
AWS
re:Invent

RET401

How Stitch Fix is delivering personalized experiences

Madhu Nunna
Solutions Architect
AWS

Ujjwal Sarin
Data Platform Engineer
Stitch Fix

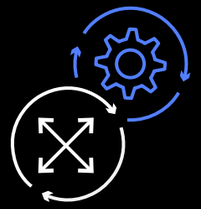
AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

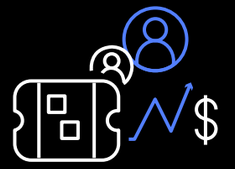
Supporting retail and consumer brand transformation



Transform your engagement



Optimize your operations



Increase your insights

Personalization offers material business benefits

EVERYBODY'S TASTES, PREFERENCES, AND NEEDS ARE DIFFERENT



Help customers discover products and services they need



Improve customer engagement with your website, app, and content



Increase conversion—purchases, subscriptions, app downloads, movie & music streams



Increase revenue for your business

Considerations for personalized recommendations

Data sources for personalization

Timeliness of feedback to customers

Continuous learning from customer behavior



Technology enablers for personalization on AWS

Predictive user engagement

(Little to no ML experience required)

- Amazon Personalize
- Amazon Pinpoint
- Amazon Neptune

IP-backed recommendation engines

(In-house ML capabilities are core differentiator)

- AWS ML portfolio (Amazon SageMaker)
- Serverless data stores (such as Amazon DynamoDB and Amazon Aurora serverless)
- In-memory cache (Amazon ElastiCache)
- Amazon CloudFront CDN
- AWS Global Accelerator
- Amazon Route 53

Are you well architected?



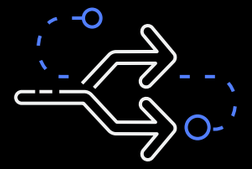
Performance



Reliability



Security



Continuous improvement



Cost efficient

Stitch Fix



Personal
Styling For
Everybody

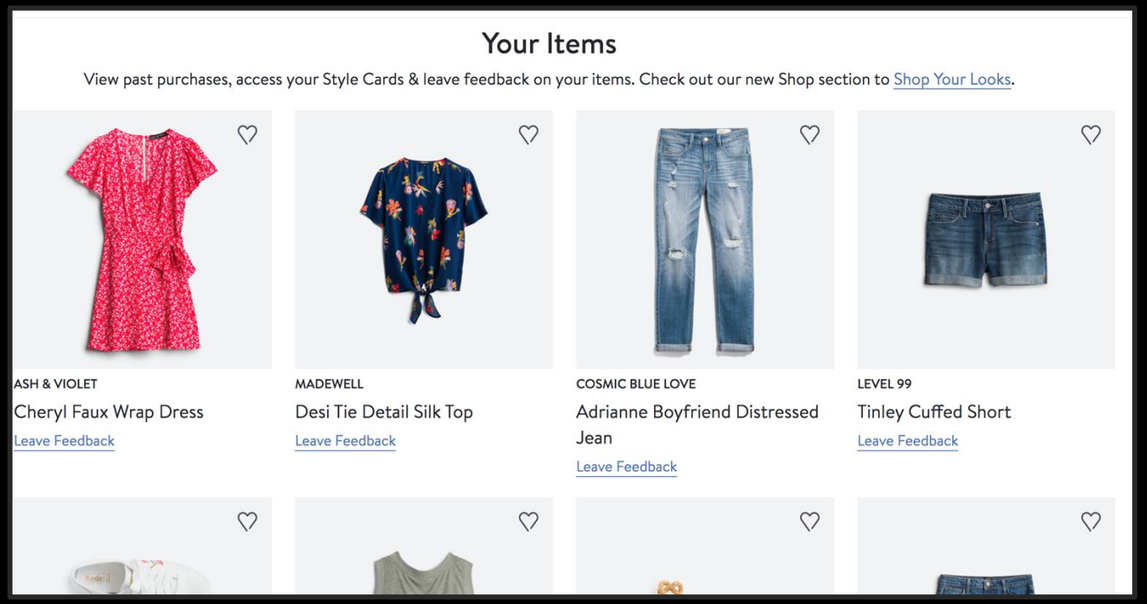
STITCH FIX

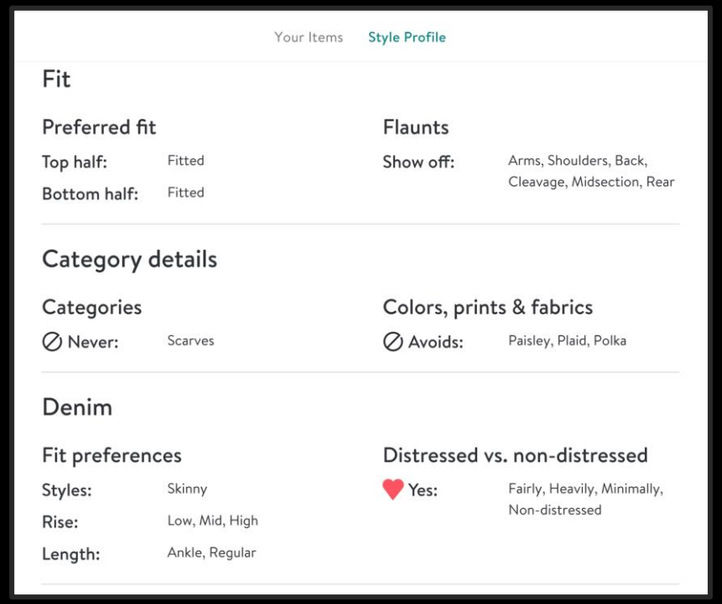


How does data work at Stitch Fix?

At Stitch Fix, we gather data from clients' Style Profiles, their Style Shuffle answers, the items they've bought



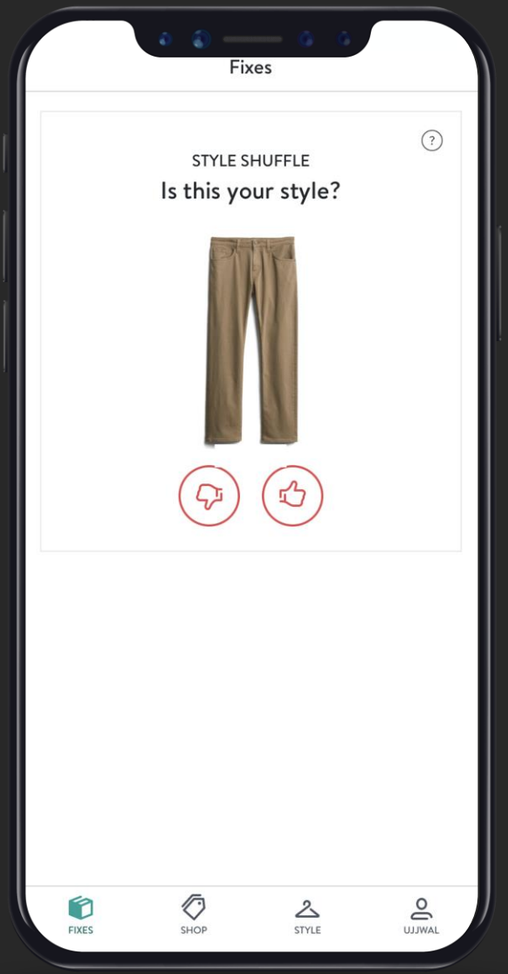


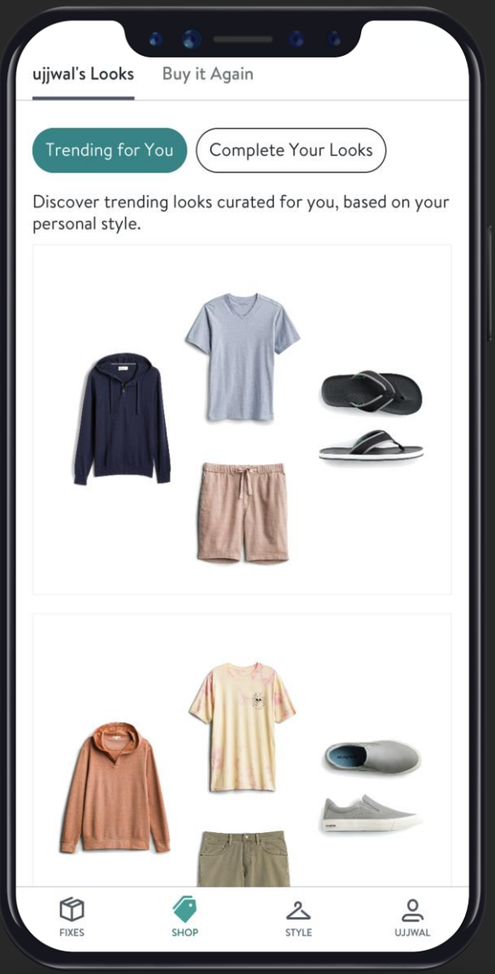


Business context

Responding to changing business context

1. Collect and save user responses in near real-time to serve recommendations and retrain the underlying recommendation model
2. Serve recent and historical data with a p99 of 80ms
3. Horizontally scalable as traffic and data grows
4. Support for data deletion and GDPR compliance
5. Low operational and cost overhead





Prior solution

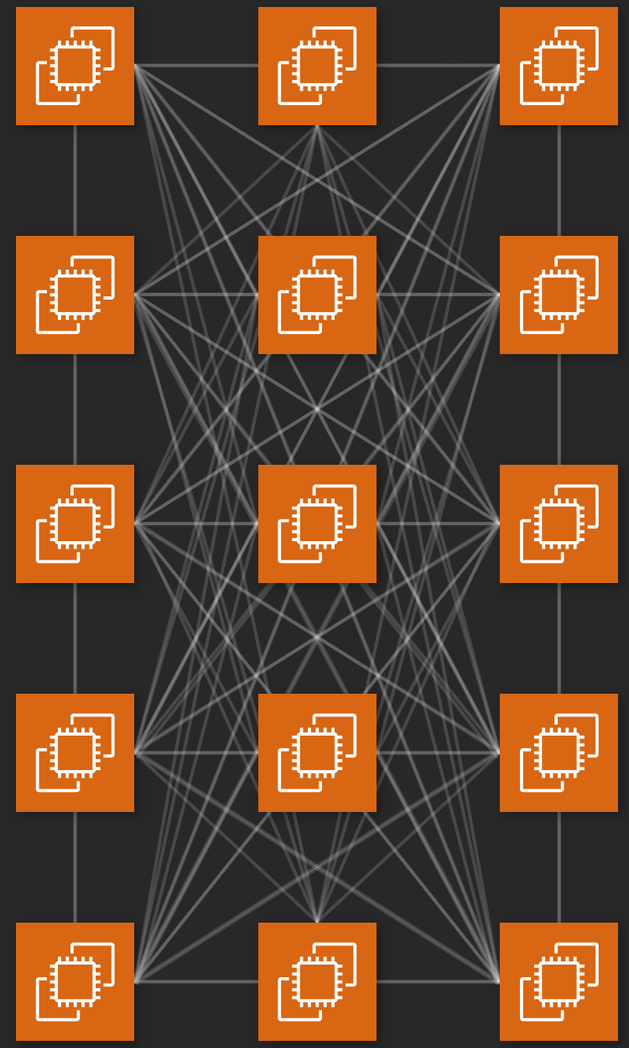
120 node Redis Cluster (Amazon ElastiCache) to hold the data in the hot tier

Challenges

Problems adding new nodes—rebalancing time, limit to how far the cluster can grow, and costs

Redis' internal representation is all strings, overhead was high

Memory-backed solution did not support disaster recovery



Requirements



Latency

- P99 latency for 3000 most recent responses per user less than 80ms
- Data keyed by client-id



Recency

- P99 update interval from event availability time (off Kafka) to being available in the datastore: less than 1 seconds



Recoverability & fault tolerance

- Ability to quickly rehydrate entire datastore from Hive tables or backups in case of a catastrophic event
- Handle network partitioning or partial node failures



Scalability & cost

- Cluster scales horizontally as read/write traffic and data grows
- Costs less than $1000/day at current traffic levels

Methodology

Abstract implementation from interface, to be able to try out different implementation strategies (DB backends)

Build a harness to measure latency, error rate, and throughput

Identify a dataset to mimic live traffic



Simulate live traffic



Test harness



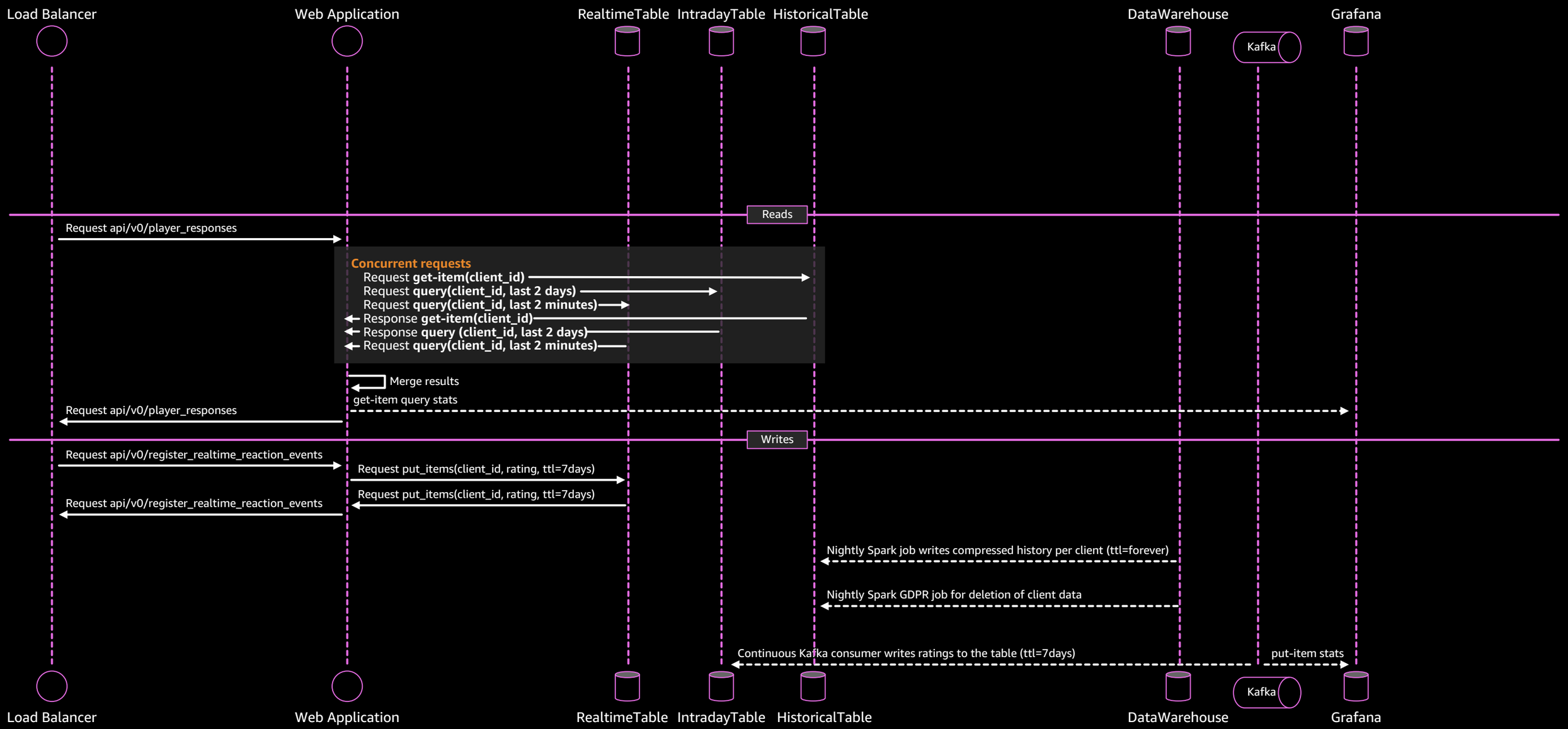
Measure

Latency | Error rate | Throughput

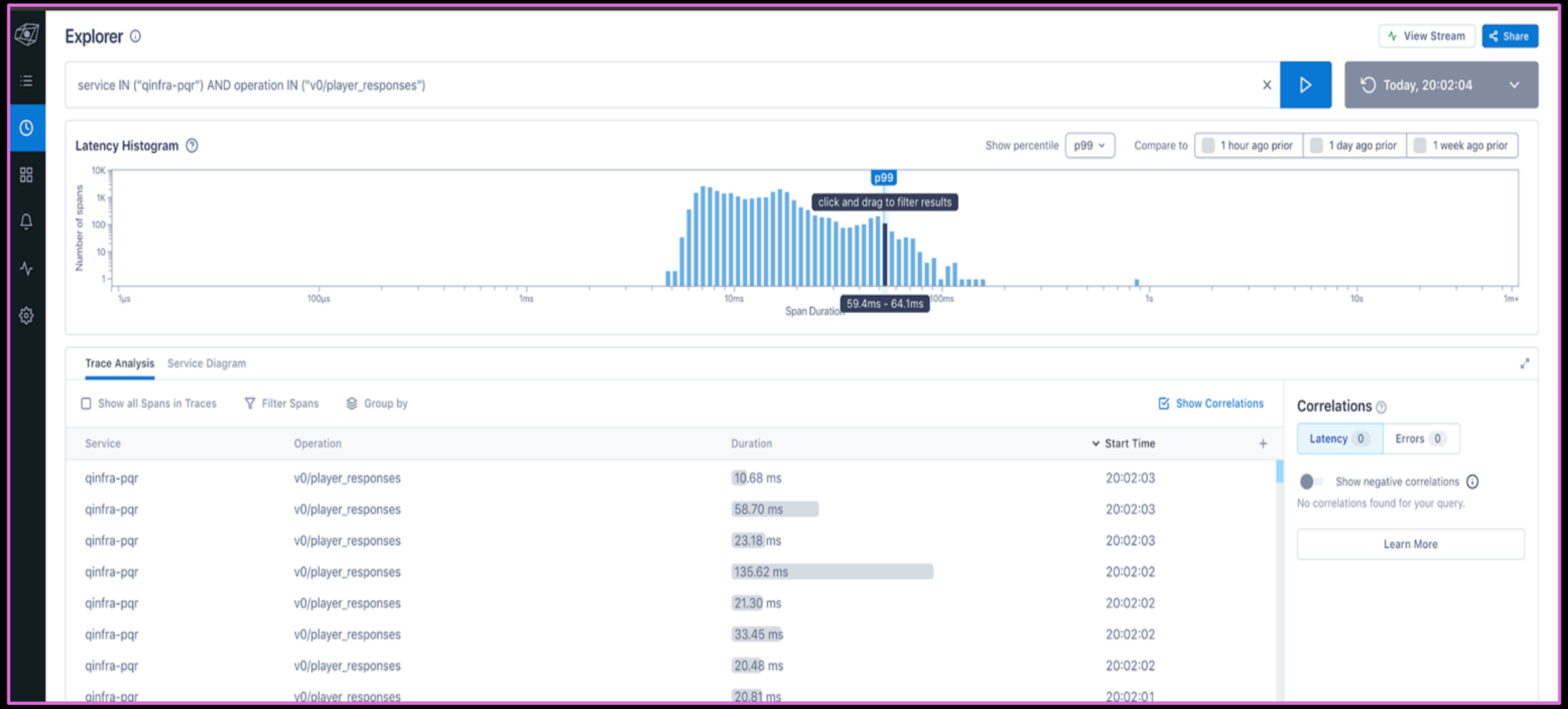
Experiment results

	Redis with Sorted Set (Prior solution)	Redis with Protocol Buffer	Aurora Serverless with JSONB	Tiered DynamoDB (Current solution)
Scaling			✓	✓
Performance	✓	✓		✓
Cost Effective ($)		✓	✓	✓
Ease of Recoverability			✓	✓
Set Append Semantics (idempotence)	✓			✓

Architecture



Latency


Explorer
View Stream
Share
service IN ("qinfra-pqr") AND operation IN ("v0/player_responses")
Today, 20:02:04
Latency Histogram
Show percentile
p99
Compare to
1 hour ago prior
1 day ago prior
1 week ago prior
Number of spans
10K
1K
100
10
1
p99
click and drag to filter results
1µs
100µs
1ms
10ms
100ms
1s
10s
1m+
59.4ms - 64.1ms
Span Duration
Trace Analysis
Service Diagram
Show all Spans in Traces
Filter Spans
Group by
Show Correlations
Service
Operation
Duration
Start Time
qinfra-pqr v0/player_responses 10.68 ms 20:02:03
qinfra-pqr v0/player_responses 58.70 ms 20:02:03
qinfra-pqr v0/player_responses 23.18 ms 20:02:03
qinfra-pqr v0/player_responses 135.62 ms 20:02:02
qinfra-pqr v0/player_responses 21.30 ms 20:02:02
qinfra-pqr v0/player_responses 33.45 ms 20:02:02
qinfra-pqr v0/player_responses 20.48 ms 20:02:02
qinfra-pqr v0/player_responses 20.81 ms 20:02:01
Correlations
Latency 0
Errors 0
Show negative correlations
No correlations found for your query.
Learn More

Key takeaways

- Understand your access patterns – that will drive your data storage decisions
- Design services to allow underlying implementation to change without impacting calling apps
- Leverage AWS expertise to ensure you are optimized for cost and performance

For more information:
https://aws.amazon.com/retail



© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Thank you!



AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.



Please complete the session survey

AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.