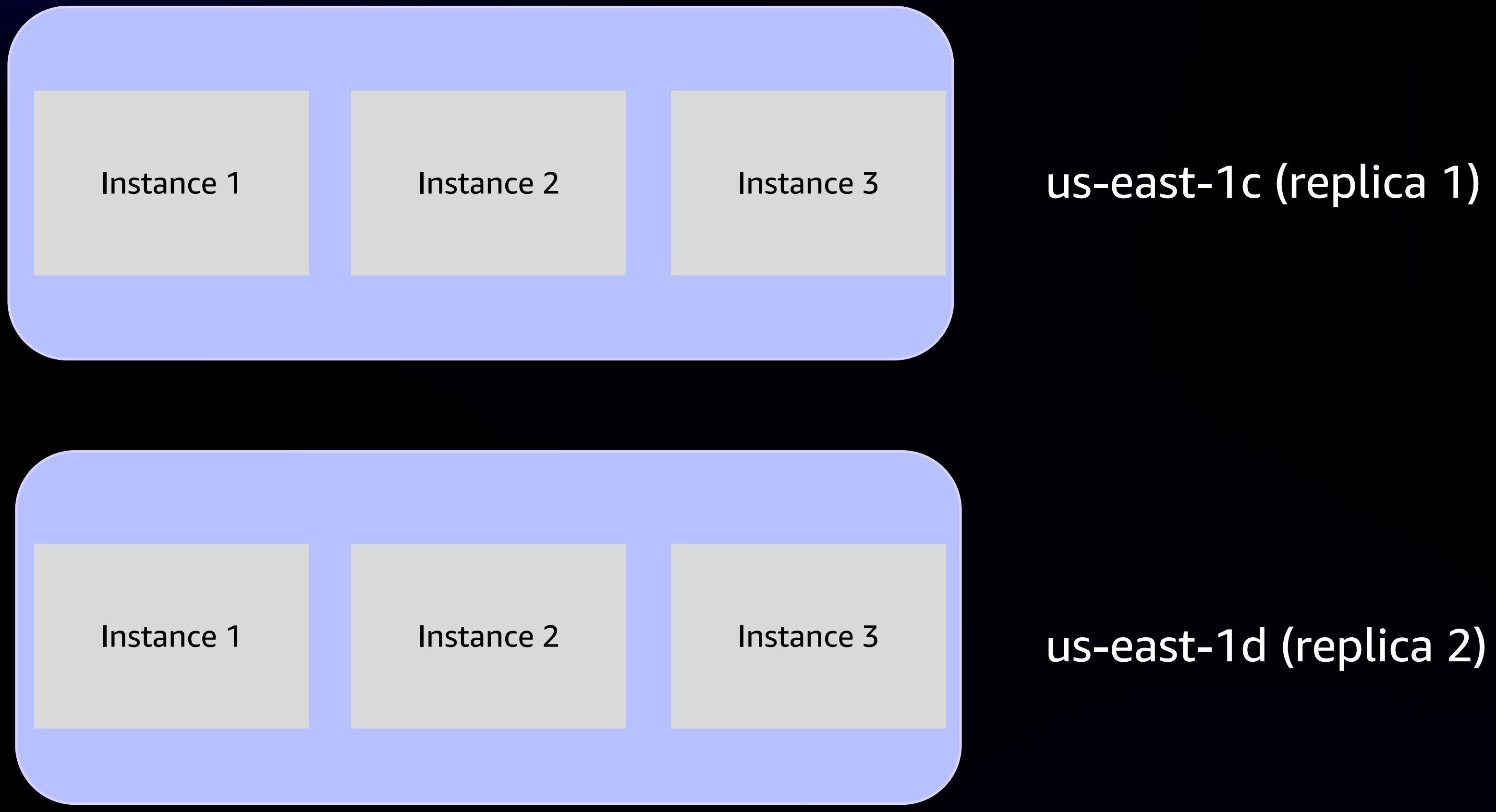
A day in the life of EVCache

- Instance crashes/disappears, AWS health check
- Memcached gets terminated
- Disks go bad
- Data get wiped out

"Instance warmer refills data"

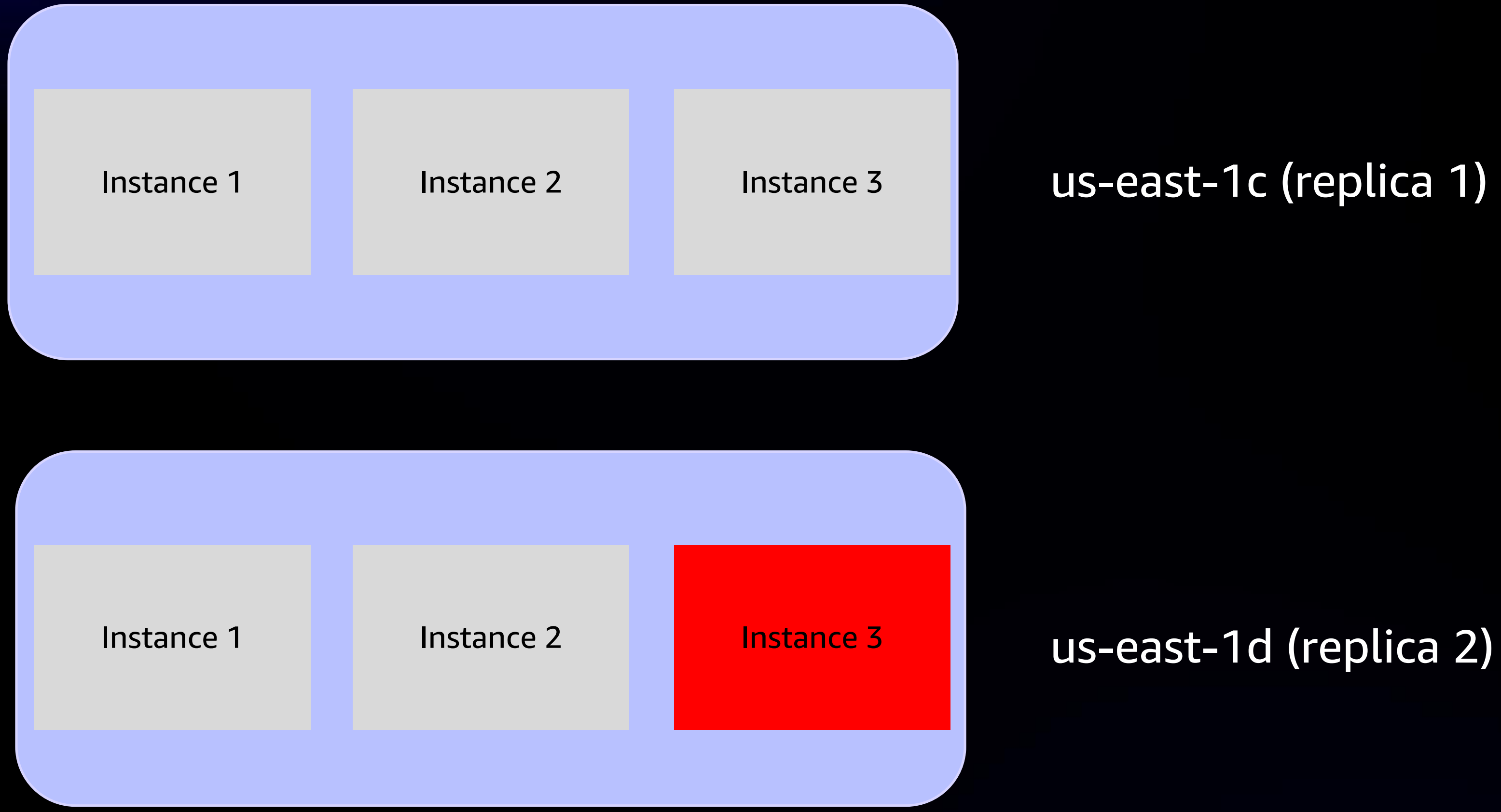


Instance warming



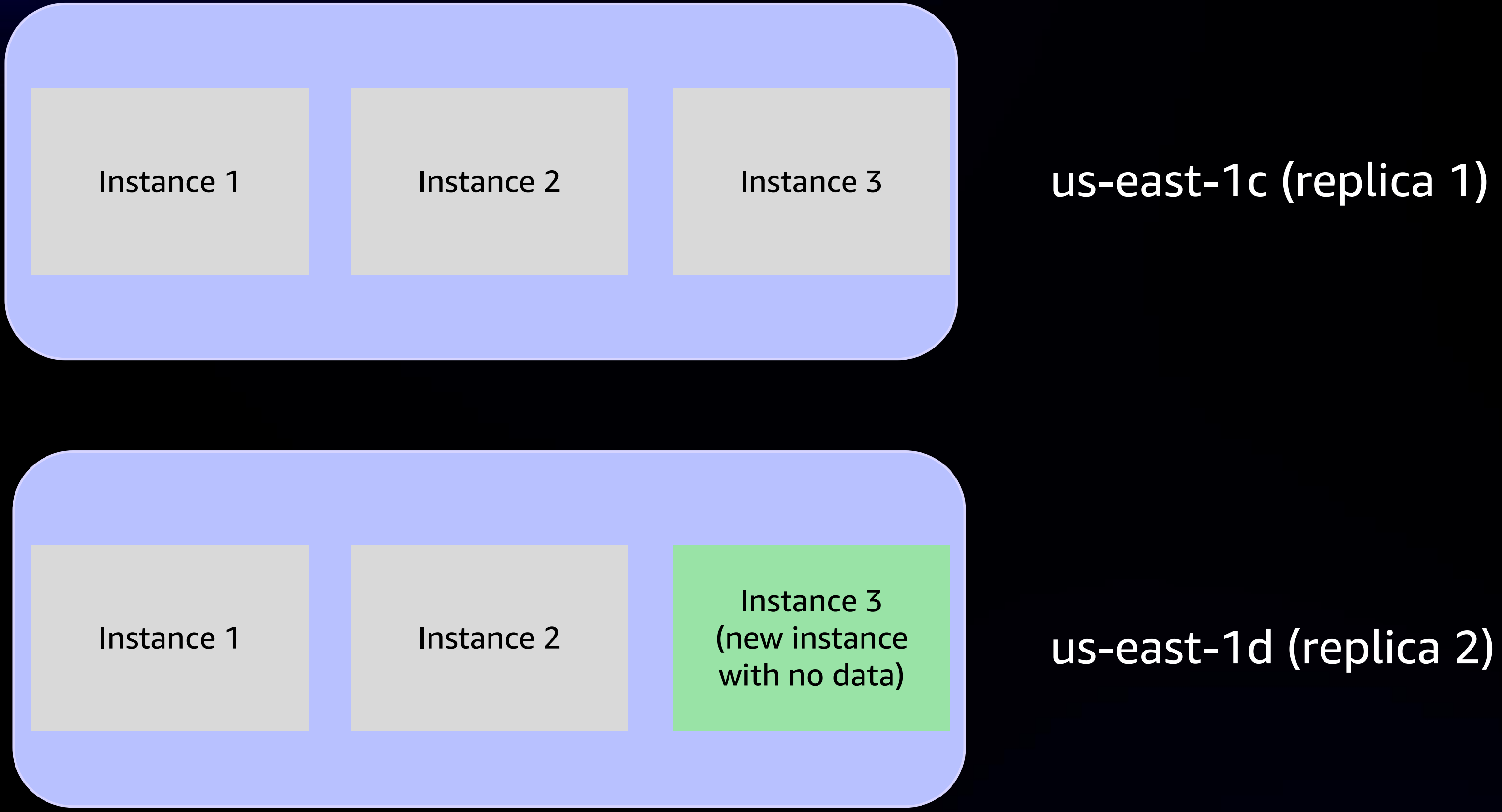
Healthy cluster

Instance warming



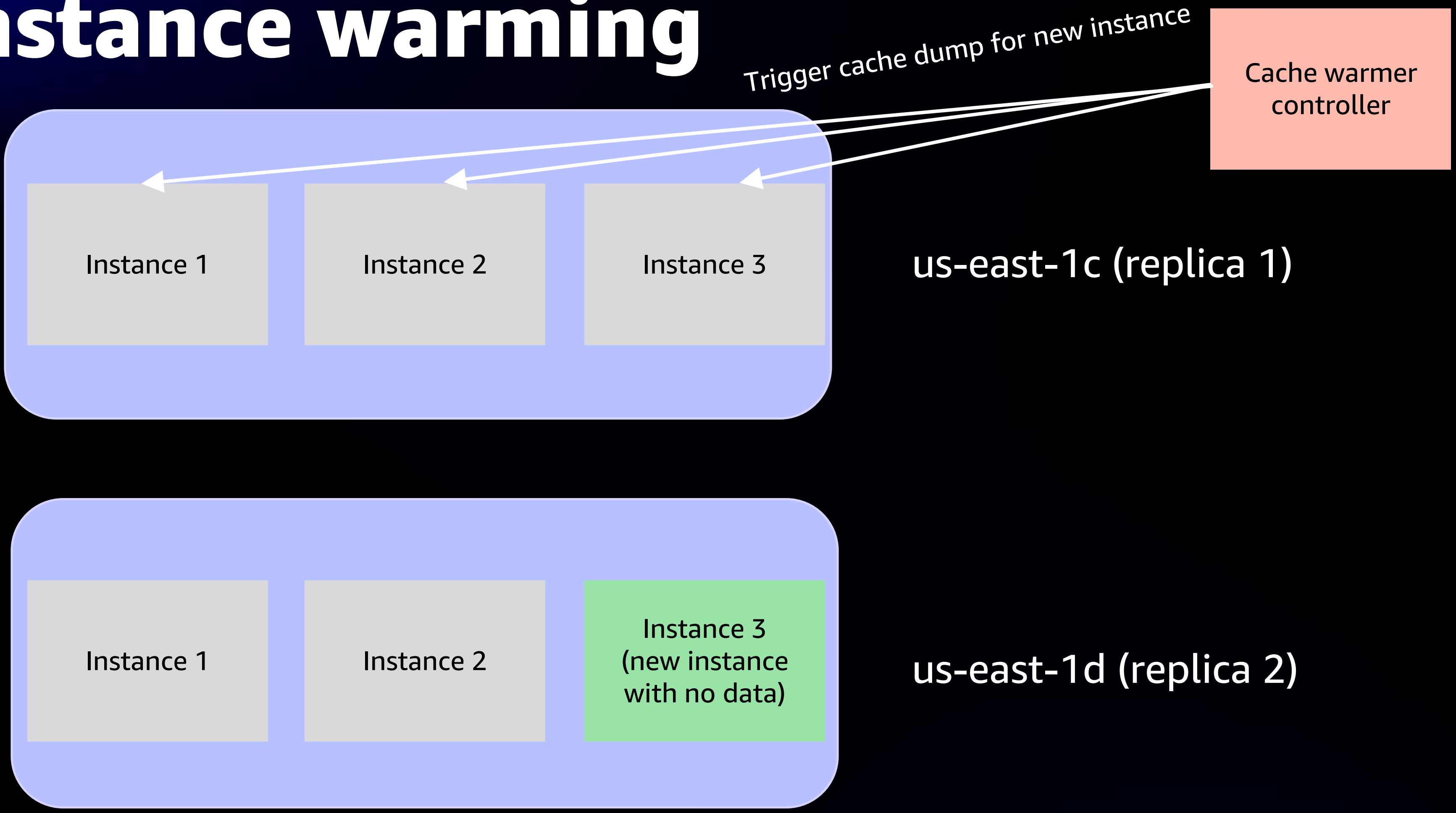
Unhealthy cluster in replica 2

Instance warming



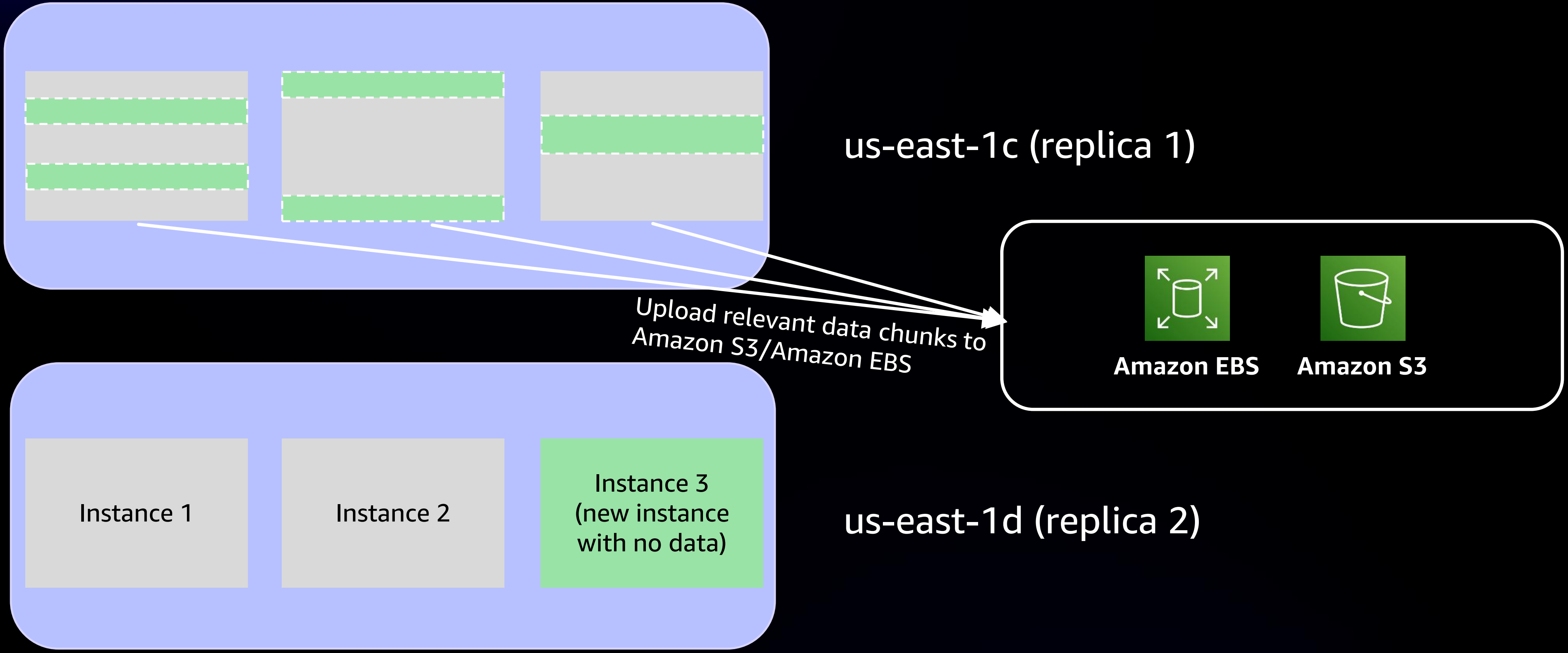
New instance coming up in replica 2

Instance warming



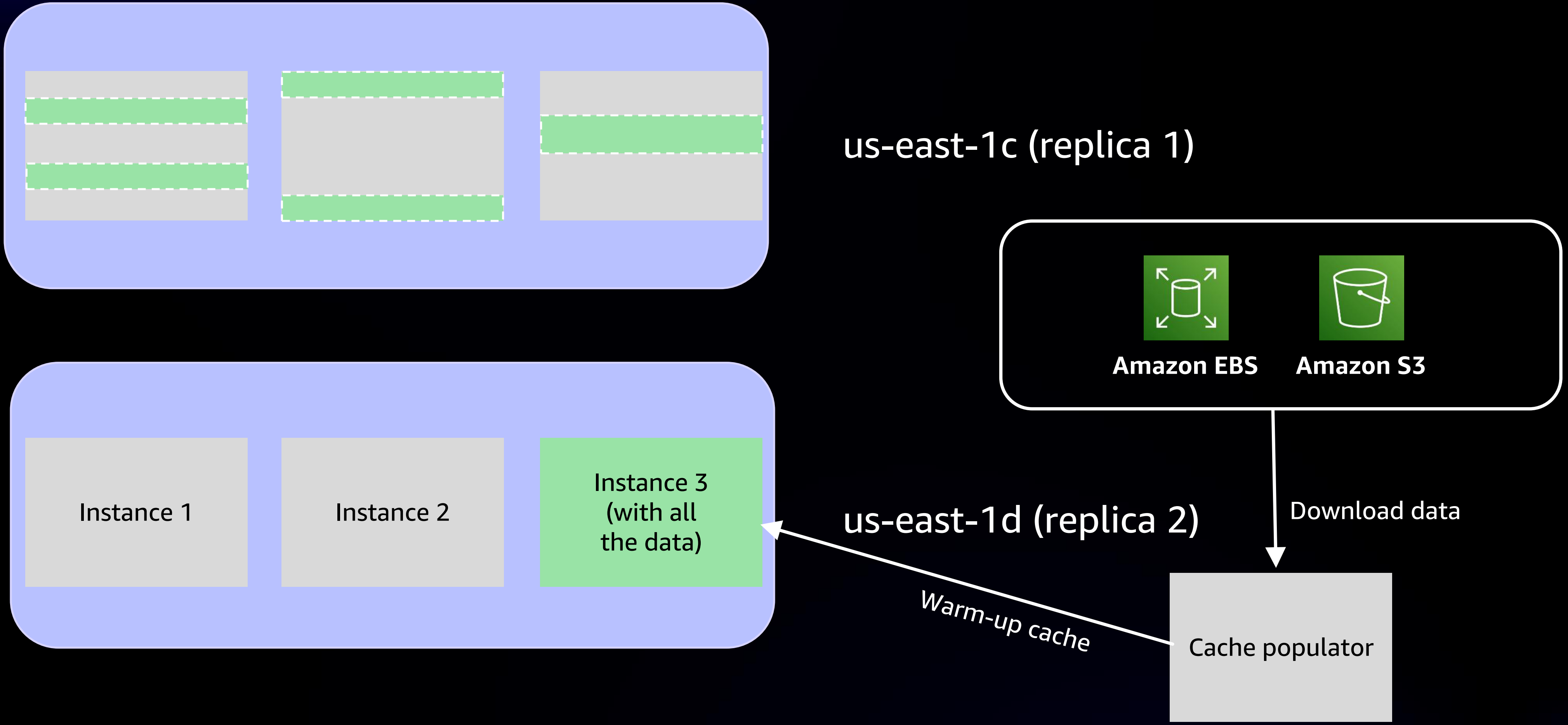
Cache warmer controller triggers cache dump for the new instance based on the new IP address

Instance warming


us-east-1c (replica 1)
Upload relevant data chunks to Amazon S3/Amazon EBS
Amazon EBS
Amazon S3
Instance 1
Instance 2
Instance 3 (new instance with no data)
us-east-1d (replica 2)

Cache warmer controller triggers cache dump for the new instance based on the new IP address

Instance warming



Cache populator downloads the data from Amazon EBS/ Amazon S3 and writes to the destination IP(s)

A day in the life of EVCache

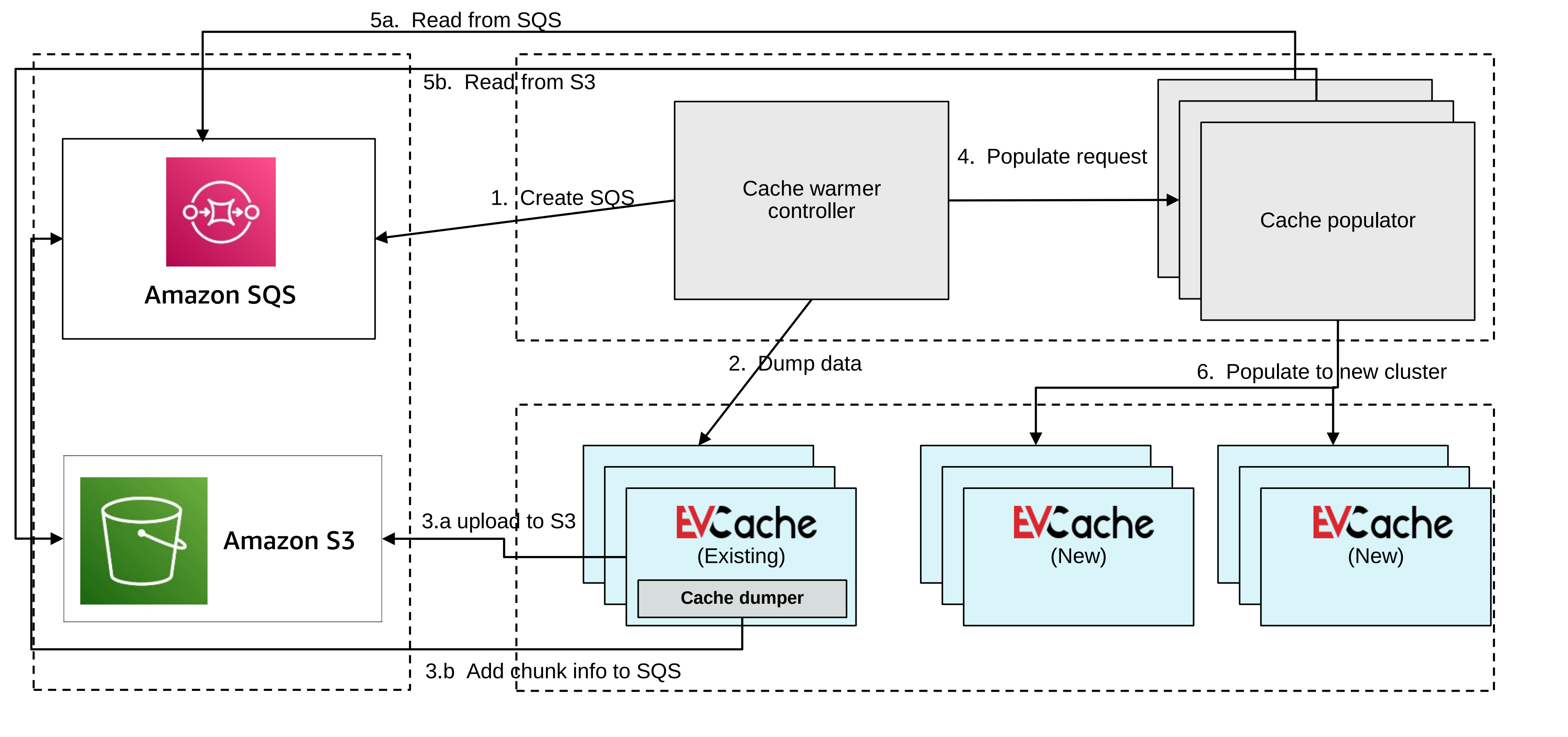
- Workload changes, scale out
- Data set sizes increase from TBs to PBs

“Cache warmer to move data, faster with Amazon EBS Multi-Attach”

Scaling up fast

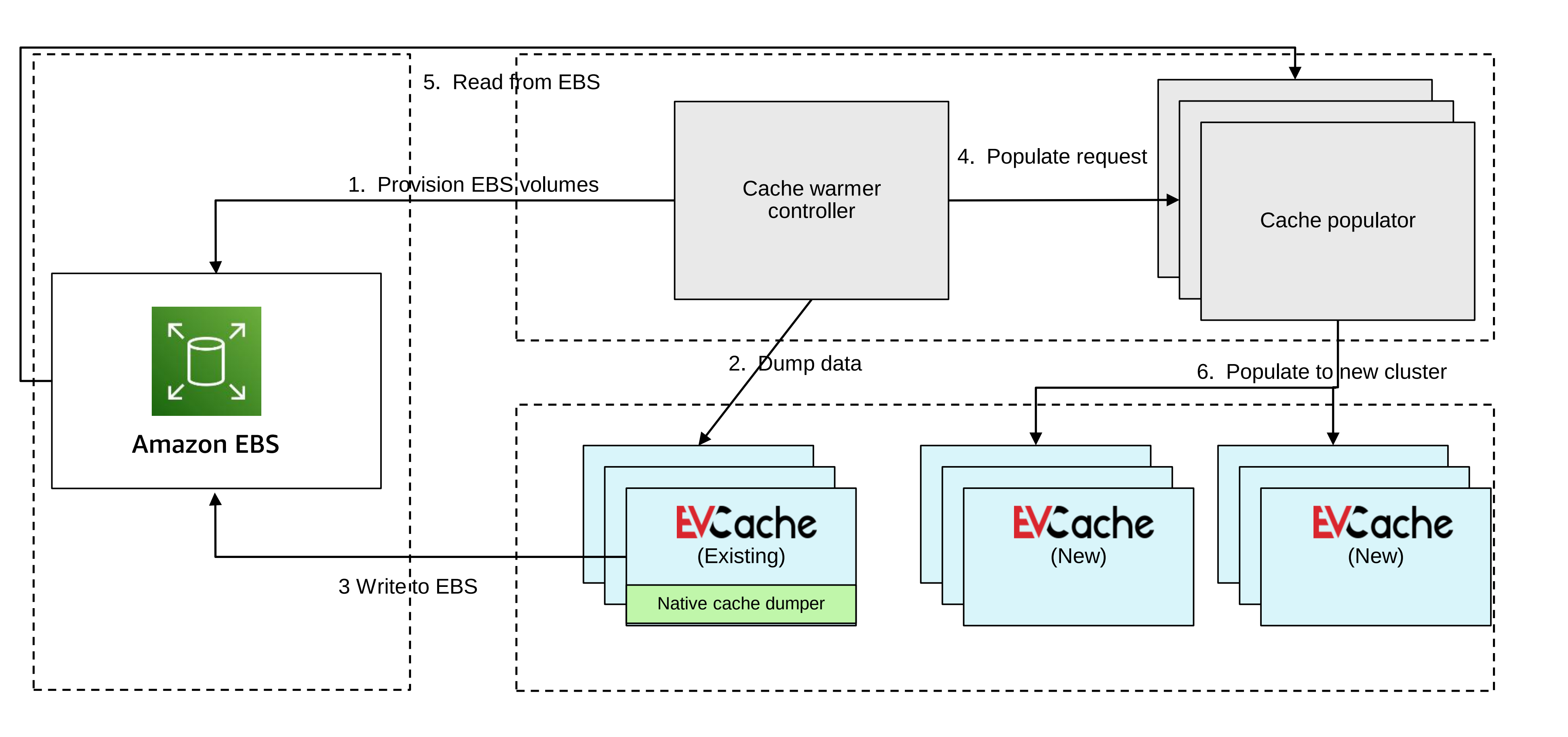
- New deployments are quite common
  - Increase/decrease capacity
  - Memcached upgrade (fixes/new features)
  - Base Amazon Machine Image (AMI) upgrades
  - Instance type changes
- New deployments cannot take read traffic right away
  - Wait until item TTL duration
  - TTL can range from a few hours to a few weeks

Scaling out fast



More context: https://netflixtechblog.com/cache-warming-agility-for-a-stateful-service-2d3b1da82642

Scaling up fast(er)



More context: https://netflixtechblog.medium.com/cache-warming-leveraging-ebs-for-moving-petabytes-of-data-adcf7a4a78c3

A day in the life of EVCache

Data corruptions by apps

"Snapshots and restores"

EDEN DUNCAN-SMITH
DANTÉ CRICHLOW
60
10
15
20
40
45
50
FROM PRODUCER SPIKE LEE AND DIRECTOR STEFON BRISTOL
SEE YOU YESTERDAY
A NETFLIX FILM
GOING BACK IS THE ONLY WAY FORWARD.
NETFLIX PRESENTS A 40 ACRES AND A MULE FILMWORKS PRODUCTION "SEE YOU YESTERDAY" EDEN DUNCAN-SMITH DANTÉ CRICHLOW MARSHA STEPHANIE BLAKE AND BRIAN "STRO" BRADLEY CASTING BY JESSICA KELLY SUZANNE SMITH CROWLEY AND REBECCA DEALY MUSIC SUPERVISOR STEPHANIE DIAZ-MATOS MUSIC BY MICHAEL ABELS COSTUME DESIGNER CHARLESE ANTOINETTE JONES EDITED BY JENNIFER LEE PRODUCTION DESIGNER JIMENA AZULA DIRECTOR OF PHOTOGRAPHY FELIPE VARA DE REY EXECUTIVE PRODUCERS MATTHEW MYERS JASON SOKOLOFF PRODUCED BY SPIKE LEE WRITTEN BY FREDRICA BAILEY & STEFON BRISTOL DIRECTED BY STEFON BRISTOL
NETFLIX
MAY 17 | NETFLIX

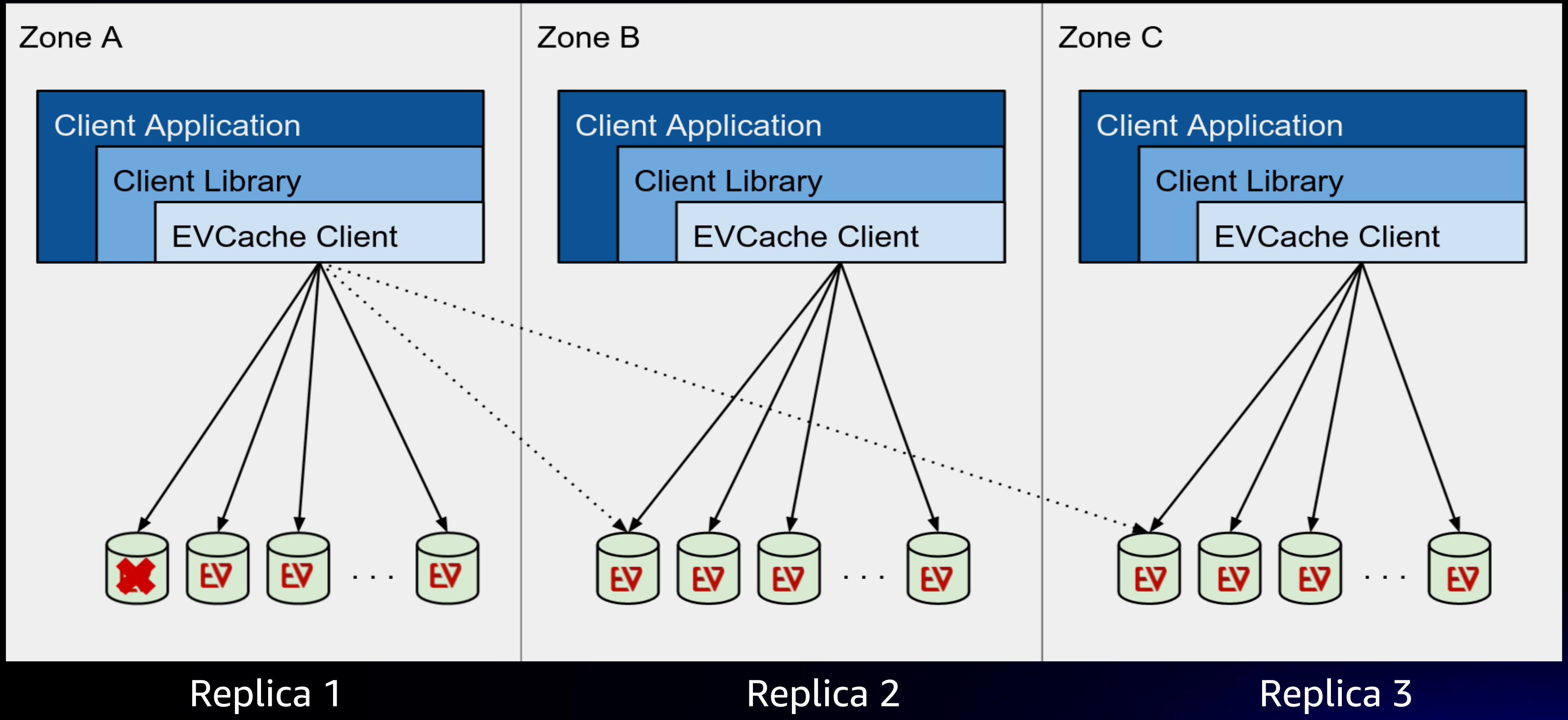
A day in the life of EVCache

- Data center/Availability Zone outages
- Co-related network failures
- Region failures

"Serve traffic from another Availability Zone or failover to another Region"

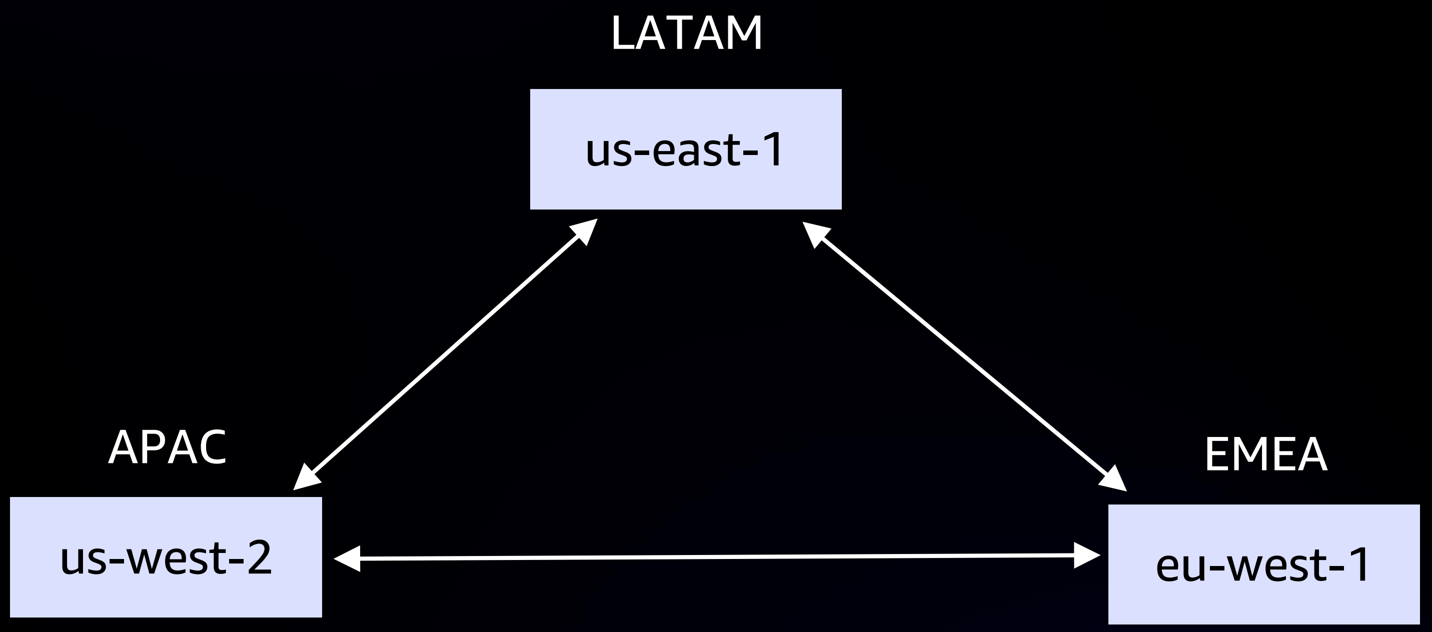
A day in the life of EVCache

Serving traffic from another data center/Availability Zone



A day in the life of EVCache

Serving traffic from another Region



Agenda

Introduction

High-level architecture

Workload characterization

Observability

A day in the life of EVCache

Incidents and takeaways

Incidents and takeaways

P95 latency issues

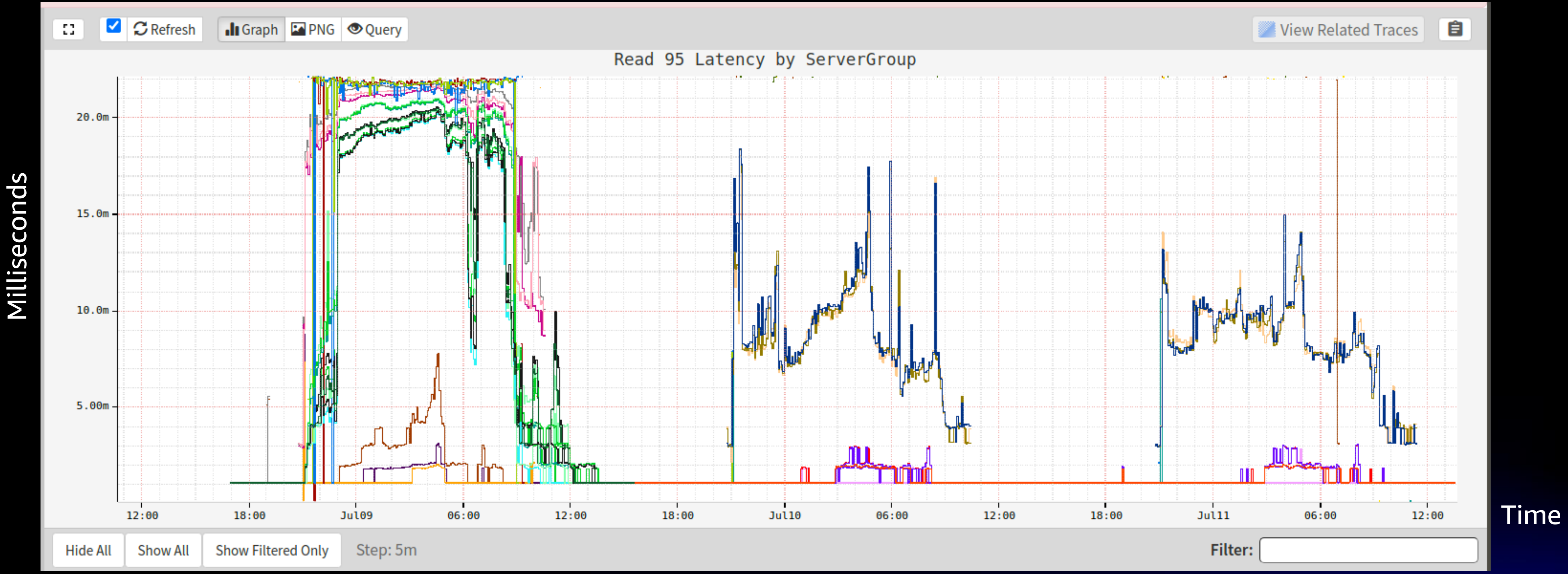
Heterogeneous client environments

Memory issues

- Application memory leaks
- Incorrect budgeting of resources
- Memory fragmentation

Triaging P95 latencies

- SLA with P95 <= ~5 ms*
- Disk latencies can come into play



* For this specific use case

Triaging P95 latencies

- Is online traffic contending with any offline traffic?
- Is the workload spiky?
- eBPF tools – biolatency, biosnoop, etc.

Heterogenous client environments

- Unable to connect from Mesos agents due to port exhaustion
- Standard tools/metrics don’t help

Identify network namespaces

```
$ ip netns list
720 (id: 6)
16190 (id: 0)
```

Heterogenous client environments

Locate the cgroup and the container

```
$ systemctl status `pidof java`
daemontools.service - Daemontools service supervision
   Loaded: loaded (/lib/systemd/system/daemontools.service; enabled; vendor preset: enabled)
   Active: active (running) since Mon 2021-04-12 01:33:36 UTC; 4 weeks 1 days ago
 Main PID: 3771 (svscanboot)
    Tasks: 1365
   CGroup: /system.slice/daemontools.service
           ├─ 720 /apps/mesos-1.7.2/libexec/mesos/mesos-containerizer launch
```

Heterogenous client environments

- Then we grouped connections by a server IP and observed the connection build-up

```
$ sudo ss -N 720 | grep "<server.ip.address.100>" | awk '{print $5}' | cut -d ':' -f 2  | wc -l
1020
```

- Clients were reconnecting and leaving old connections in the CLOSE_WAIT state
- Aggressive connection cleanup without leaving JVM to close them

Triaging memory issues

```
[Thu Oct 28 05:28:26 2021] nvme nvme2: pci function 0000:00:1e.0
[Thu Oct 28 05:28:26 2021] nvme 0000:00:1e.0: enabling device (0000 -> 0002)
[Thu Oct 28 05:28:26 2021] nvme nvme2: 2/0/0 default/read/poll queues
[Thu Oct 28 05:28:26 2021] kworker/u24:2: page allocation failure: order:4,
mode:0x40dc0(GFP_KERNEL|__GFP_COMP|__GFP_ZERO), nodemask=(null),cpuset=/,mems_allowed=0
[Thu Oct 28 05:28:26 2021] CPU: 1 PID: 16990 Comm: kworker/u24:2 Tainted:
P           OE     5.3.0-1023-aws #25~18.04.1-Ubuntu
[Thu Oct 28 05:28:26 2021] Hardware name: Amazon EC2 i3en.3xlarge/, BIOS 1.0 10/16/2017
[Thu Oct 28 05:28:26 2021] Workqueue: nvme-wq nvme_scan_work
```

Triaging memory issues

First step is knowing state of memory

```
$ free -m
                 total       used       free     shared  buff/cache   availableMem:
                 31704      31348        242          8         113           37
Swap:              0          0          0
```

^ Looking at this, we can deduce that the system is running low on memory

```
$ cat /proc/buddyinfo | ./analyze.sh
  15.54 MiB    Node 0, zone DMA        1     0     0     1     2     1     1     0     1     1     3
  134.90 MiB   Node 0, zone DMA32   1891   990   512   553   322   221   121    31     1     0     0
  94.72 MiB    Node 0, zone Normal  1646  2053  3238   693     0     0     0     0     0     0     0
```

^ and is also highly fragmented, as higher order pages are not available

Triaging memory issues

Next is to look at top consumers of memory

```
$ ps -o pid,user,%mem,command ax | sort -b -k3 -r | head -n 2
  PID USER     %MEM COMMAND
2284 nfsuper  95.2 /apps/memcached/bin/memcached <masked the parameters>

$ pmap -x 2284 | tail -n 1
total kB         32168392 30923660 30922808
```

^ Since application in question is Memcached, we have a good understanding of how much the Memcached process itself should take, but if for some reason the RSS memory keeps climbing, then there is a high likelihood of memory leak in the application

Triaging memory issues

- Amount of memory consumed by TCP or UDP buffers on the system can be profiled by checking the output of `cat /proc/net/sockstats`
  - Sockstats shows the number of sockets opened on that machine and the memory occupied by the node
  - Whenever the application is slow or there is a spike in traffic, TCP memory usage goes up, leaving user space processes little to no memory to operate

Triaging memory issues

A few other things to look at

- What is the QPS or rate at which the requests are served on this cluster?
- What is the read/write network bandwidth on these nodes?
- Amount of memory consumed by kernel – monitor via slabtop

Takeaways

- Architectural patterns used in operating data stores at scale on AWS
- Know your use cases and workload patterns
- Invest in observability from day one
- Continuously improve on reliability and scalability

Thank you!

Tharanga Gamaethige

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

Prudhviraj Karumanchi

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