AWS re:Invent

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

ADM301

Under the hood at Amazon Ads

Rachit Chawla
Director Software Dev.
Amazon

Anuj Joshi
Principal Engineer
Amazon

Kun Liu
Director Machine Learning
Amazon

Ish Singh
Senior Solutions Architect
AWS

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Agenda

- Introduction – Amazon ads
- Omnichannel marketing with Amazon ads
- Ads pillars, challenges, and architectural evolution
- Evolution of ad campaigns and ad serving at Amazon
- Improving ad selection and relevance with deep learning
- Learnings and best practices

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Amazon ads – Helping grow businesses and brands of all sizes!



Build your brand



Grow sales

Connect with shoppers



Measure results



aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Build an omnichannel marketing strategy using Amazon ads

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Build omnichannel strategy with Amazon ads

- Build awareness
- Increase consideration
- Engage customers
- Drive purchase
- Retail and brand readiness



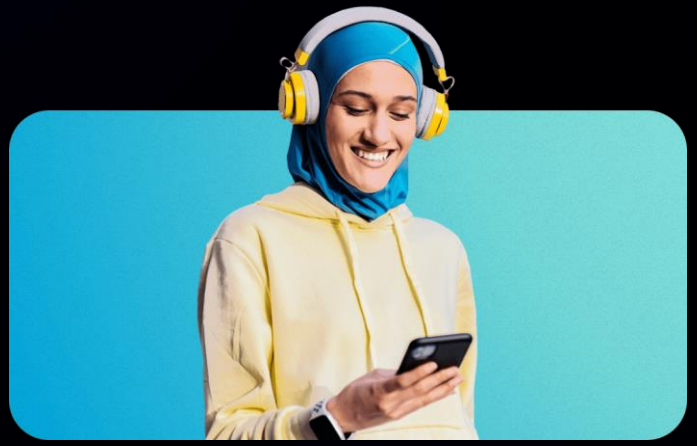
aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Working backwards from the customers



Advertisers



Shoppers



3P developers

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Advertising pillars, challenges, and architectural evolution

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

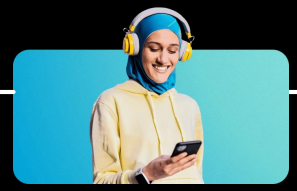
The big picture – Amazon ads



Advertisers



3P developers



Shoppers

Budgeting

Campaign management

Ad server

Data lake

Ad events

Machine learning

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Building a seamless campaign management experience for our advertisers

Campaign management





Use cases

- Beyond CRUD APIs – list/search/query
- Managing millions of campaigns with millisecond latencies

Tens of billions of campaign objects

Highly available >99.9999%

Low latency read/writes

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Highly available and scalable budget service

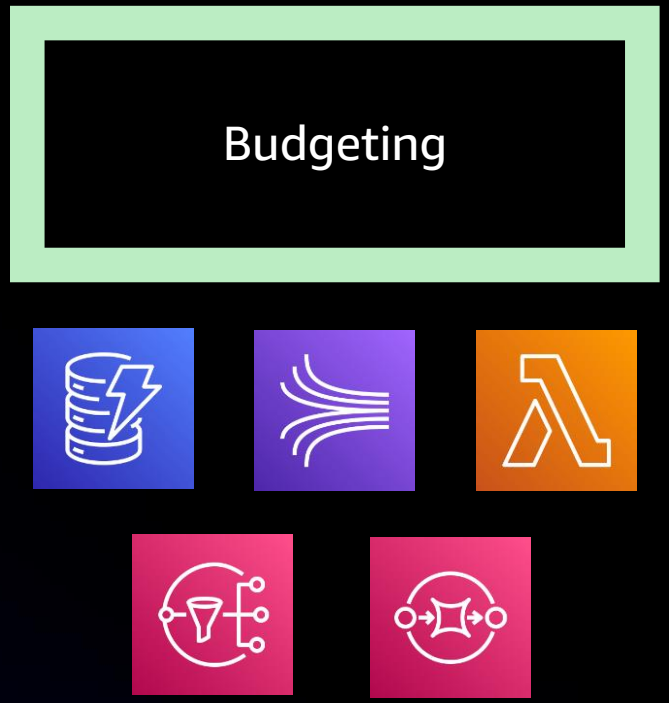
USING AMAZON DYNAMODB, AWS LAMBDA, AND AMAZON KINESIS TO BUILD A HIGHLY AVAILABLE AND SCALABLE BUDGET MANAGEMENT SYSTEM

Use case

- High scale and auto-scale for peak events (Prime day)
- Near real-time budget enforcement (spend ≤ budgets)

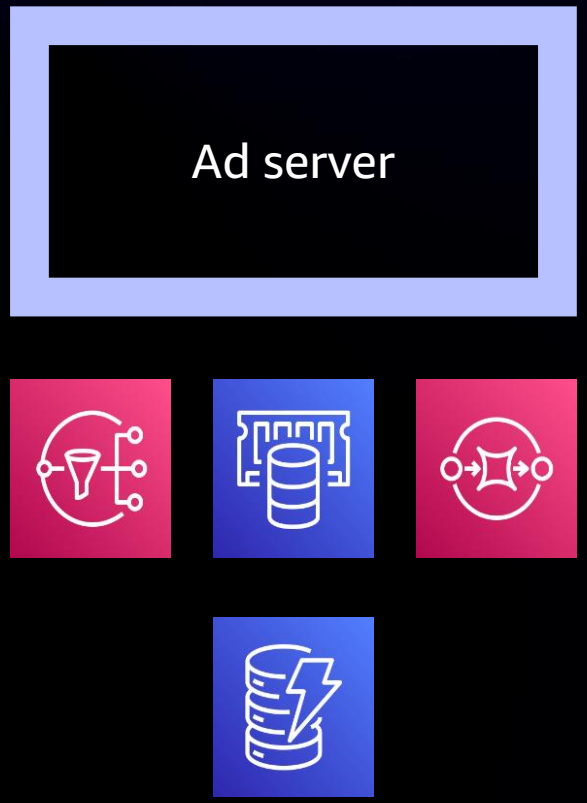
Cost reduction of more than 25%

Low latency of <1 second



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Low-latency, high-throughput ad serving



Use case

- Low-latency, high-scale request rate, large number of ads/requests to process
- Large amount of data needs to be cached and precomputed in this architecture

Hundreds of millions of ads per second within 120 ms

Network bandwidth of Tbps

Exponential growth

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Managing hundreds of petabytes of data

Use case

- React to ad events in real time, improving shopper experience and overall advertiser efficiency
- Ingest real-time ad events, enabling analytics and model training within an hour of the event

Analytics



Analyze hundreds of petabytes of data

End-to-end analytics within hours

Ingestion of real-time ad events



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Deep learning models to score product ads to improve ad relevance

Machine learning



Use case

- Scoring relevant product ads for our shoppers, resulting in higher clicks and conversions
- Build a system to score thousands of ads per request at 100K+ TPS in tens of milliseconds

Predicting hundreds of millions of ads per second across hundreds of models

Running offline & online ML models

Scoring thousands of ads per request within milliseconds

aws

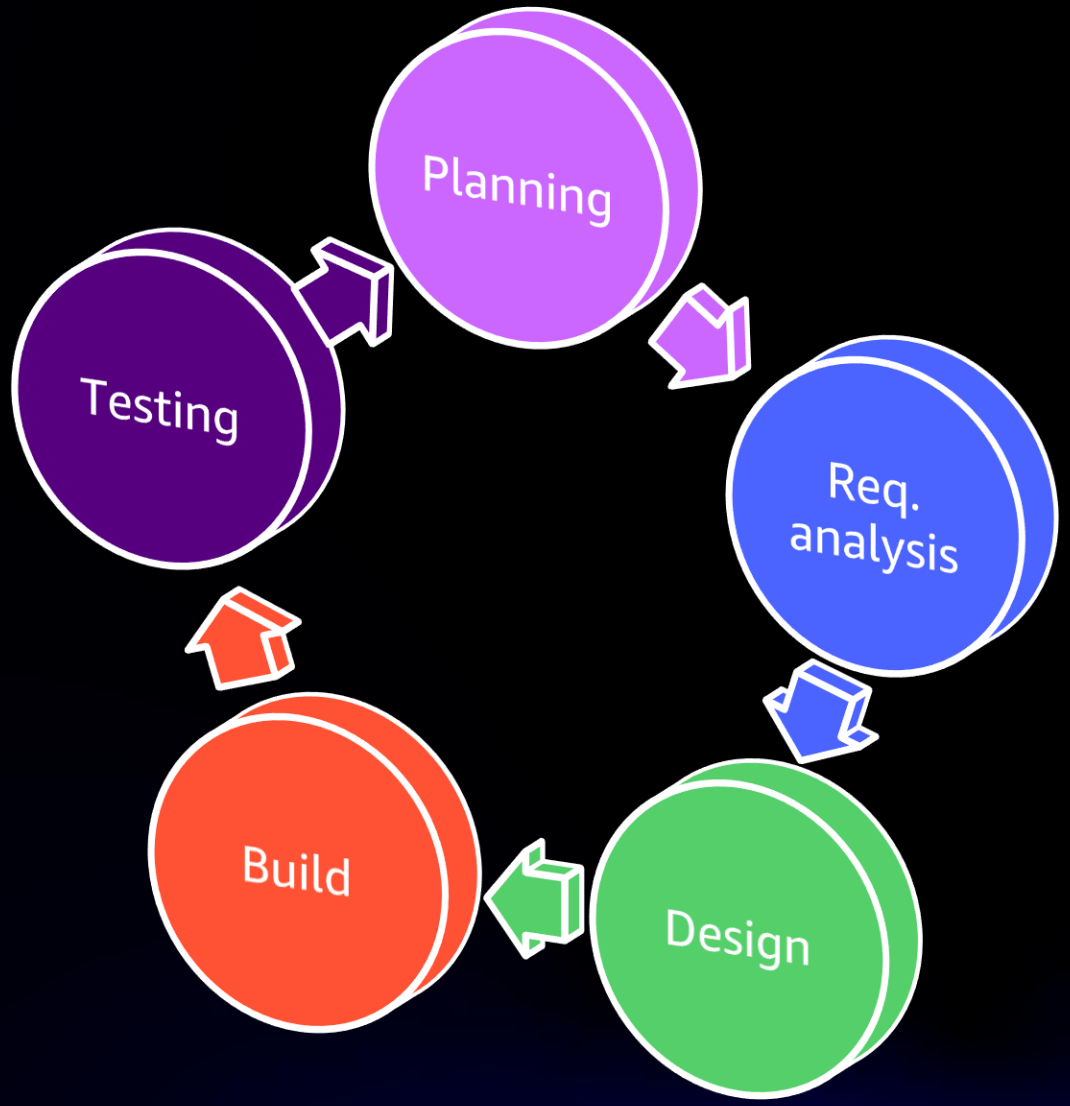
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Amazon Ads architecture that works at scale

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Design principles

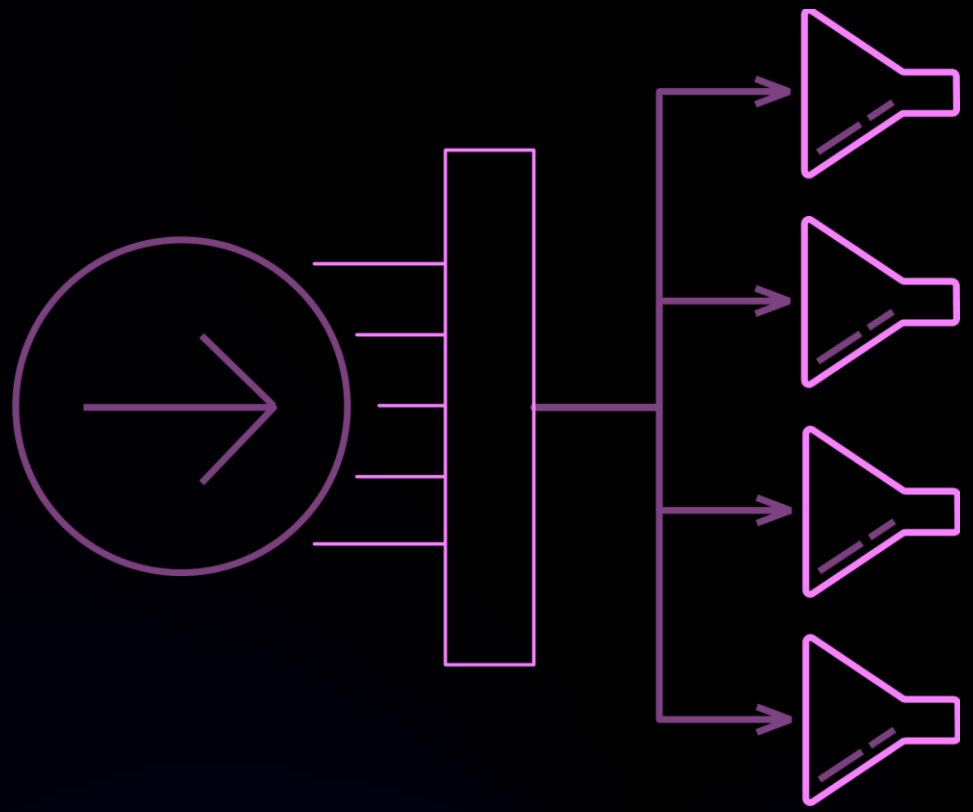
1. Iterative development



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Design principles

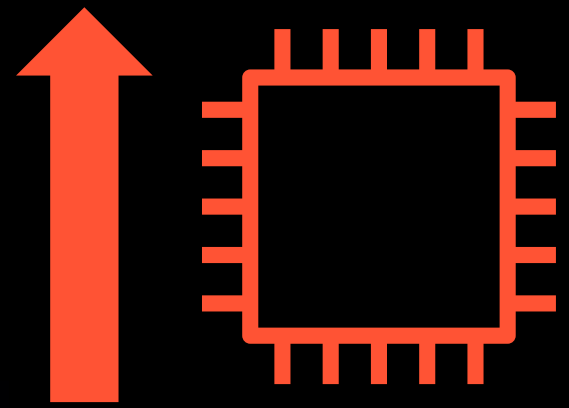
1. Iterative development
2. Event-driven architecture



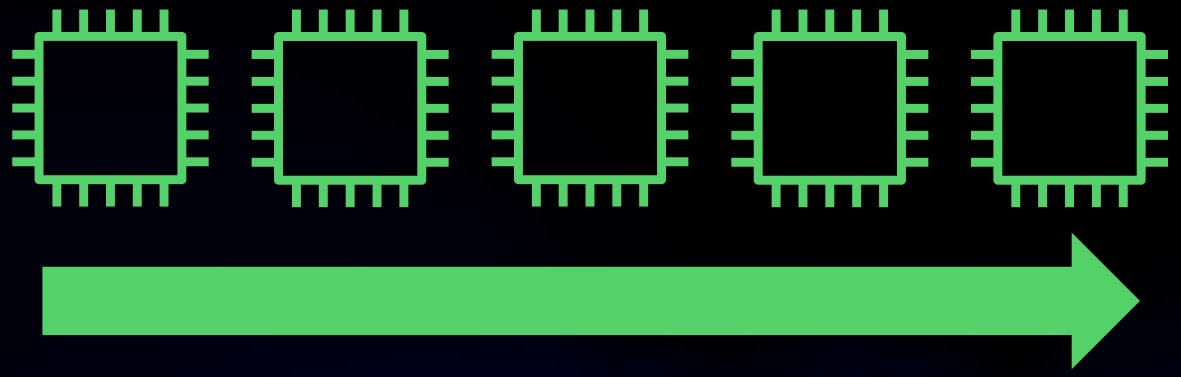
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Design principles

1. Iterative development
2. Event-driven architecture
3. Horizontal scaling



Vertical scaling

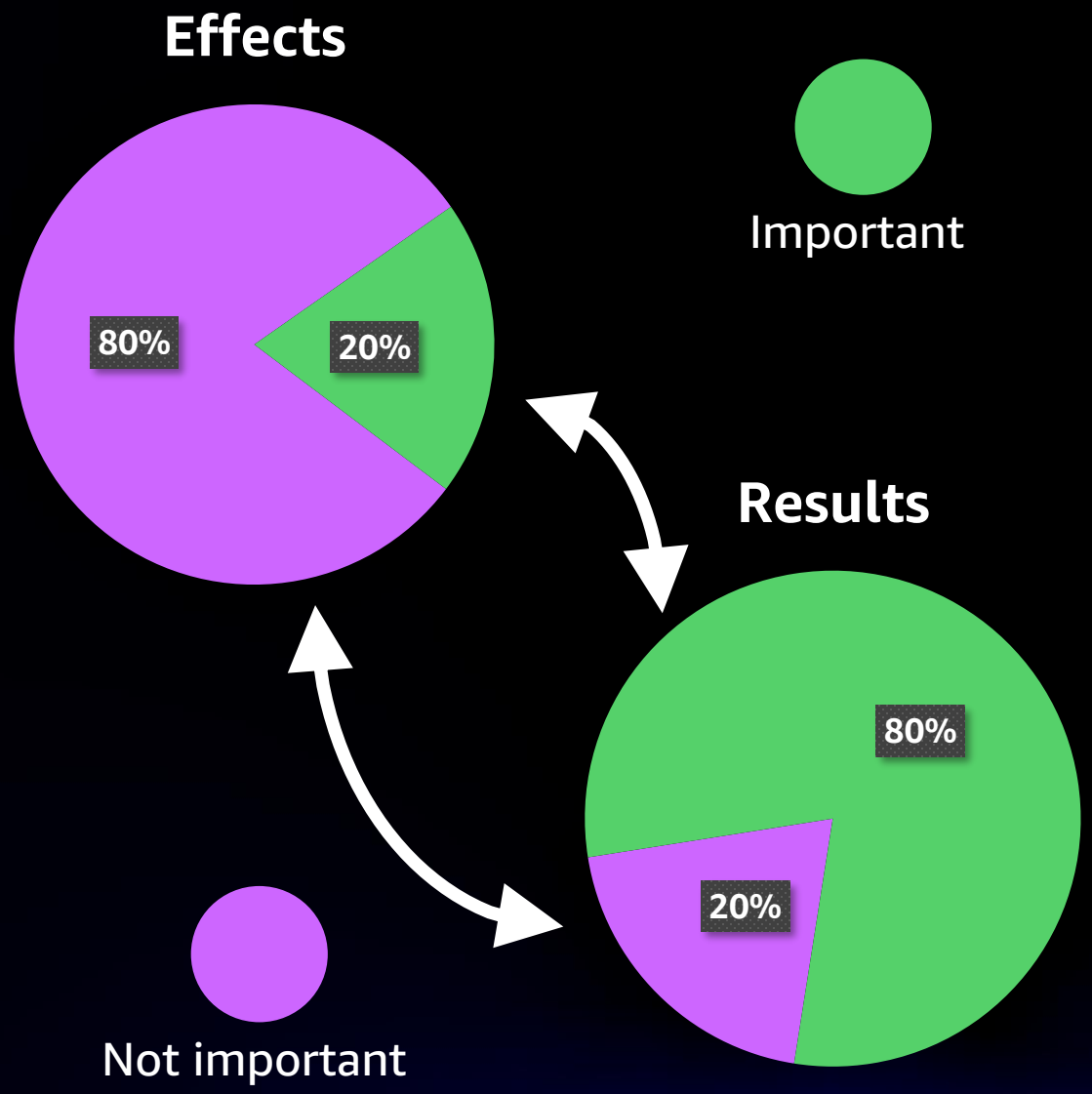


Horizontal scaling

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Design principles

1. Iterative development
2. Event-driven architecture
3. Horizontal scaling
4. Pareto principle



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Design principles

1. Iterative development
2. Event-driven architecture
3. Horizontal scaling
4. Pareto principle
5. Invent and simplify



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Campaign management



Advertisers



Shoppers

Budgeting

Campaign management

Ad server



3P developers

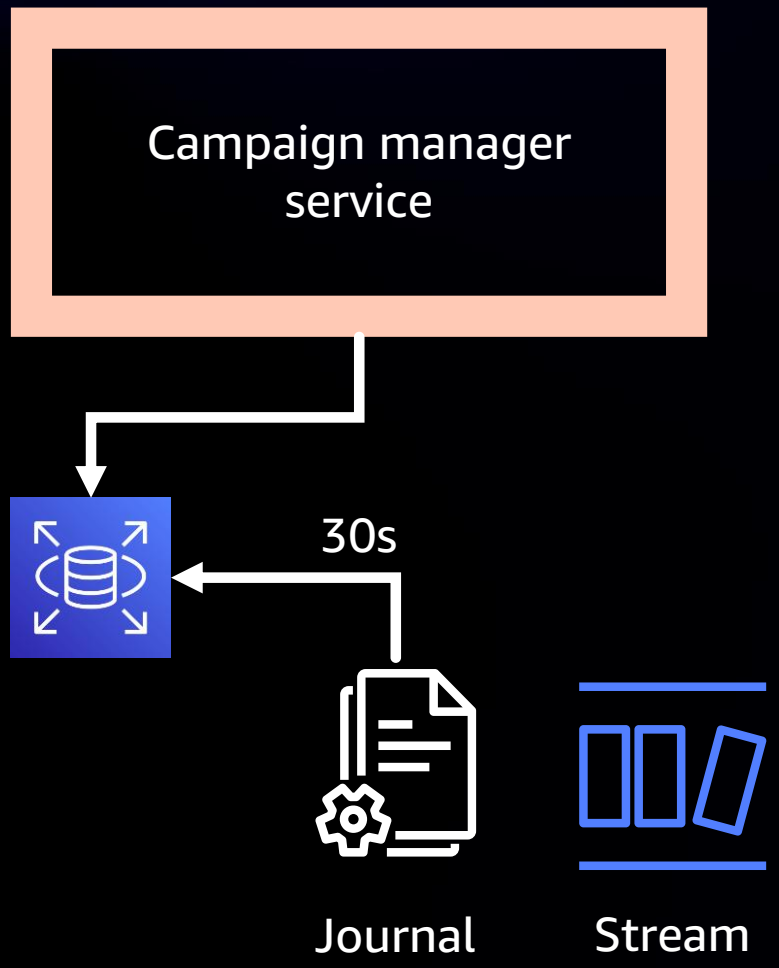
Data lake

Ad events

Machine learning

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Campaign management – The beginning



Java service was writing to Amazon RDS

Poller job would create a journal for the latest updates and stream to the clients

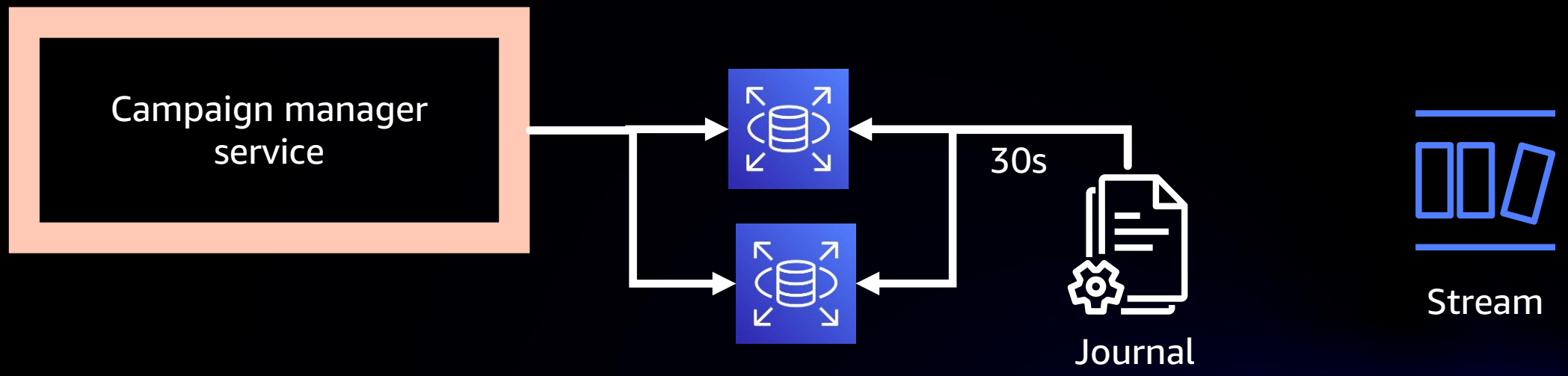
aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Campaign management – Sharding to scale

Used AdvertiserId to create multiple shards

Poller got a little complex

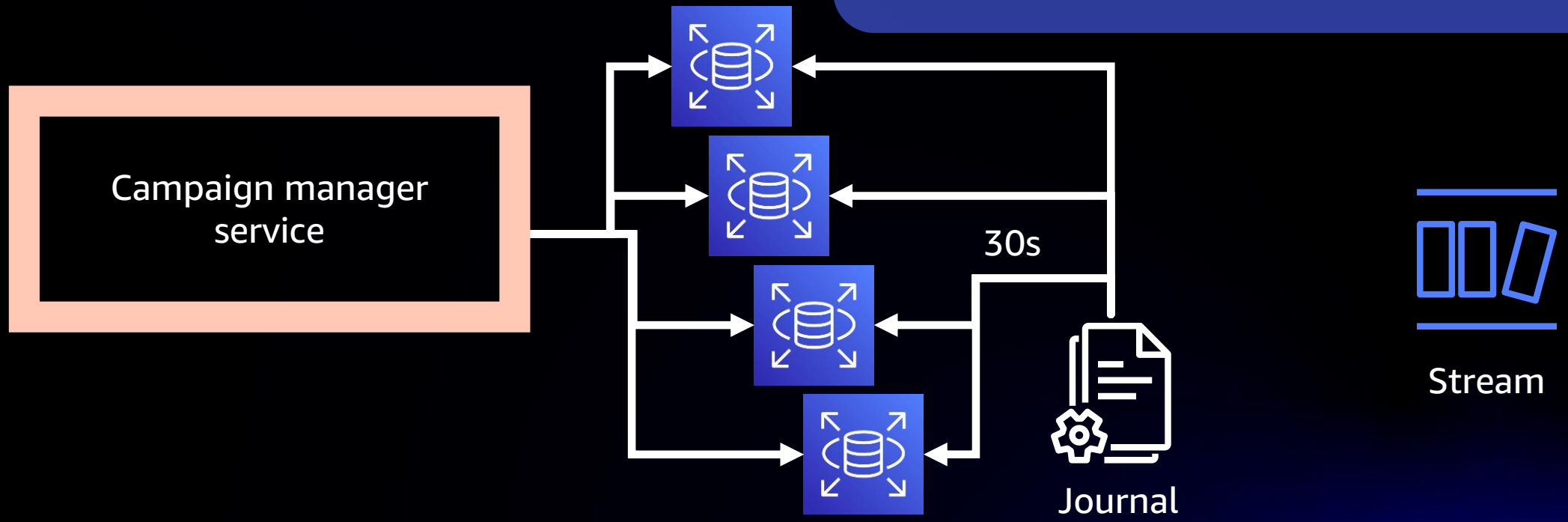


© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Campaign management – RDS bottleneck

Increased load on Amazon RDS led to downtime

Manual re-sharding was hours of outages as well

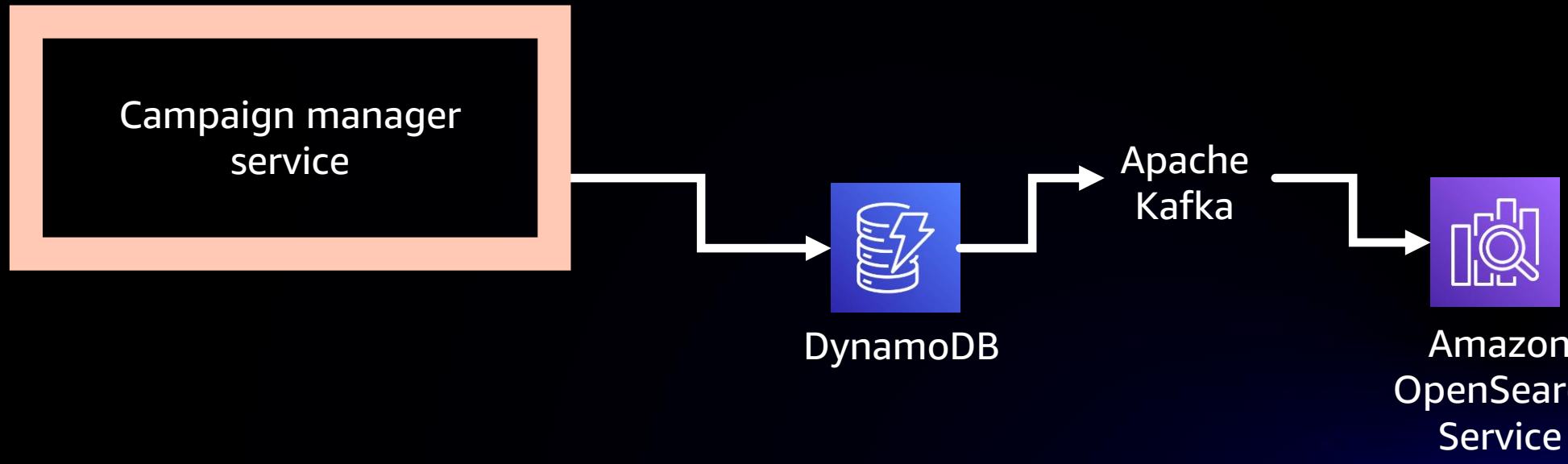


© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Campaign management – First overhaul

Replaced Amazon RDS with DynamoDB and Amazon OpenSearch Service (formerly Amazon Elasticsearch)

Connected DynamoDB streams to Kafka

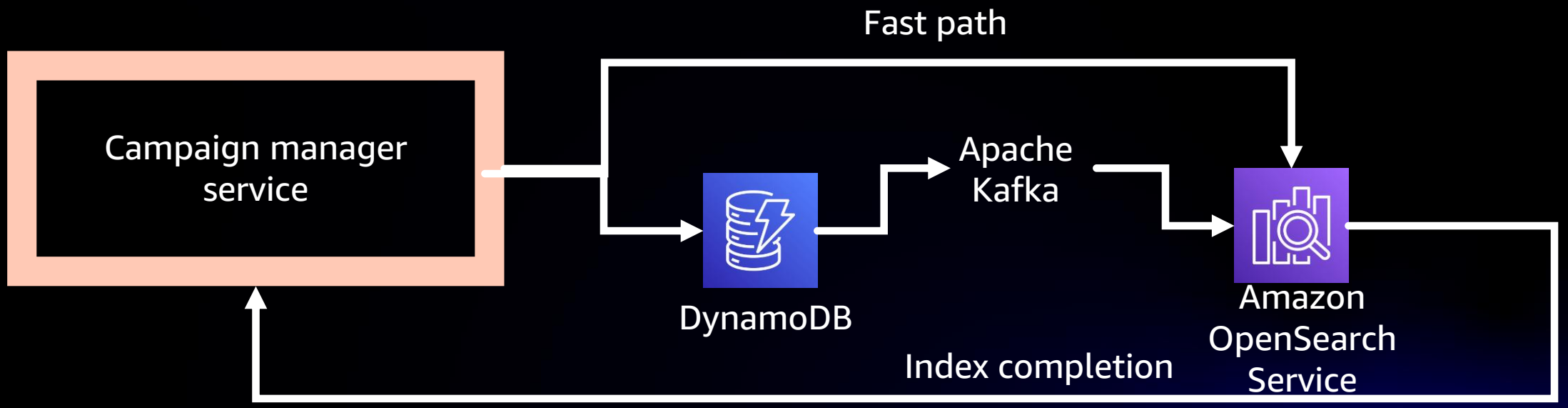


© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Campaign management – Problem #1

Read after write inconsistency

Fast path and dirty flag

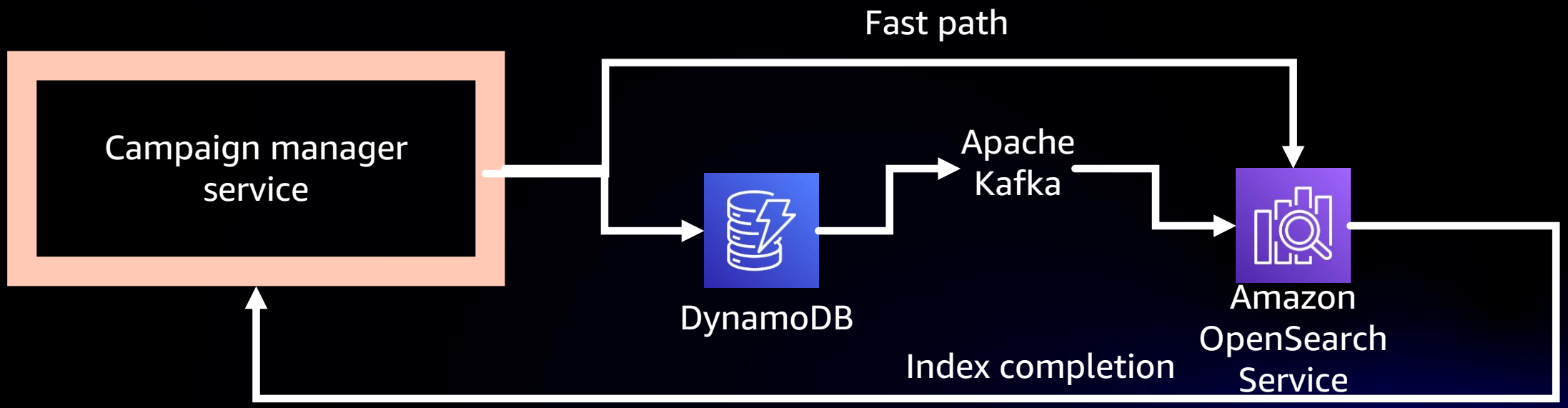


© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Campaign Management – Problem #2

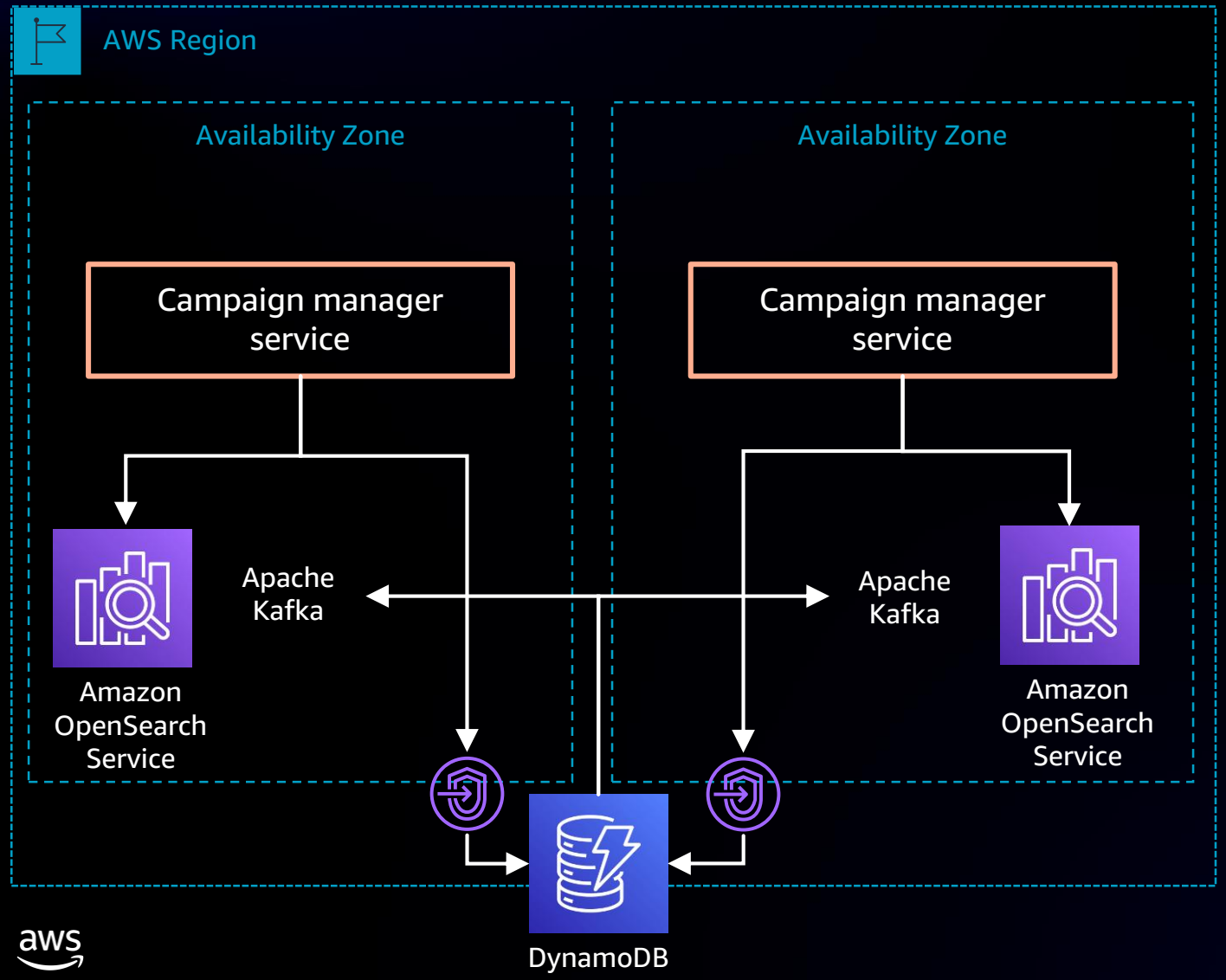
Amazon OpenSearch Service indexing was slow at our throughput

Simple upgrade did the trick!



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Campaign management – Success



1. Process thousands of events
2. High availability >99.9%
3. Horizontally scalable
4. Real-time notifications
5. Global Campaign data replication

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Building ad server to scale



Advertisers



3P developers



Shoppers

Budgeting

Campaign management

Ad server

Data lake

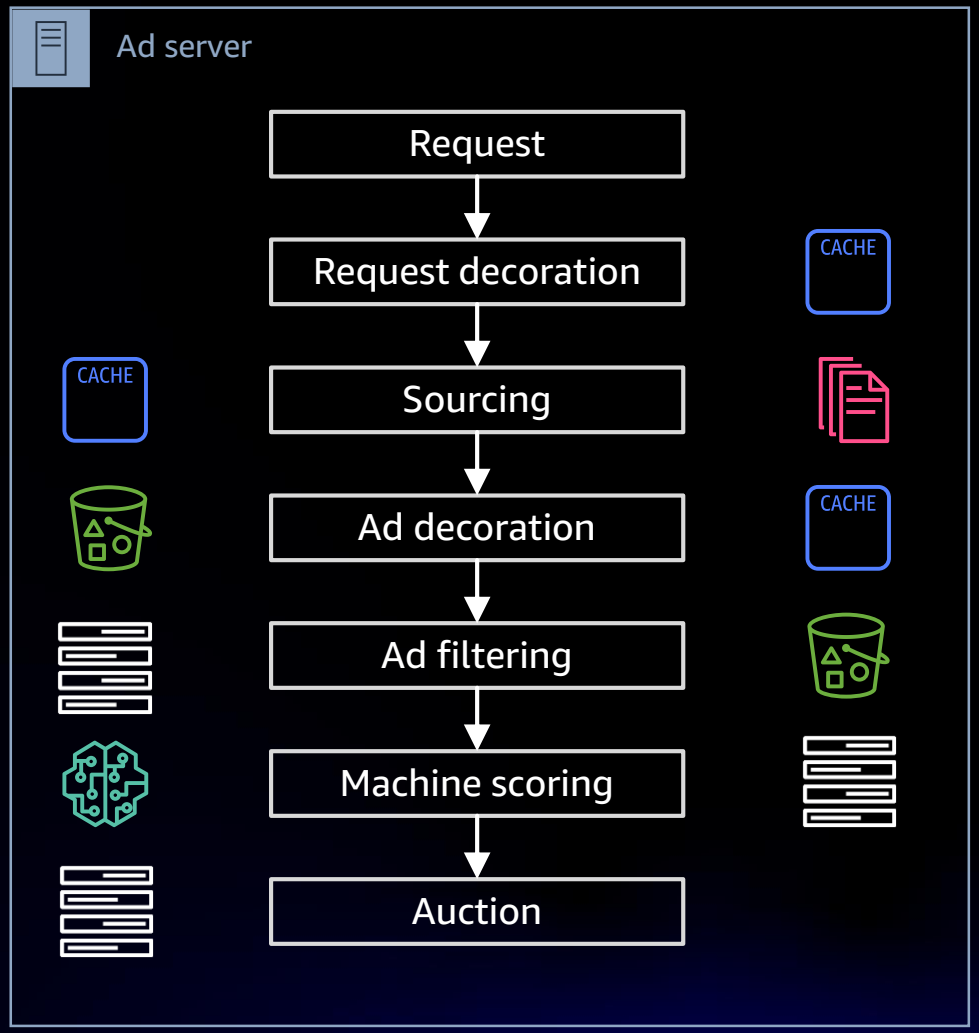
Ad events

Machine learning

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Ad server – We started (Vertically scaled)

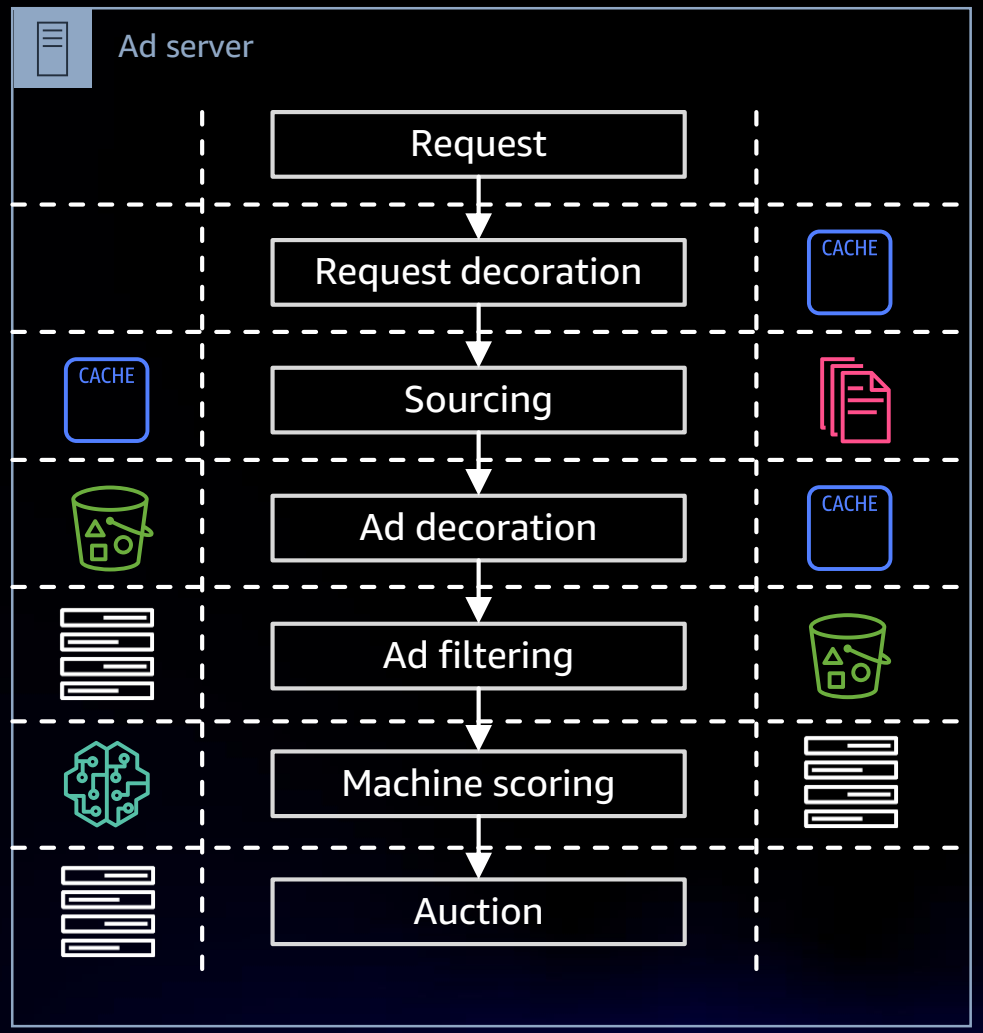
1. All going on a single host
2. 300 GB search index
   150 GB of machine learning models
   300 GBs of caches
3. 1.6M ads per second and 120 ms latency requirement



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Ad server – Tenets

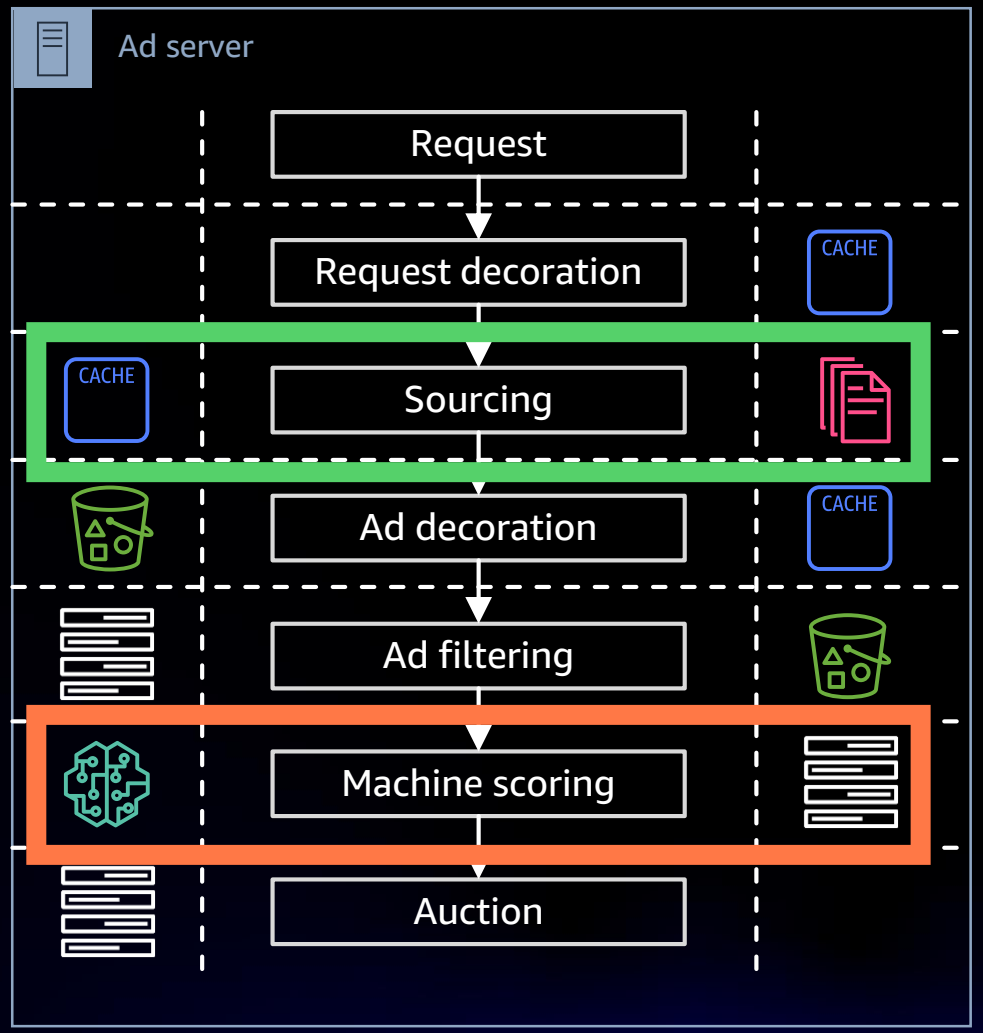
1. Distributed service-oriented architecture
2. Precomputed cached signals
3. Horizontally scalable
4. Transparency



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Ad server – Going distributed

1. Moved sourcing and ML as remote services
2. Moved host caches over the network
3. Replaced in-memory files with remote caches
4. Employed purpose-specific hardware



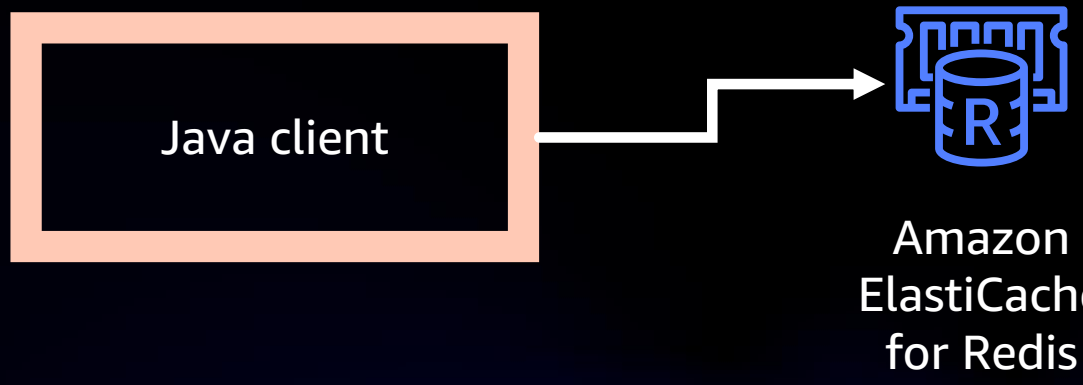
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Ad server – Caching at scale

Redis cluster invoked by a standard Java client

P99 latencies of 4 ms were not acceptable

Largest cache would read 5 keys per ad – 6M kps



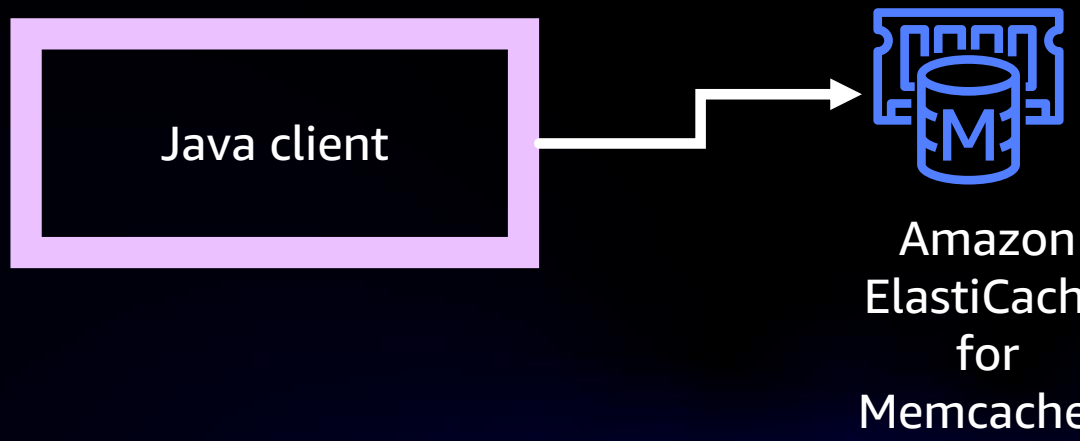
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Ad server – Caching at scale

Payload of 700 bytes, network clogged: 700B • 6M/GB • Gb = 31 Gbps

Each host was vulnerable for data loss – at this throughput, a 1% miss rate is 60K TPS of data recovery calls

We did not get replication!



aws

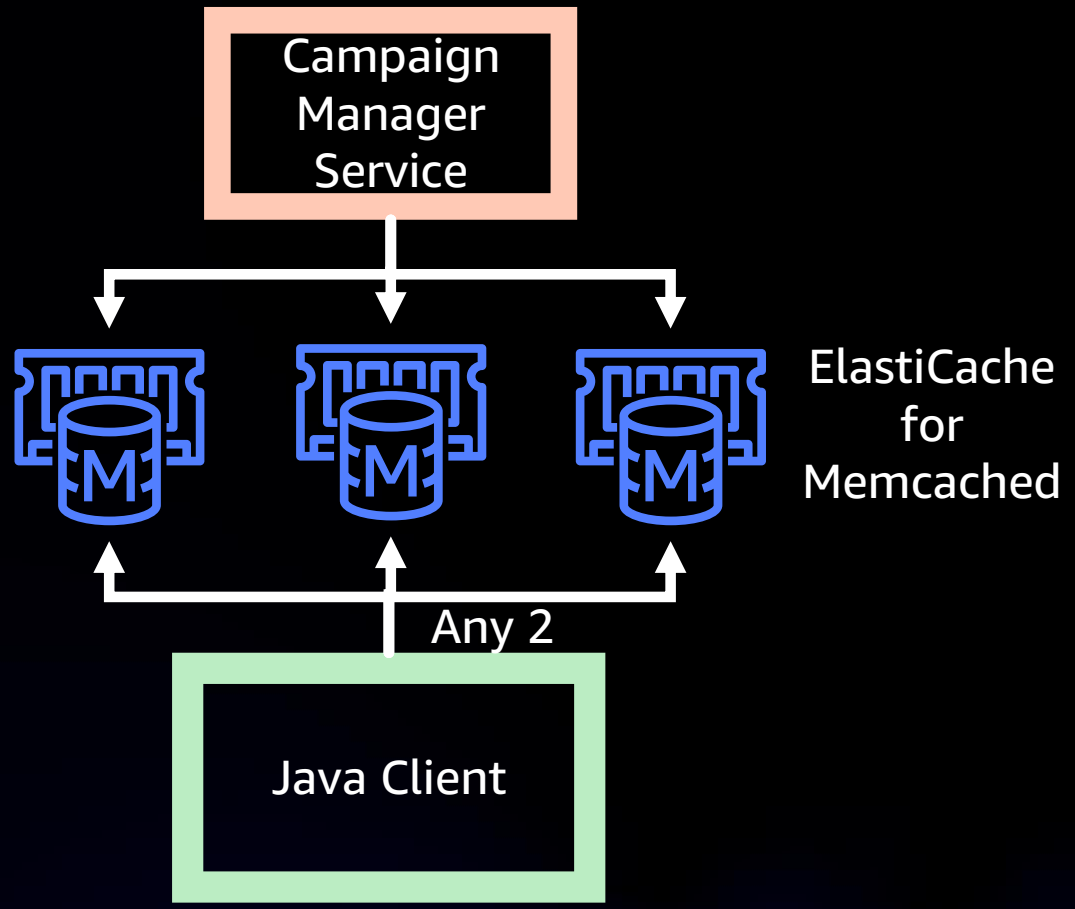
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Ad server – Caching at scale

We added data replication to ElastiCache for Memcached

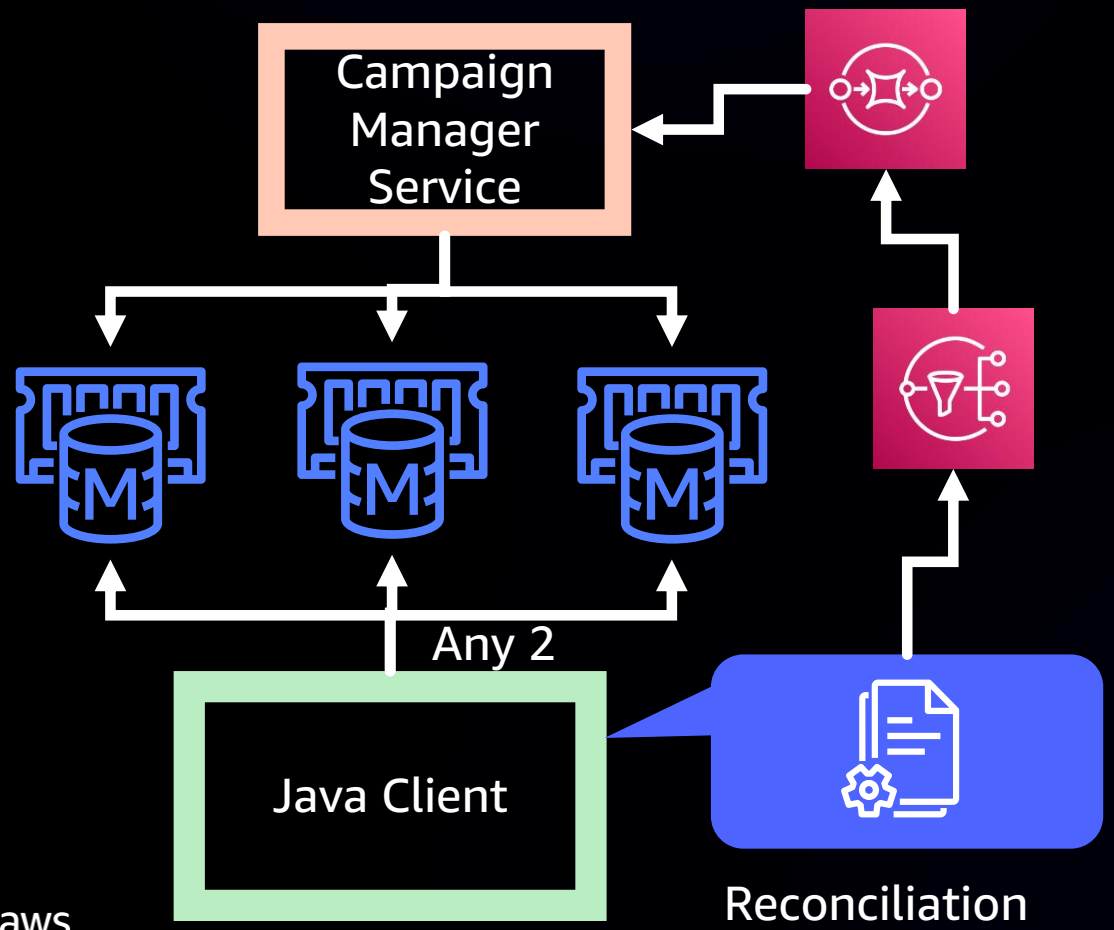
We would read from any 2 replicas and take the first response we received

Achieved higher availability!



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Ad server – Caching at scale



We needed reconciliation

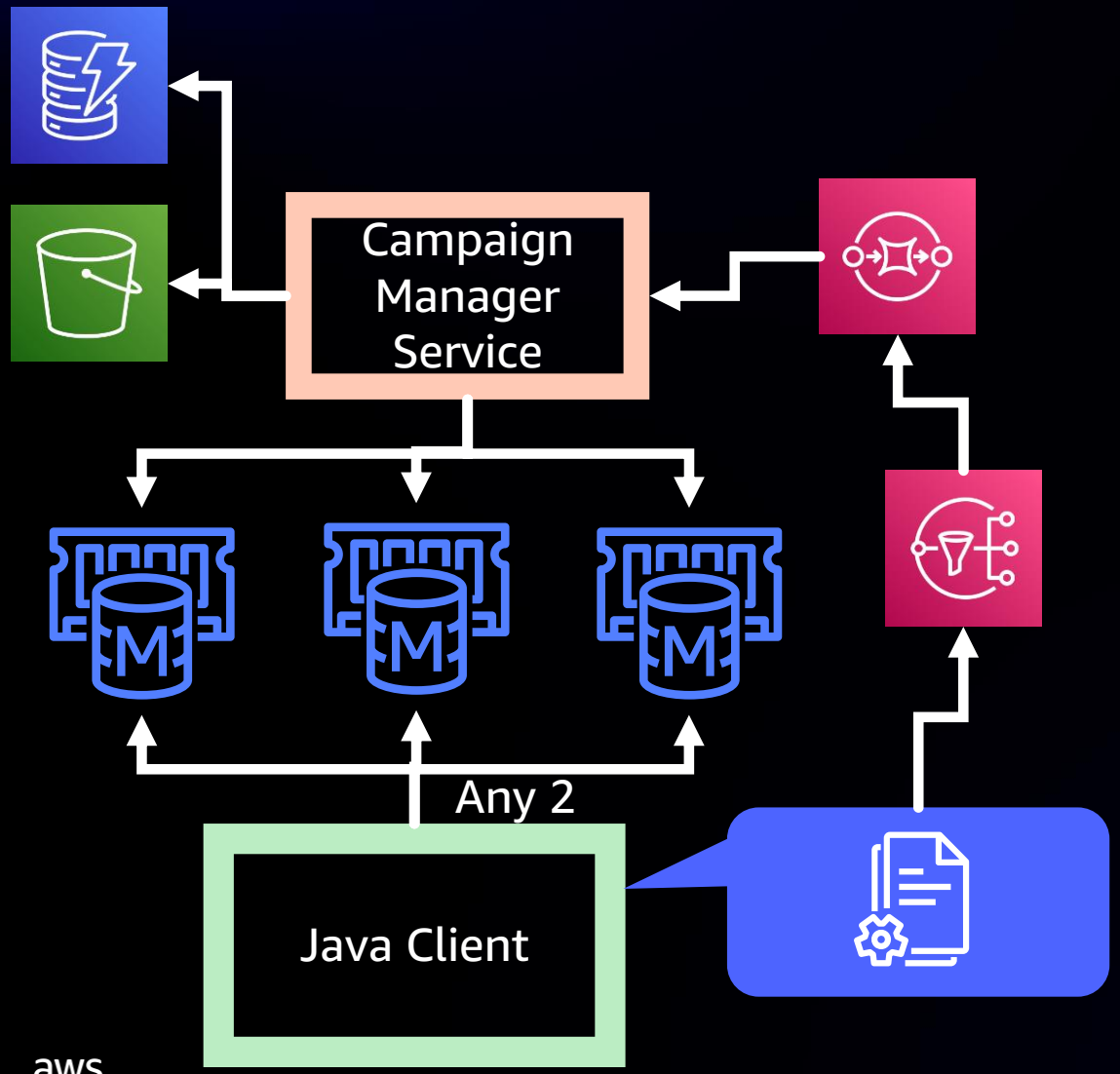
Burst traffic of 60K TPS would overload the downstream systems

Built monitoring: if a cache node fails to not cause reconciliation outage

Separated read/write fleet configs

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Ad server – Caching at scale



We needed reconciliation

Burst traffic of 60K TPS would overload the downstream systems

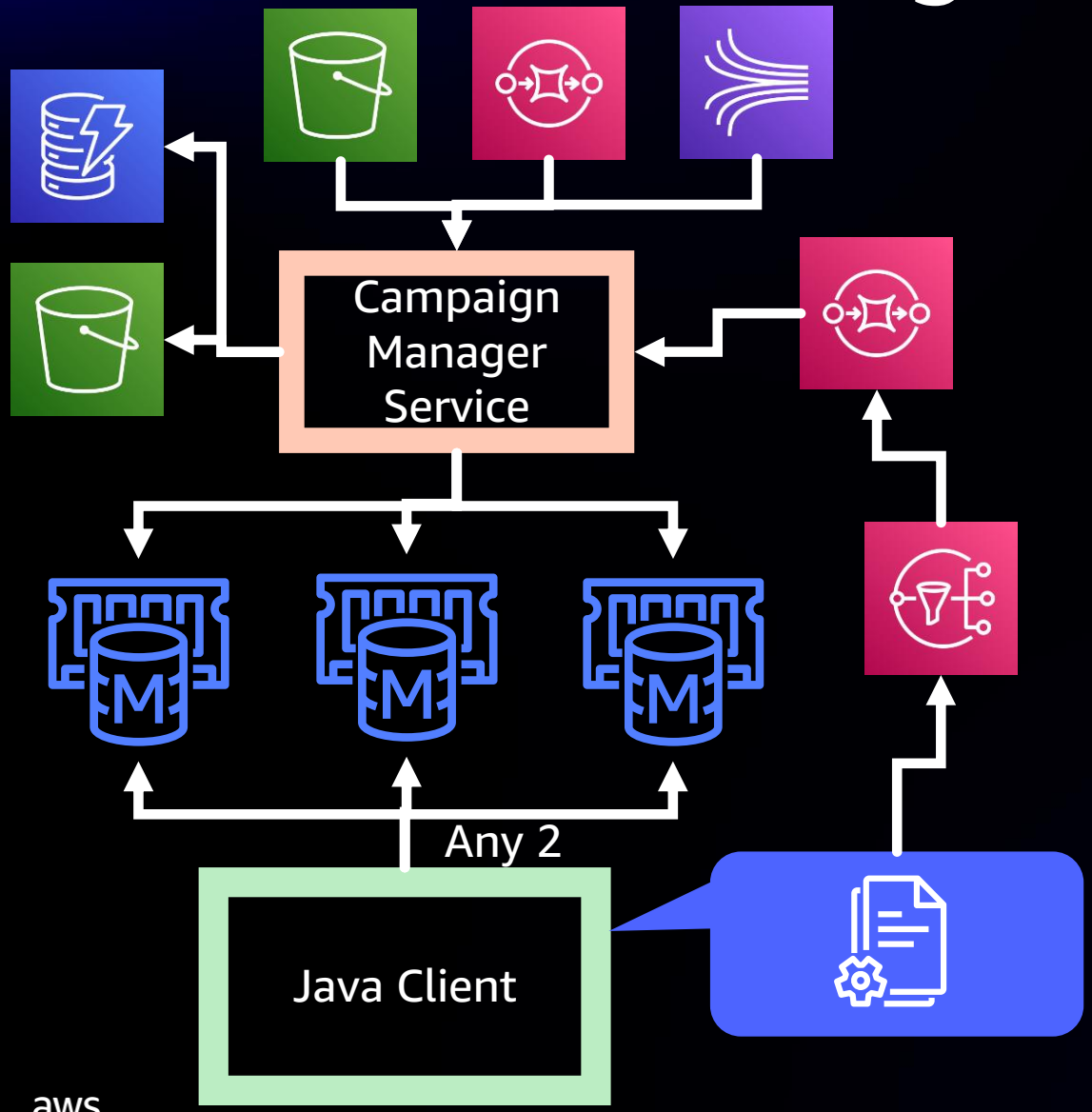
Built monitoring: If a cache node fails to not cause reconciliation outage

Separated read/write fleet configs

Complete data failure/recovery – data backup

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

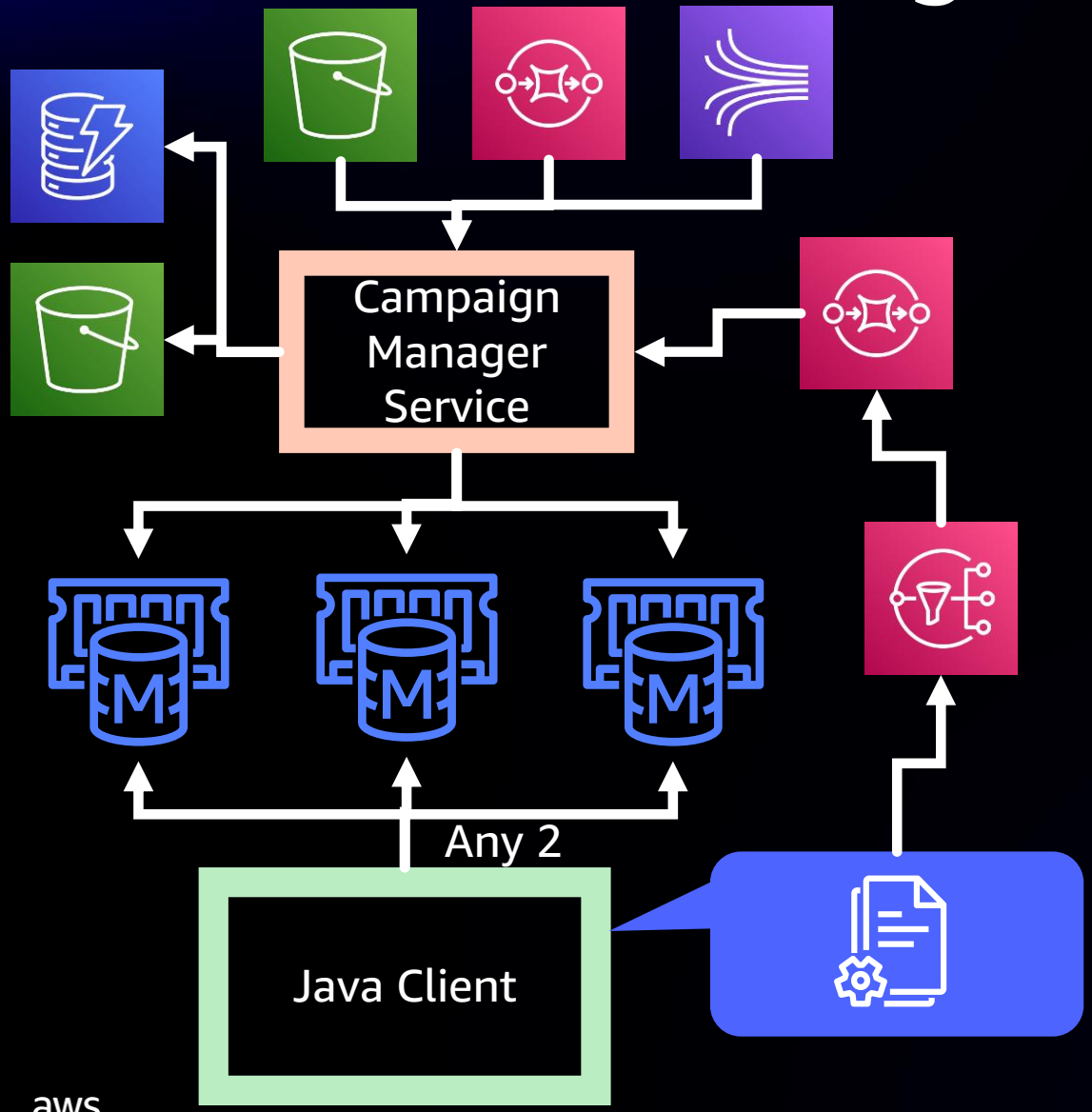
Ad server – Caching at scale



Added data ingestion through other service

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Ad server – Caching at scale



Added data ingestion through other service

High object creation causes JVM GC pauses – upgraded to ZGC in JDK 15

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Ad server – Caching at scale

- Going off-box/over the network may not be always bad – even for low-latency caches
- Caches are quite stable, with sufficient redundancy, we can achieve high data availability



- Tip: You can configure the Java client to work for consistency
- Win – Support for as large a dataset we need to store, tested for 200 hosts in a Memcached cluster ~150 TB
- Win – Support for 500M keys per second
- Win – Support for indefinite throughput; 200 hosts can support 5 Tbps

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Handling ad events at scale



Advertisers



Shoppers

Budgeting

Campaign management

Ad server



3P developers

Data lake

Ad events

Machine learning

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

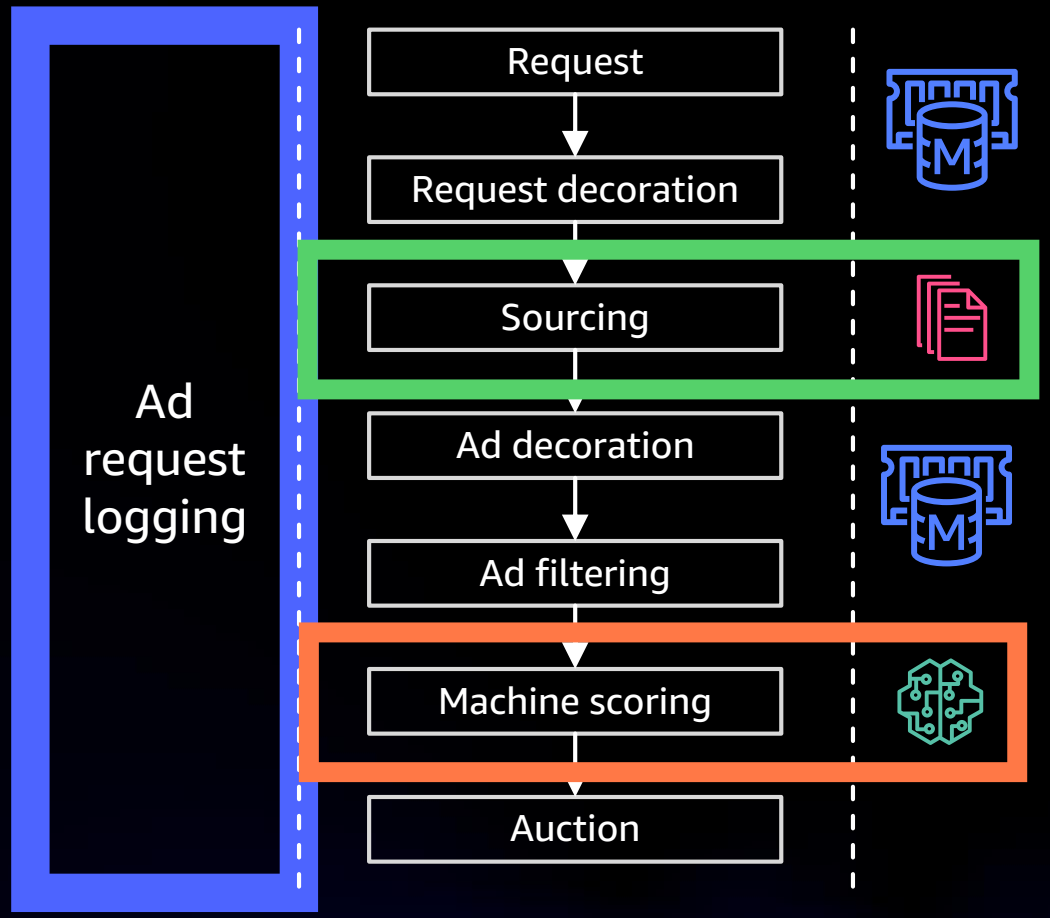
Ad events – Challenges

Up until now, the ad server was quite opaque

Real-time data was required for debugging/introspection

Machine learning systems could be more agile and responsive

100s of millions of ads with 4 TBs of data

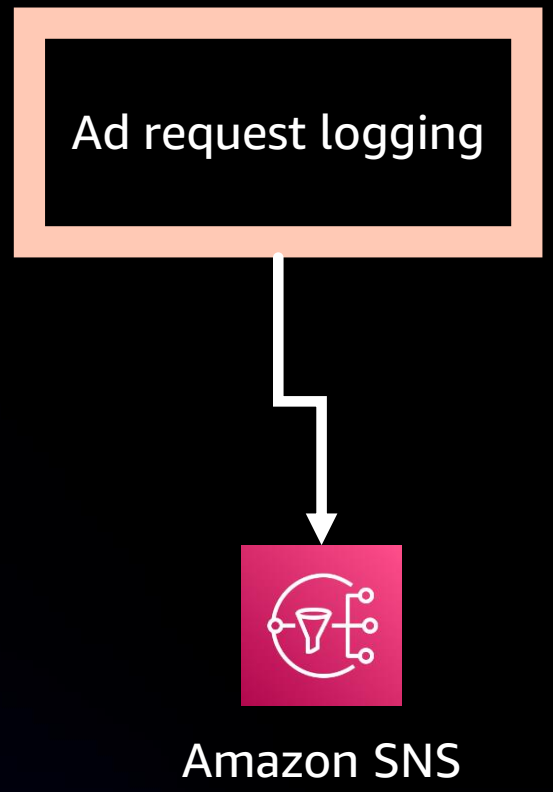


© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Ad events – Simple solution with Amazon SNS

1. Simply serialize the payload and publish to Amazon SNS

   Requests + 1,000 ads > 256 Kb SNS limit



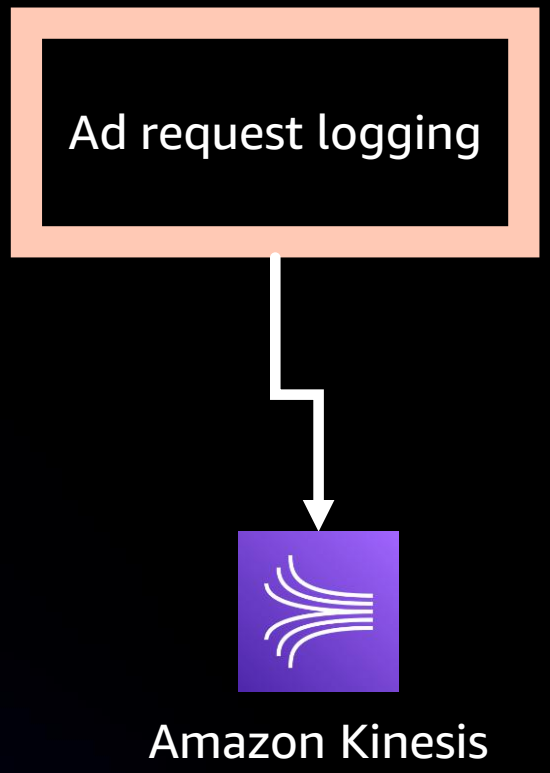
aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Ad events – Solution with Amazon Kinesis

1. Simply serialize the payload and publish to Amazon SNS
   – Requests + 1,000 ads > 256 Kb SNS limit

2. We tried Kinesis (payload could be up to 1 MB)
   – At p100, requests could have 20K ads > 1 MB
   – Saw Kinesis throttling on hot shards



aws

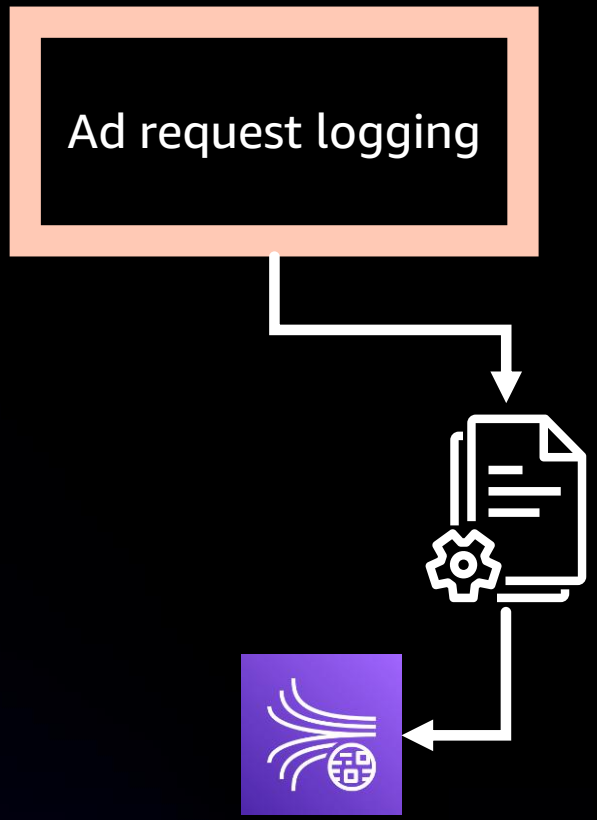
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Ad events – Solution with Amazon Kinesis

1. Simply serialize the payload and publish to Amazon SNS
   – Requests + 1,000 ads > 256 Kb SNS limit

2. We tried Kinesis (payload could be up to 1 MB)
   – At p100, requests could have 20K ads > 1 MB
   – Saw Kinesis throttling on hot shards

3. Broke payload into requests and ads
   – Publish multiple events
   – Join the events later using the RequestId

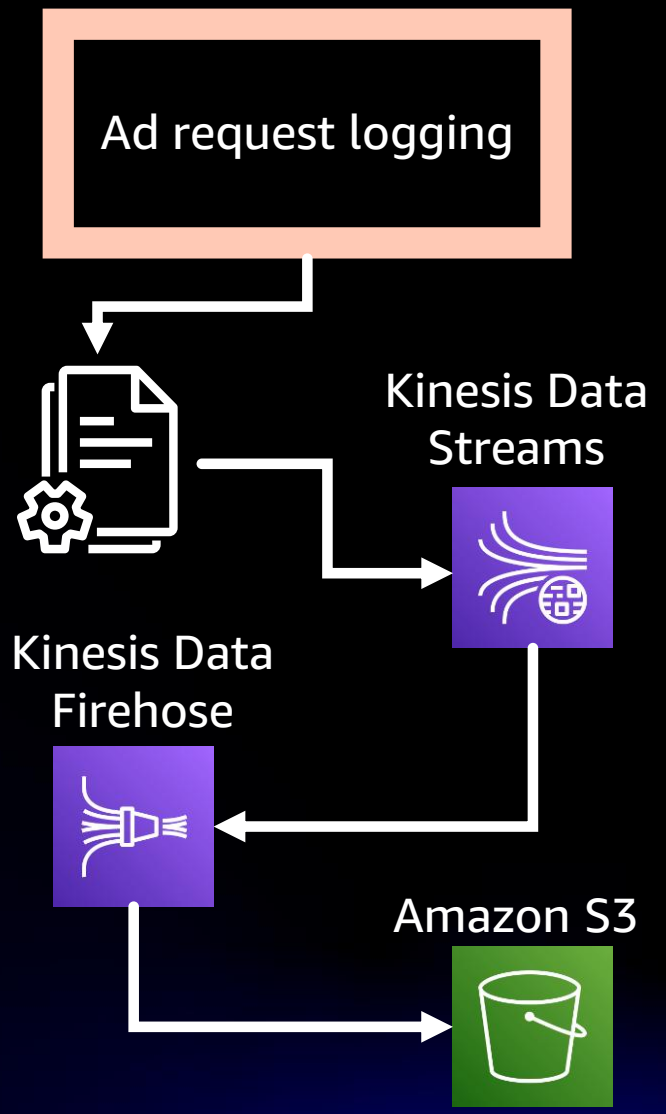


© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Ad events – Amazon Kinesis Data Firehose to Amazon S3

1. Kinesis Data Firehose got real-time logs to S3
   - We were able to log 120 TB/hour
   - Very expensive – many millions per month

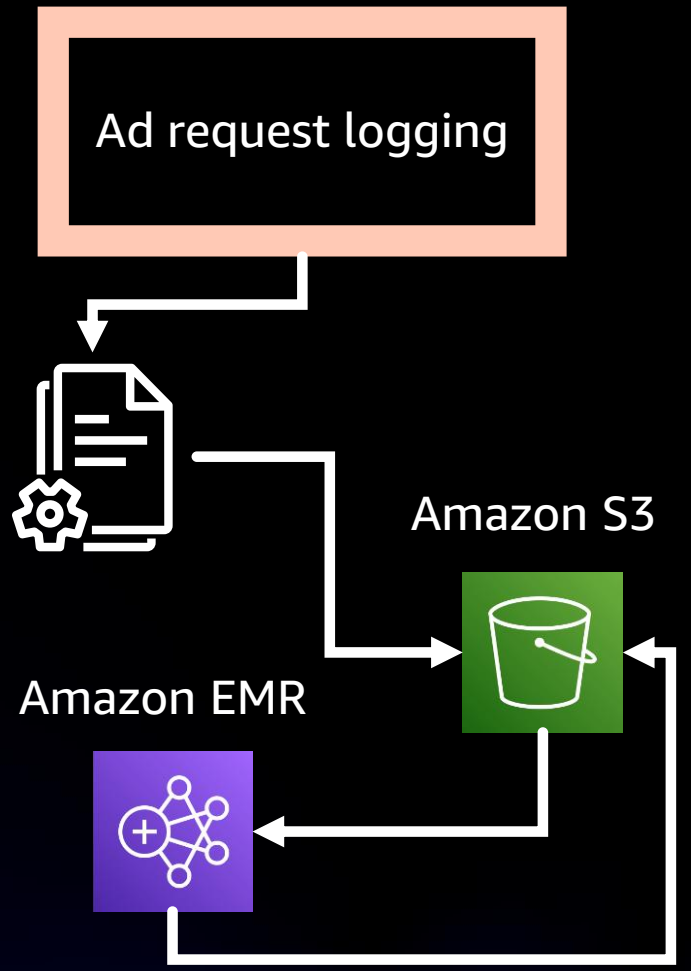
2. Realized requirements were not accurate
   - Did not need 100% of events to be logged
   - Did not need to react within seconds



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Ad events – Amazon EMR to consolidate S3 files

1. We saved 98% of costs
2. End-to-end latencies for micro Amazon S3 files are 3–5 seconds
3. End-to-end latencies to use data for analytics are now 10 minutes
4. Reduced the logging to just 1%, logging at 1.2 TB per hour



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Handling ad events at scale



Advertisers

Budgeting

Campaign management



Shoppers

Ad server



3P developers

Data lake

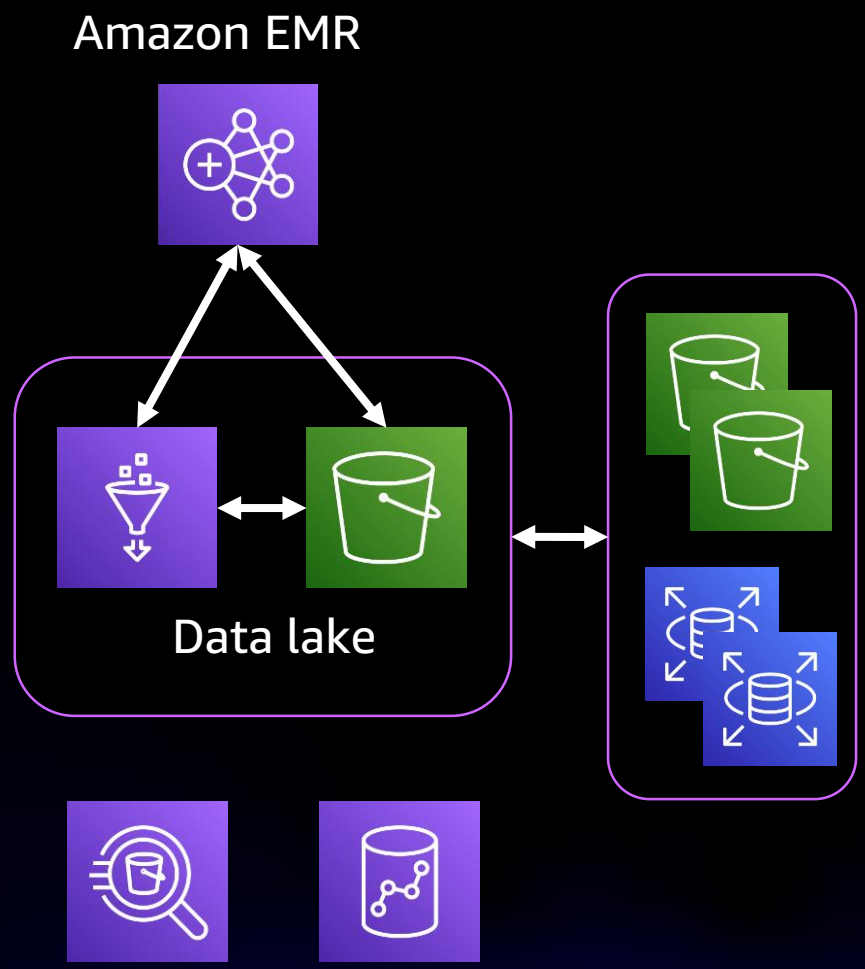
Ad events

Machine learning

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Data lake and real-time transforms

1. We aggregated Data into a centralized Data Lake
2. Data Querying using EMR and Glue
3. User Interface over Athena and Redshift



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Data lake and real-time transforms

Win: We were able to run the jobs within 4-6 hours

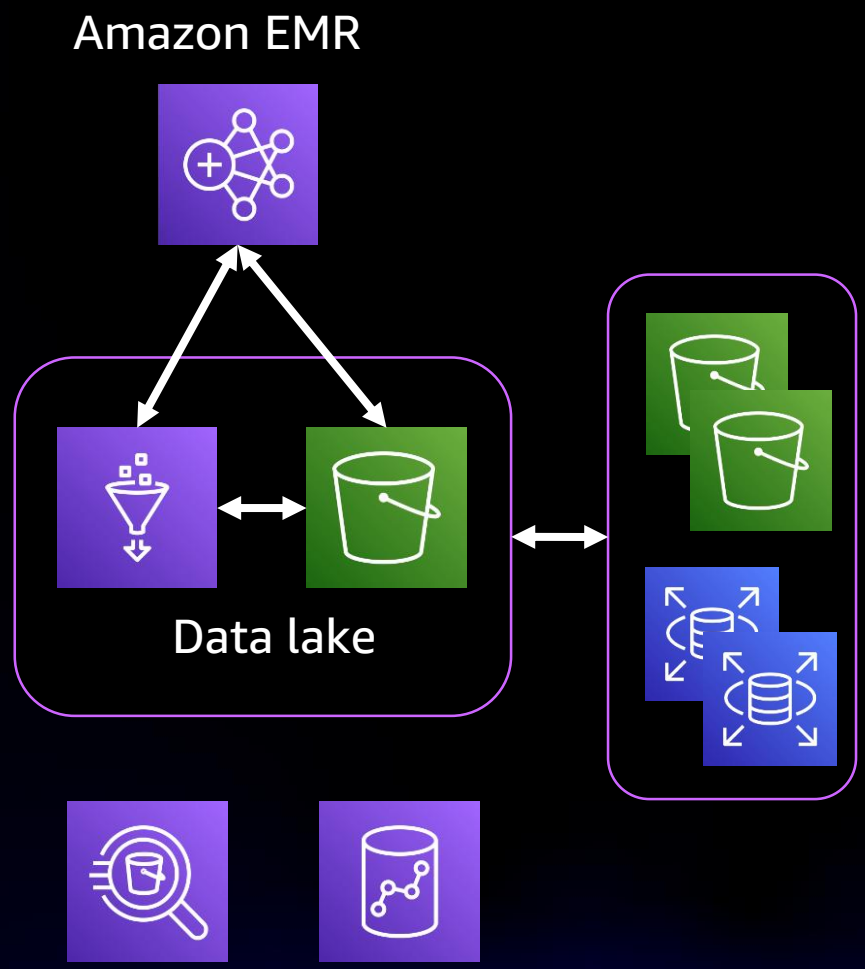
Win: Petabytes of data was available to be queried

Challenge : Still more data was needed to be processed

Challenge : Jobs were slow to react to customer behavior

Challenge : User interface could be faster

Challenge : Online offline data skew.

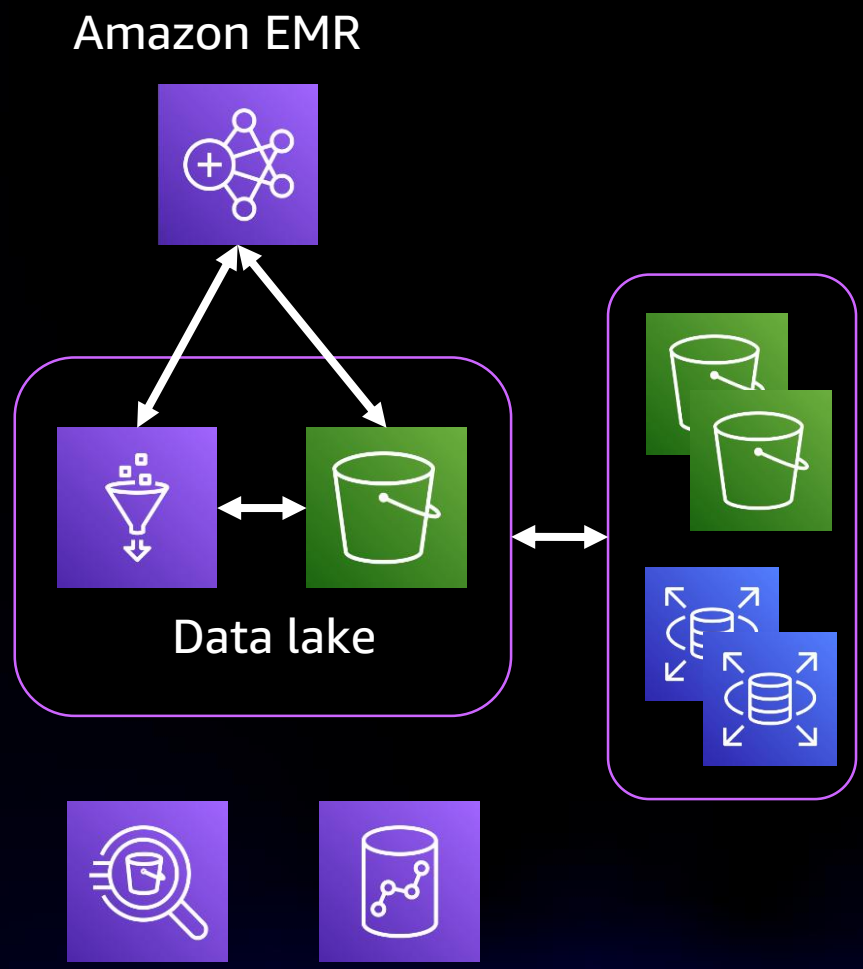


© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Data lake and real-time transforms

With more and more teams providing data to Amazon S3, we started capping Amazon S3 read/write throughput

– We sharded the Amazon S3 buckets across random prefixes 16^4 partitions



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

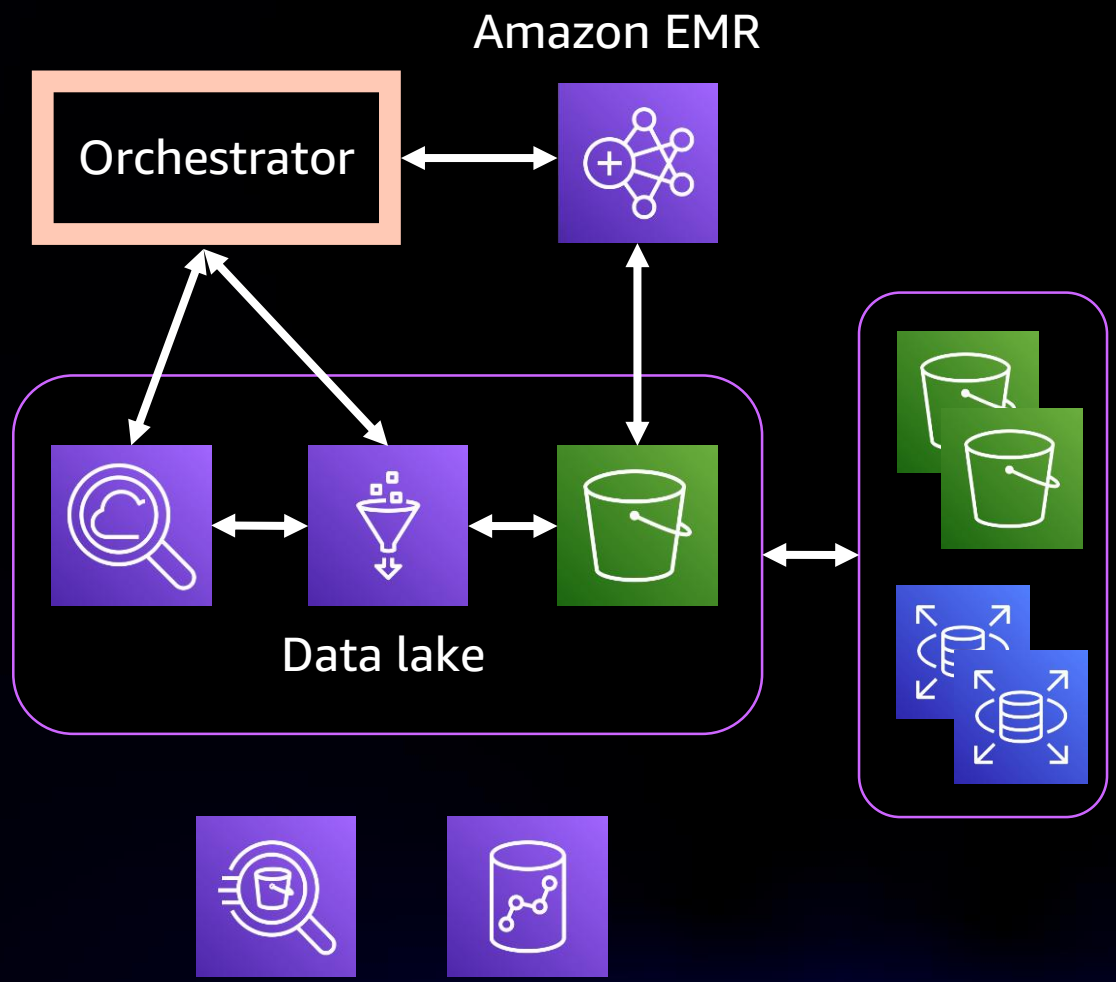
Data lake and real-time transforms

With more and more teams providing data to Amazon S3, we started capping Amazon S3 read/write throughput

– We sharded the Amazon S3 buckets across random prefixes 16^4 partitions

For very large queries, AWS Glue was slow to return the Amazon S3 partitions, slowing down queries and Amazon EMR jobs

– We created a search index using Amazon OpenSearch Service and created an orchestrating service around Amazon EMR and Amazon OpenSearch Service



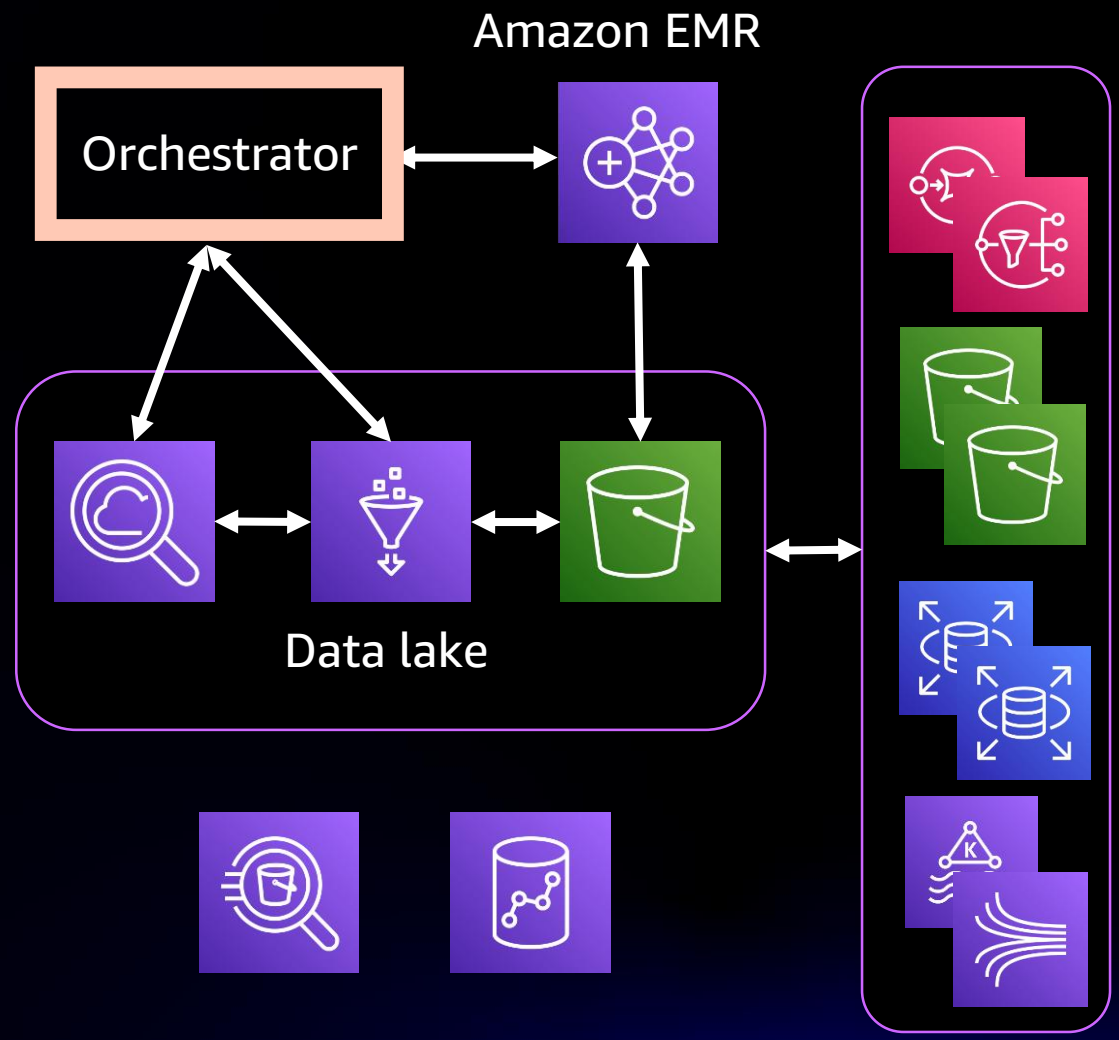
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Data lake and real-time transforms

Data processing was taking hours, primarily because these were architected as batch jobs

– We enabled stream processing using Kinesis Data Streams, Kinesis Analytics, and Kafka

– This enabled real-time data joins and data ingestion



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Data lake and real-time transforms

Data processing was taking hours, primarily because these were architected as batch jobs

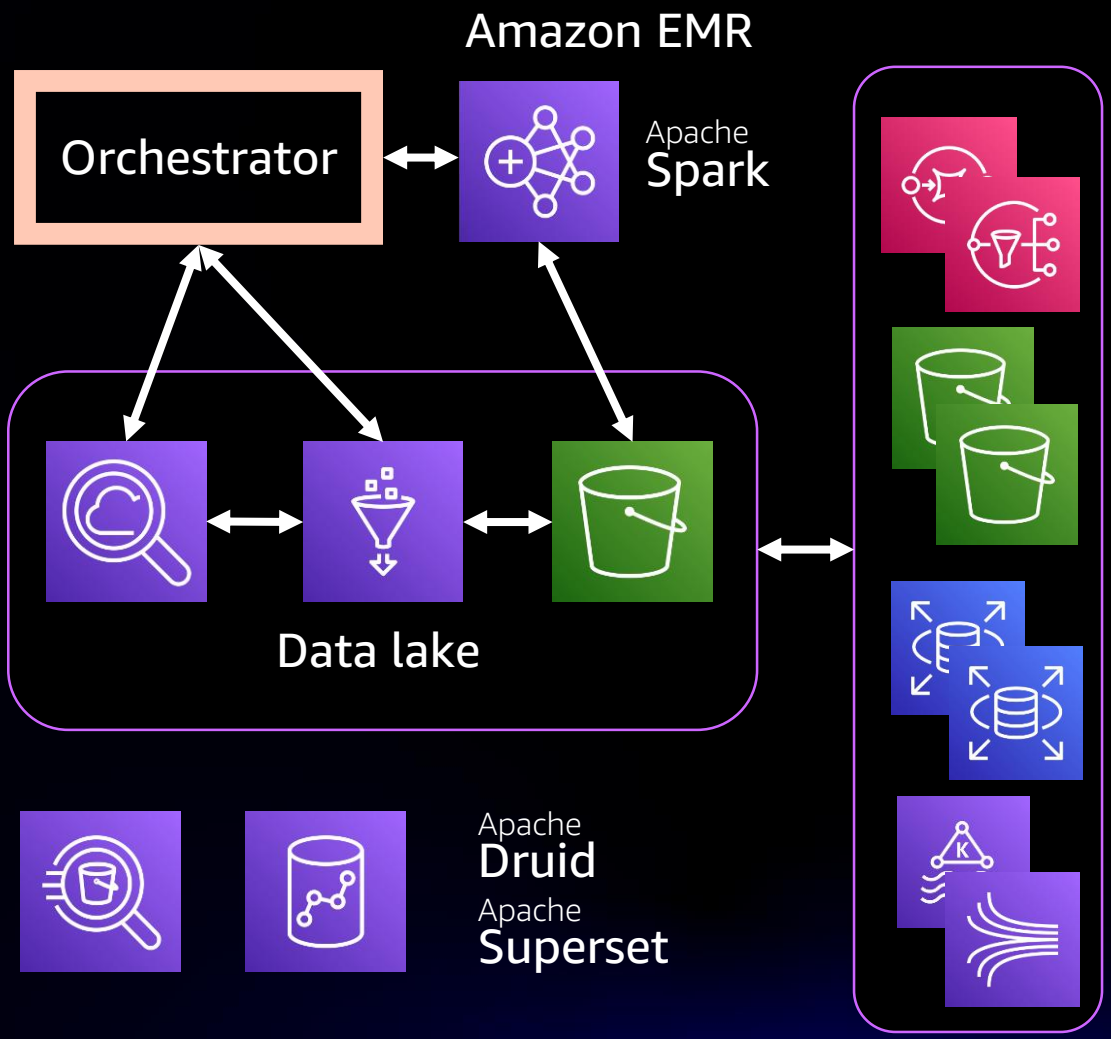
– We enabled stream processing using Kinesis Data Streams, Kinesis Analytics, and Kafka

– This enabled real-time data joins and data ingestion

Simplest of the queries would take 20–30 minutes

– We enabled Amazon EMR through Apache Spark, which fastened data processing

– We indexed most recent 1 week of data in Druid/Superset for faster queries

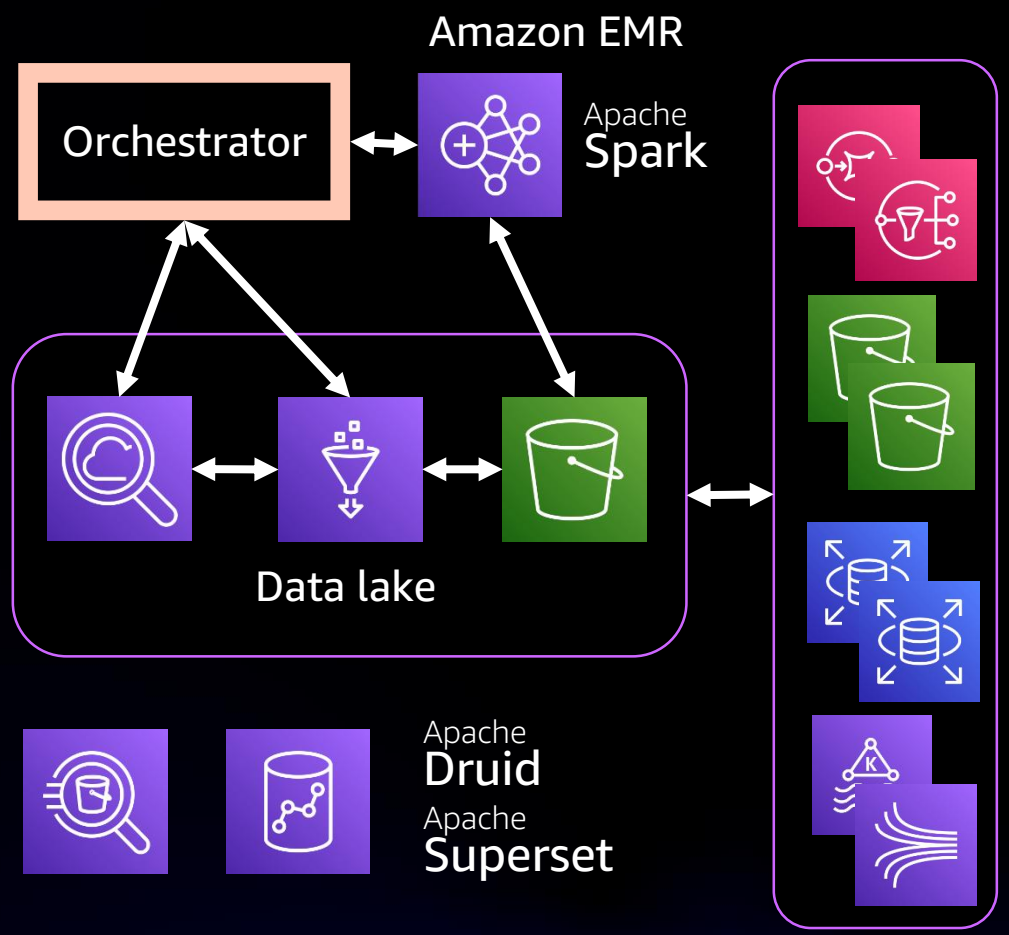


aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Data lake and real-time transforms – Success

1. Now storing hundreds of petabytes of data
2. Can offer data availability within 15 minutes
3. Real-time data joins and analytics are enabled
4. Uses Amazon S3 tiers to reduce storage costs
5. Reuses Amazon EMR clusters to optimize costs

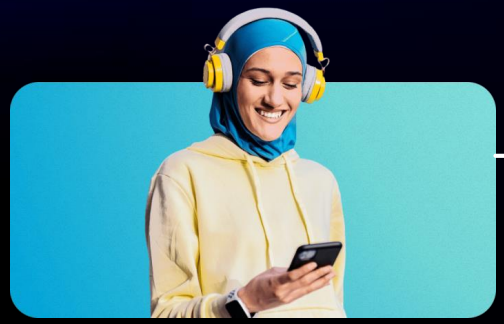


© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Solving ads challenges with machine learning

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Overview of ML challenges in ads



Shoppers

Sourcing

Model training

Logging

Model publishing

Shopper response prediction

Allocation and pricing

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Focus of the talk



Build ultra-low-latency, high-throughput ML inferencing systems

1. Score thousands of ads within 20 ms
2. Host hundreds of models with different hardware/software optimizations
3. Keep hardware costs under control

hundreds of K	Peak TPS and growing
hundreds+	Deep learning (DL) models
hundreds of M	Ads scored per second
20 ms at p99	Model inferencing latency

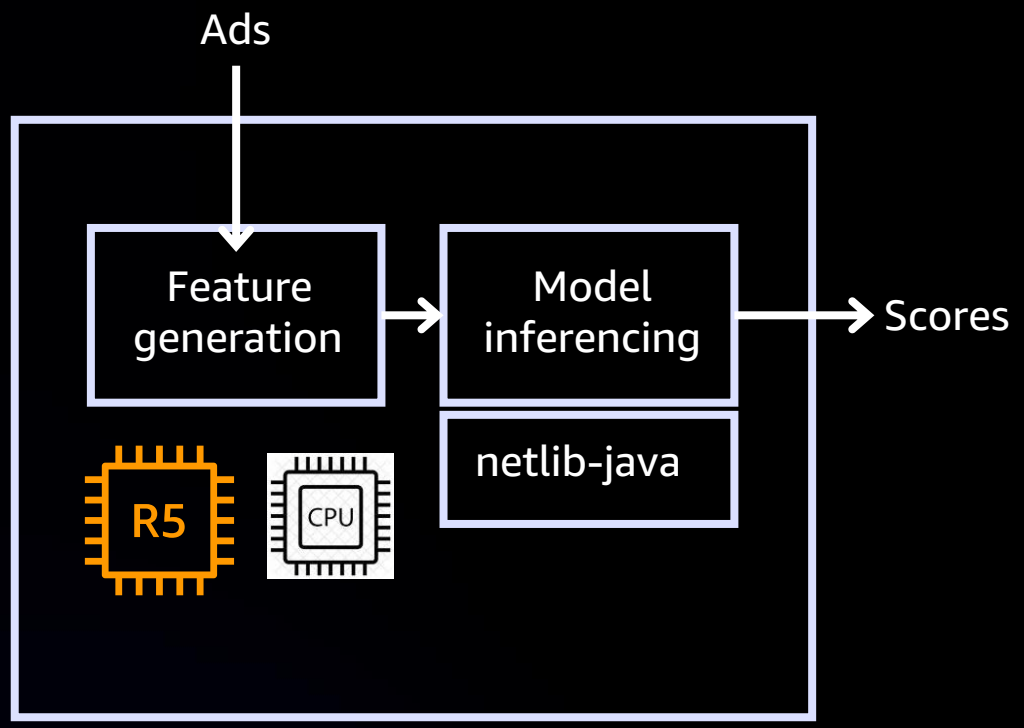
aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Deep learning inferencing architecture

Real-time inferencing

1. BLAS (basic linear algebra subprograms)
2. Multi-layer perceptron (MLP) networks (3–5 layers, <100M learnable parameters)
3. 100% traffic coverage within 20 ms at p99
4. Cannot scale to more complex deep learning models (attention, transformers, etc.)

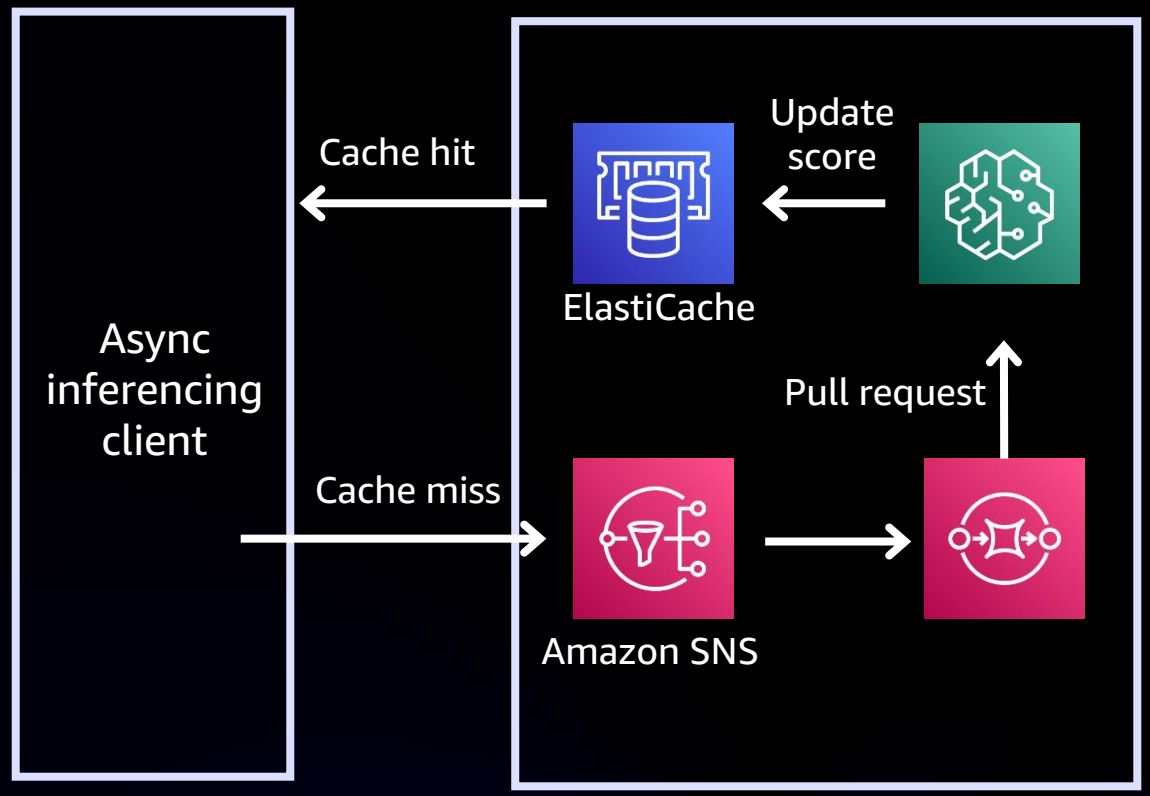


© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Deep learning inferencing architecture

Asynchronous inferencing

1. Pre-compute scores for <query, Ad> pairs for online cache lookup
2. Fall back to a simpler model when cache misses, and asynchronously update cache
3. Support complex deep learning models that cannot be inferenced in real time
4. Constrained by cardinality of keys, effective for head and torso traffic

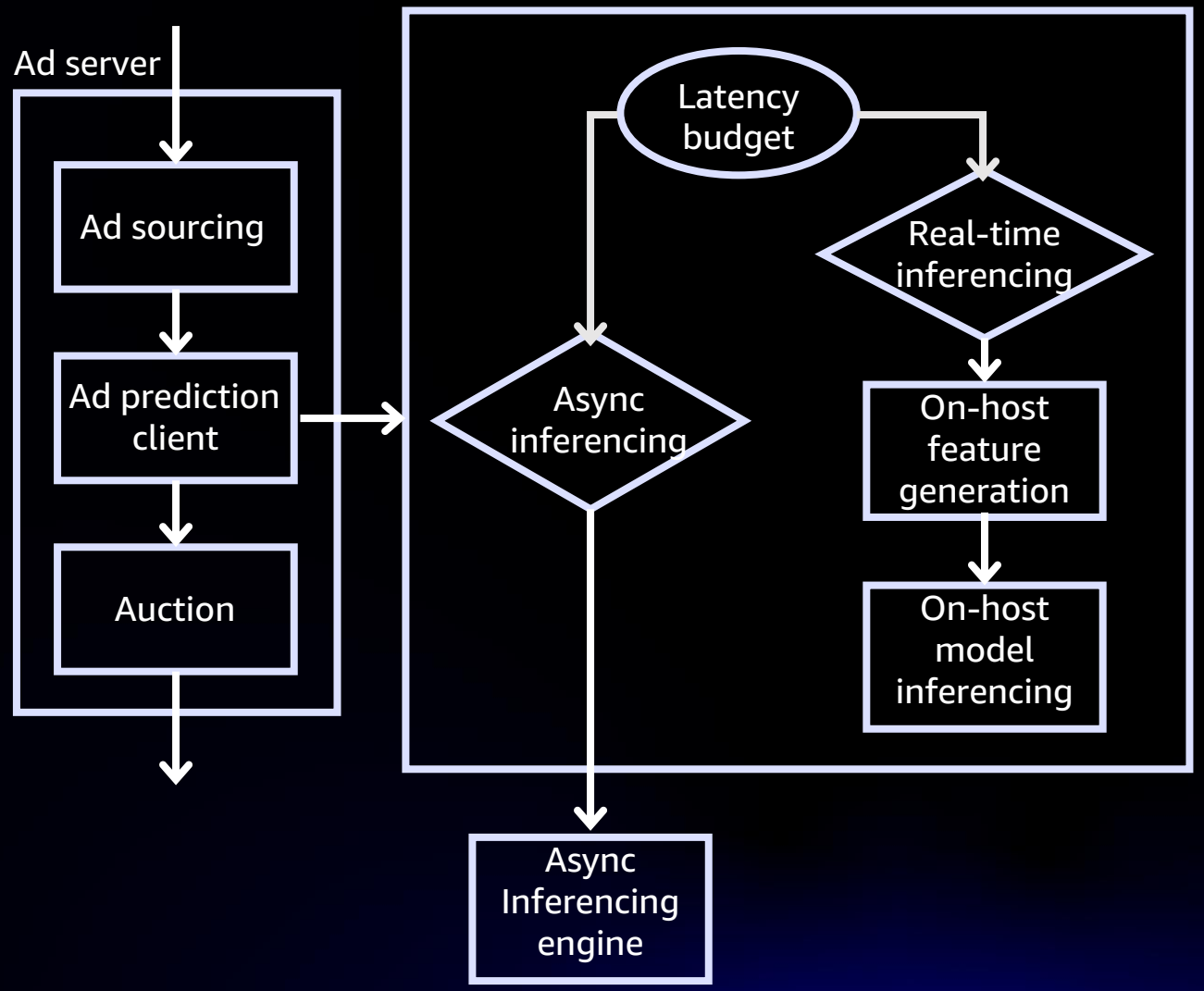


© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Deep learning inferencing – Put it all together

Hybrid serving infra

1. CPU-based architecture on AWS
2. Real-time and asynchronous model inferencing
3. Latency-aware model and serving infra selection



aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

2021 new challenges

Need to scale to **significantly more complex** deep learning models

1. Prohibitively high inference latency for real-time predictions on CPUs

2. Prohibitively high key cardinality for asynchronous inferencing

10X parameters — More learnable parameters (hundreds of millions to billions)

15X FLOPs — More floating point operations per second

50X latency — Higher latency per inference on r5.24xlarge (20 ms to 1,000 ms)

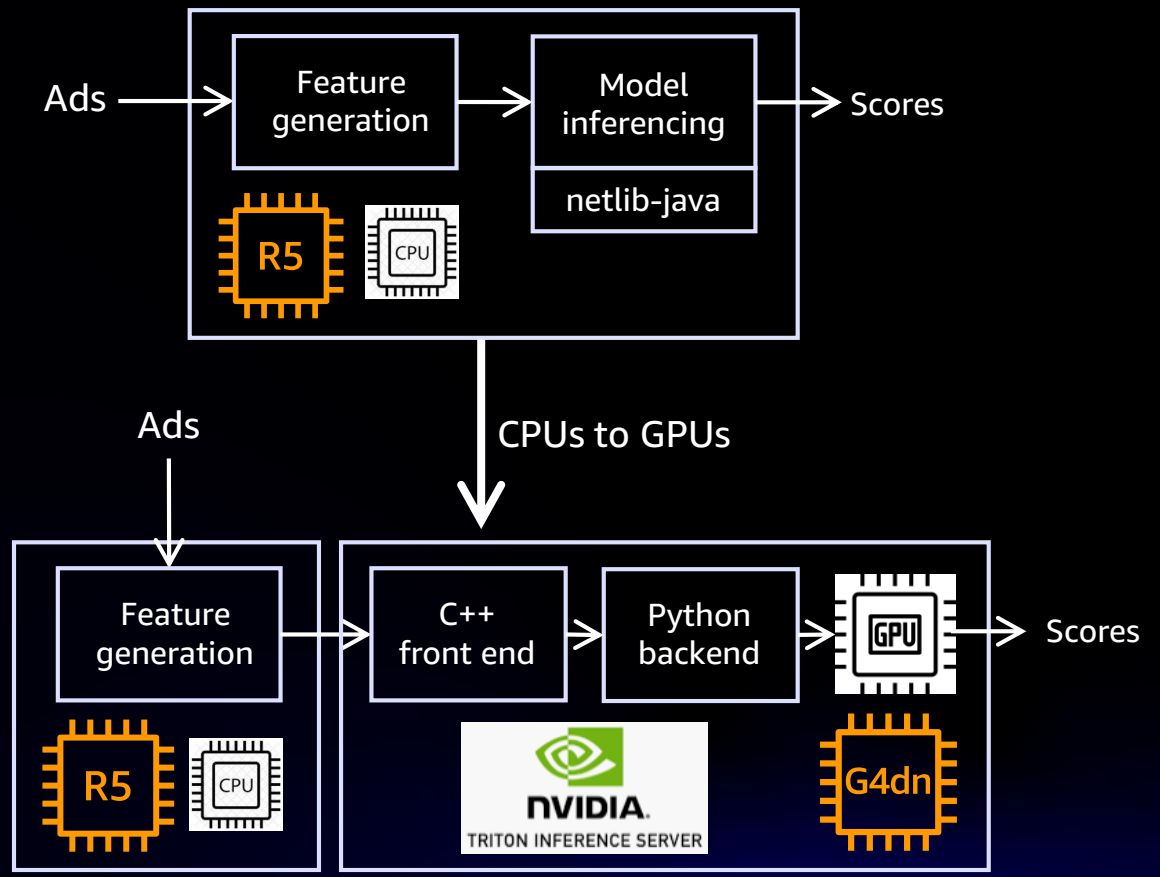
aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

GPU-based inferencing

GPU inferencing on Amazon ECS

- Integrated with NVIDIA Triton model server with Apache TVM deep learning compiler
- Optimize the tail latency
- Compress the payload
- Tune model server parallelism
- Reduce CUDA driver overhead



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

2021 new challenges



Need to scale to **many more** deep learning models

1. Horizontally scalable: a monolithic service cannot scale to many models

2. Customized optimization: each model has its own hardware (CPUs, GPUs) and software (e.g., Amazon SageMaker multi-model server, NVIDIA Triton) optimization needs

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

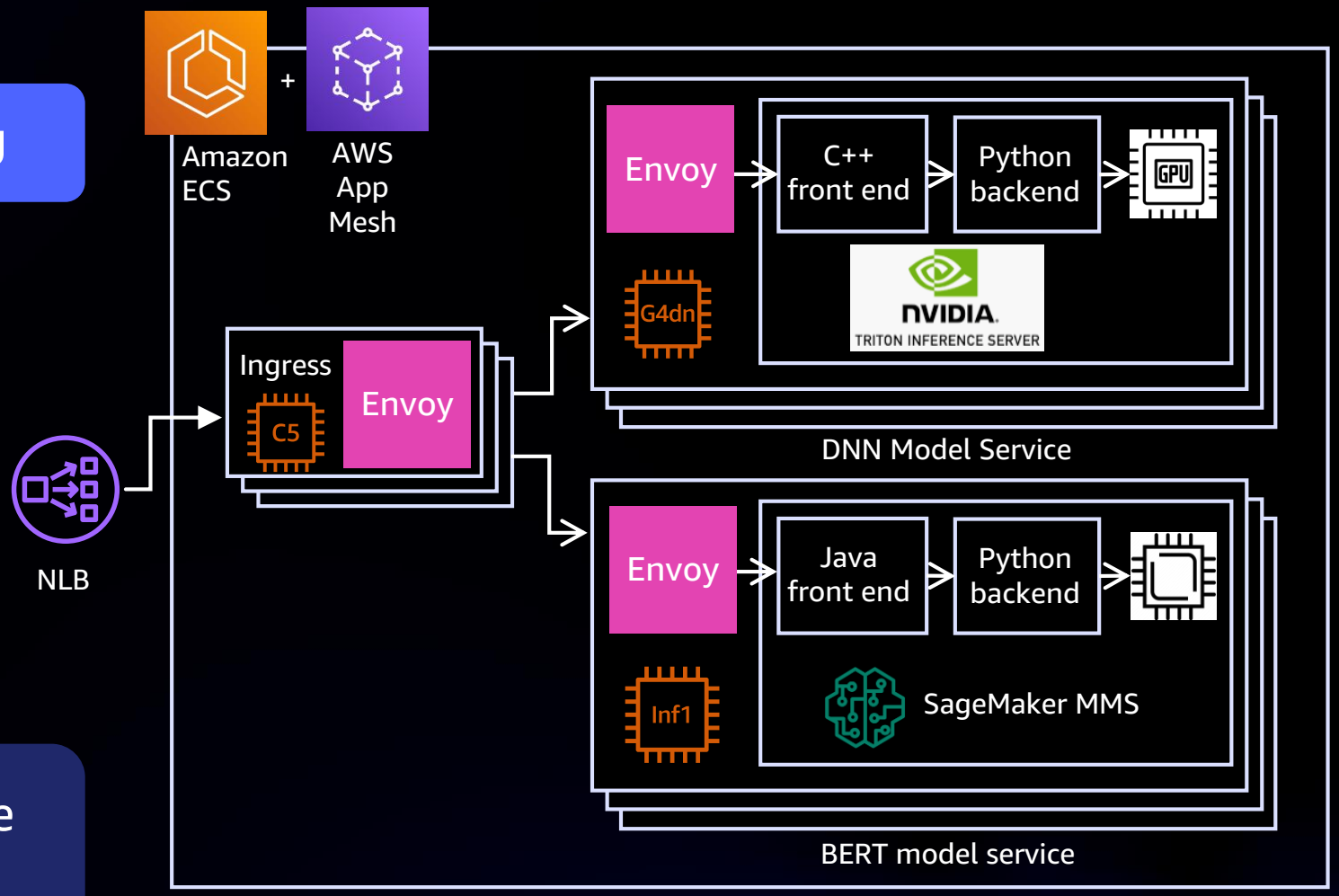
Microservice inferencing architecture

A new data plane to scale model hosting

Models partitioned an served through microservices

Requests routing via AWS App Mesh

Each service can have different hardware and software optimizations



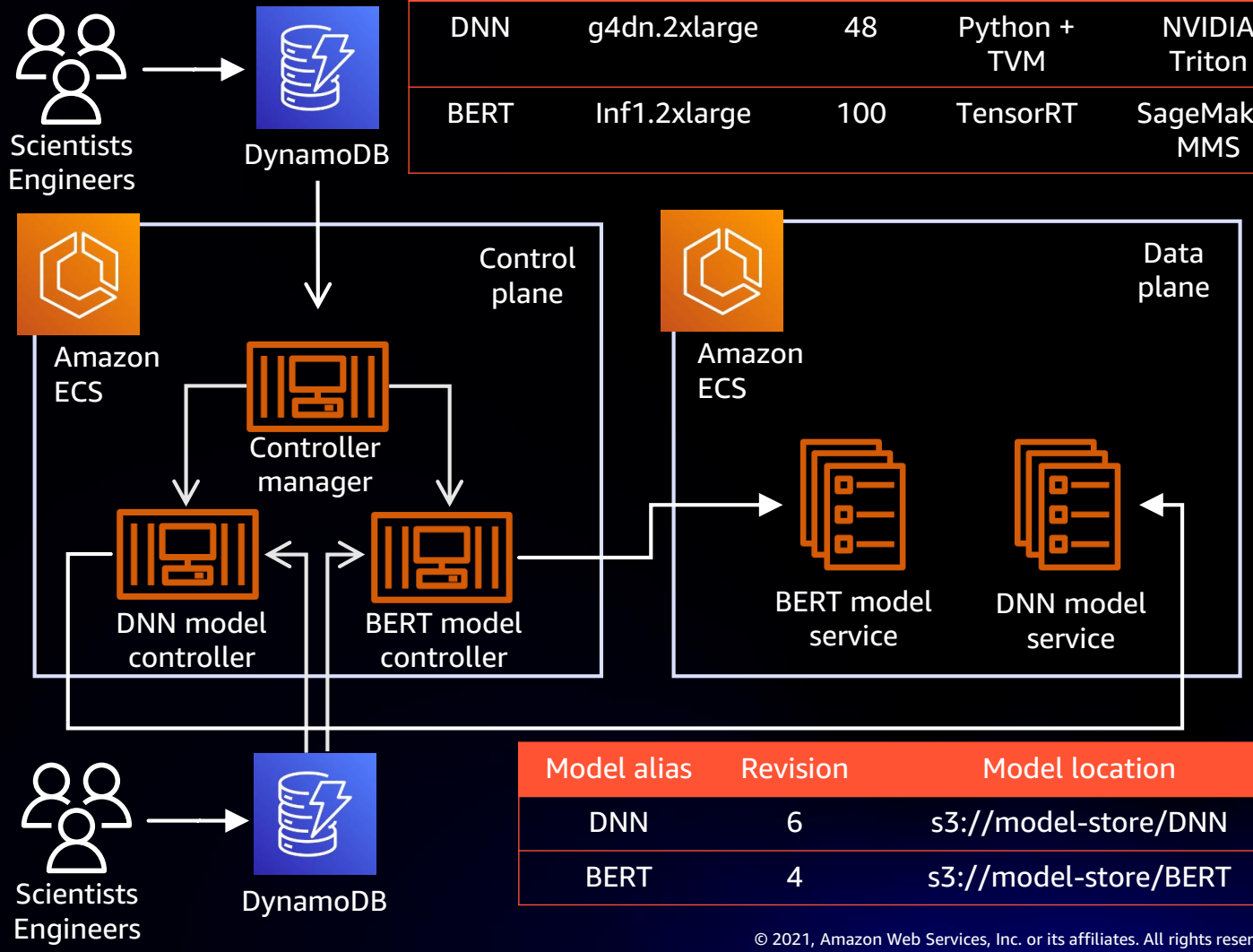
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Microservice inferencing architecture

A new control plane to enable config-driven model, service, fleet management

Feedback control loop to monitor latest state of data plane, and make changes when needed

Model alias	Instance type	Replica	Runtime	Model server
DNN	g4dn.2xlarge	48	Python + TVM	NVIDIA Triton
BERT	Inf1.2xlarge	100	TensorRT	SageMaker MMS



Model alias	Revision	Model location
DNN	6	s3://model-store/DNN
BERT	4	s3://model-store/BERT

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Learnings on building ML inferencing systems



One solution does not fit all

- Dynamic latency budget estimator
- Open architecture with customized hardware and software optimizations



Inferencing optimization is full-stack

- Model server: AWS MMS, NVIDIA Triton
- DL optimization software/library: cuBLAS, TensorRT, Apache TVM
- Hardware: R5, P3, G4dn

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Thank you!

Ish Singh
isingheg@amazon.com

Kun Liu
kliu@amazon.com

Anuj Joshi
januj@amazon.com

Rachit Chawla
rachit@amazon.com

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.