AWS re:Invent

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

DAT307

Deep dive on Amazon MemoryDB for Redis

Jon Fritz
Head of Product Management,
In-Memory Database and Caching Services
AWS

Mahesh Lagishetty
Vice President, Data Engineering
TSYS

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

Trends that impact the way data is generated and consumed

Explosion of data



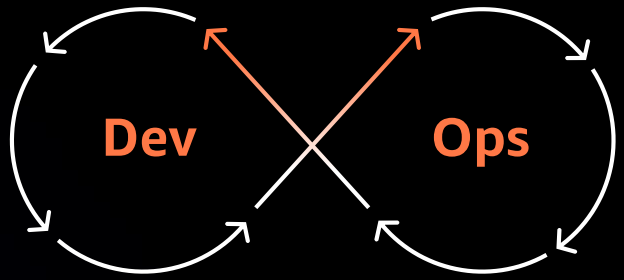
Data grows 10 times every 5 years, driven by network-connected smart devices

Microservices change data and analytics requirements



Microservices architecture decreases the need for one-size-fits-all databases and increases the need for real-time monitoring and analytics

Rapid rate of change driven by DevOps

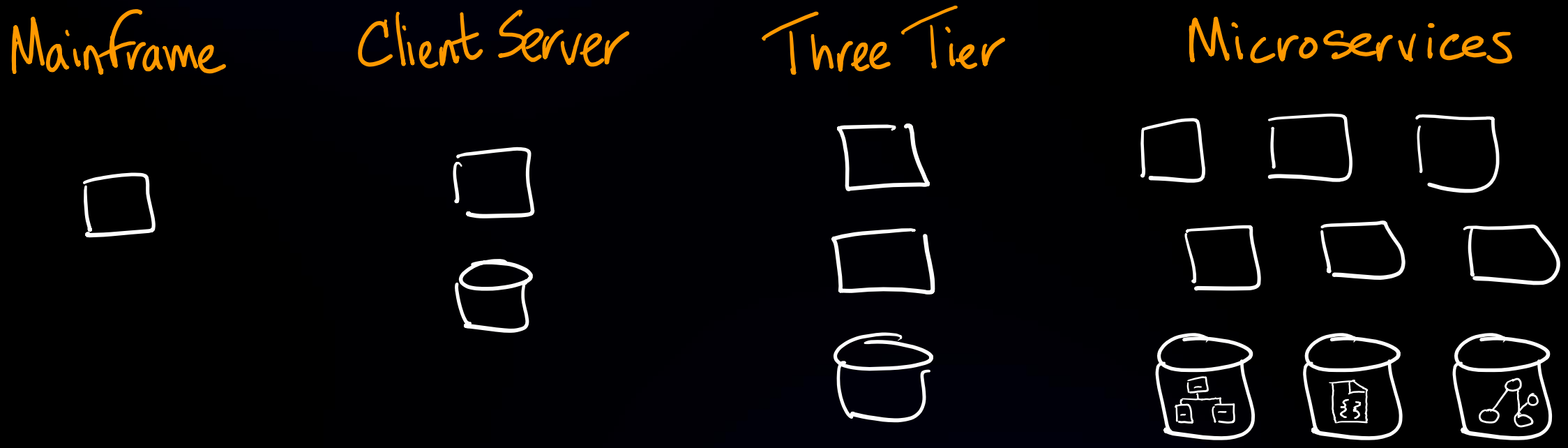


The transition from IT to DevOps increases the rate of change

aws

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

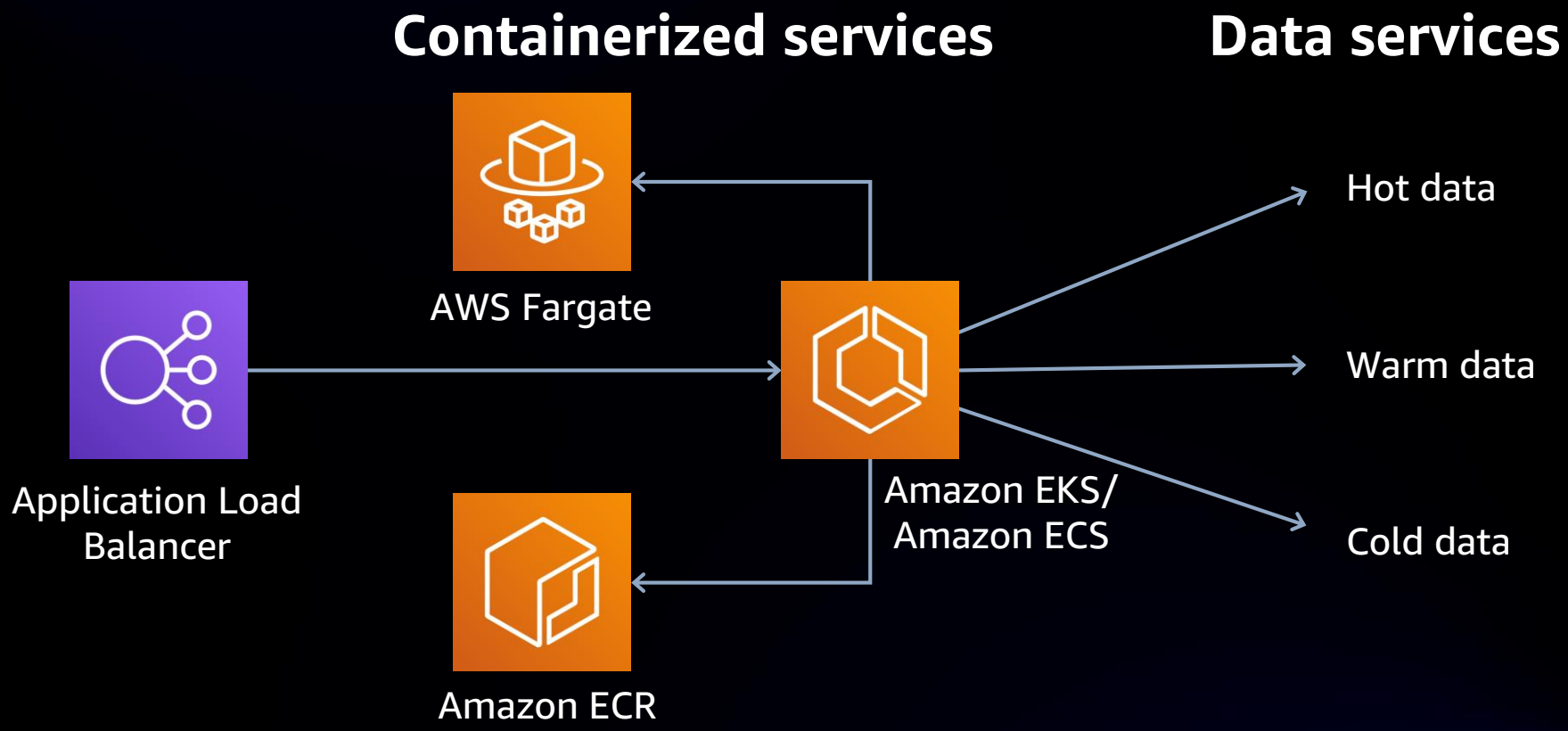
Application architecture and patterns have evolved



Microservices change how applications are built in the cloud

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

Microservices applications: More emphasis on hot data



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

Hot data requires a fast, flexible, and friendly database

1. Fast
   - Single user action can involve many microservices, each with a tiny latency budget
   - Millions of concurrent users can drive throughput of millions of requests per second
   - Individual services doing more computation and accessing more data
2. Flexible
   - High application velocity and development requires database agility
   - Application evolution to support unforeseen business requirements
   - Minimize impedance mismatch between application and database models
3. Friendly
   - Minimize tedious database admin tasks
   - Simple to do simple things – also enable complex things

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

Challenges with in-memory data stores



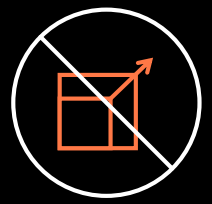
Difficult to manage

Manage server provisioning, software patching, setup, configuration, and backups



Hard to make robust

Many in-memory databases do not have data durability, which can lead to data loss during outages or failover; also, need to implement fast error detection and remediation



Difficult to scale

Online scaling can be error-prone, replication performance needs to be monitored during scaling operations, and data loss can occur if issues arise



Expensive

Invest in people, processes, hardware, and software

aws

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



Introducing Amazon MemoryDB for Redis

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

Purpose-built databases



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

Amazon MemoryDB for Redis

REDIS-COMPATIBLE, IN-MEMORY DATABASE WITH MULTI-AZ DURABILITY

Ultra-fast performance

Microsecond read and single-digit millisecond write latencies with millions of TPS

Redis compatibility

Flexible and friendly Redis APIs and data structures

Durability and high availability

Multi-AZ transactional log for durability and replicas for high availability

Fully managed

AWS managed hardware and software setup, configuration, monitoring, and snapshots

High scalability

128 TB of storage per cluster (with 1 replica per shard)

Security

Amazon VPC, encryption at-rest and in-transit, access control list (ACL)

aws

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

Industry use cases

In-Memory

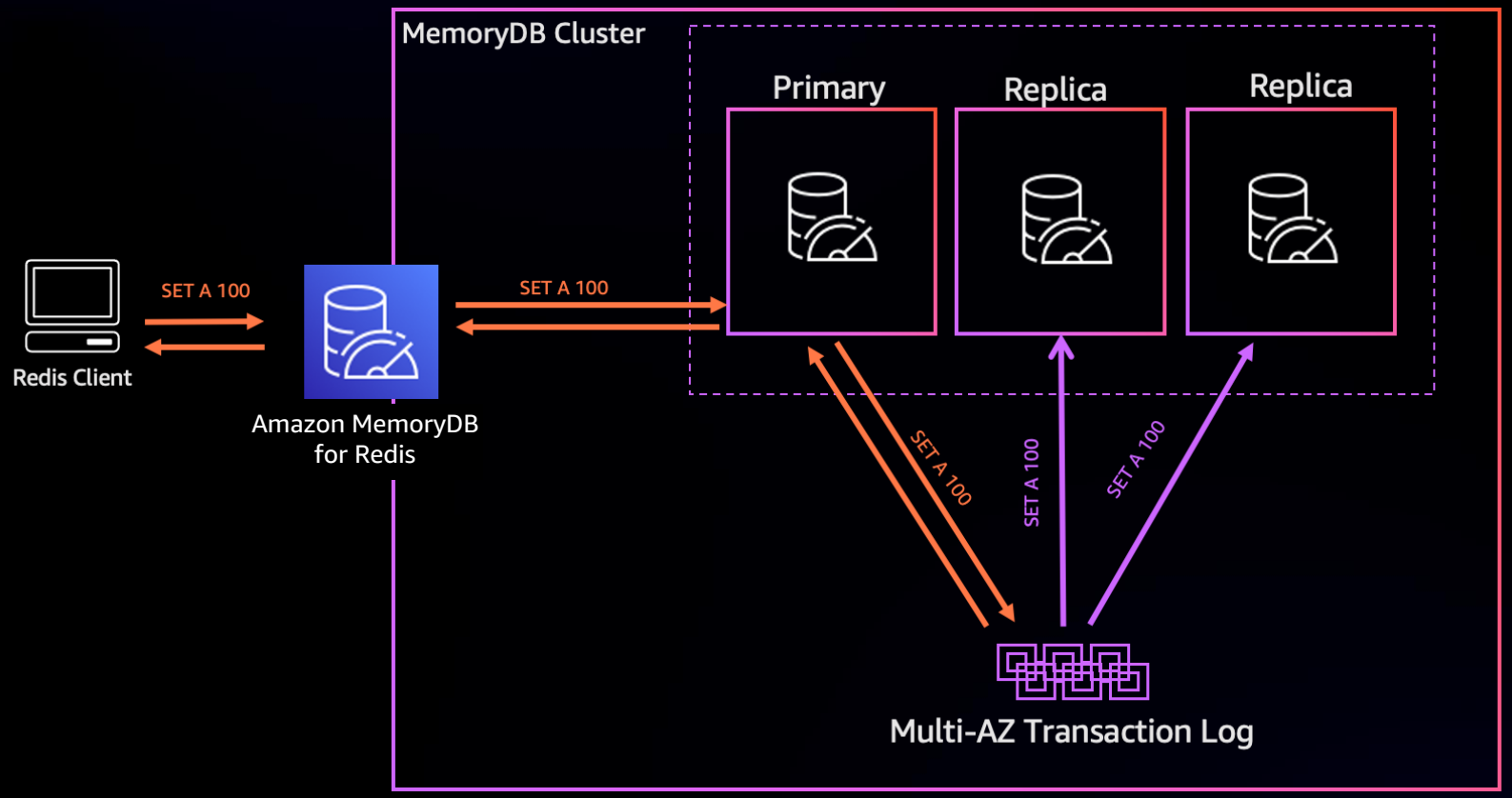


Amazon MemoryDB

- Retail: customer profiles and accounts, inventory tracking and fulfillment
- Gaming: leaderboards, player data stores, session history
- Banking and finance: user transactions, fraud detection
- Media and entertainment: user data stores, real-time streaming
- Internet of Things (IoT): real-time analytics, spatial lookups

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

Fast and highly scalable

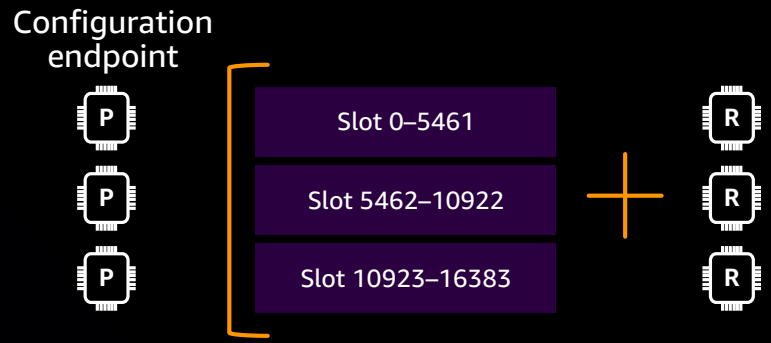


- Database design
  - All data in memory
- Low latency
  - Microsecond reads
  - Single-digit millisecond writes
- High throughput
  - Sharded parallelism and pipelined writes

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

Clustered architecture

Feature	Redis cluster
Recovery time	Typically <200 ms planned failover, <10 second unplanned failover
Failover impact	Writes affected on failed shard Reads available
Node scale	Up to 500 nodes (from 83 to 500 shards) 1 primary and 0–5 replicas per shard
Storage	More than 100 TB with 1 replica per shard
Online scaling	Shards, primary nodes, and read replicas
Migration path	Backup/restore snapshot, third-party tooling
Scalability and performance	• Achieve greater throughput through horizontal scaling • Horizontal/vertical scaling supported
Scaling operation	Cluster resizing • Horizontal scaling to add/remove shards • Read scalability to add/remove replicas



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

Performance: R6g.16xlarge node performance benchmark

390K reads per second: Scale further with replicas

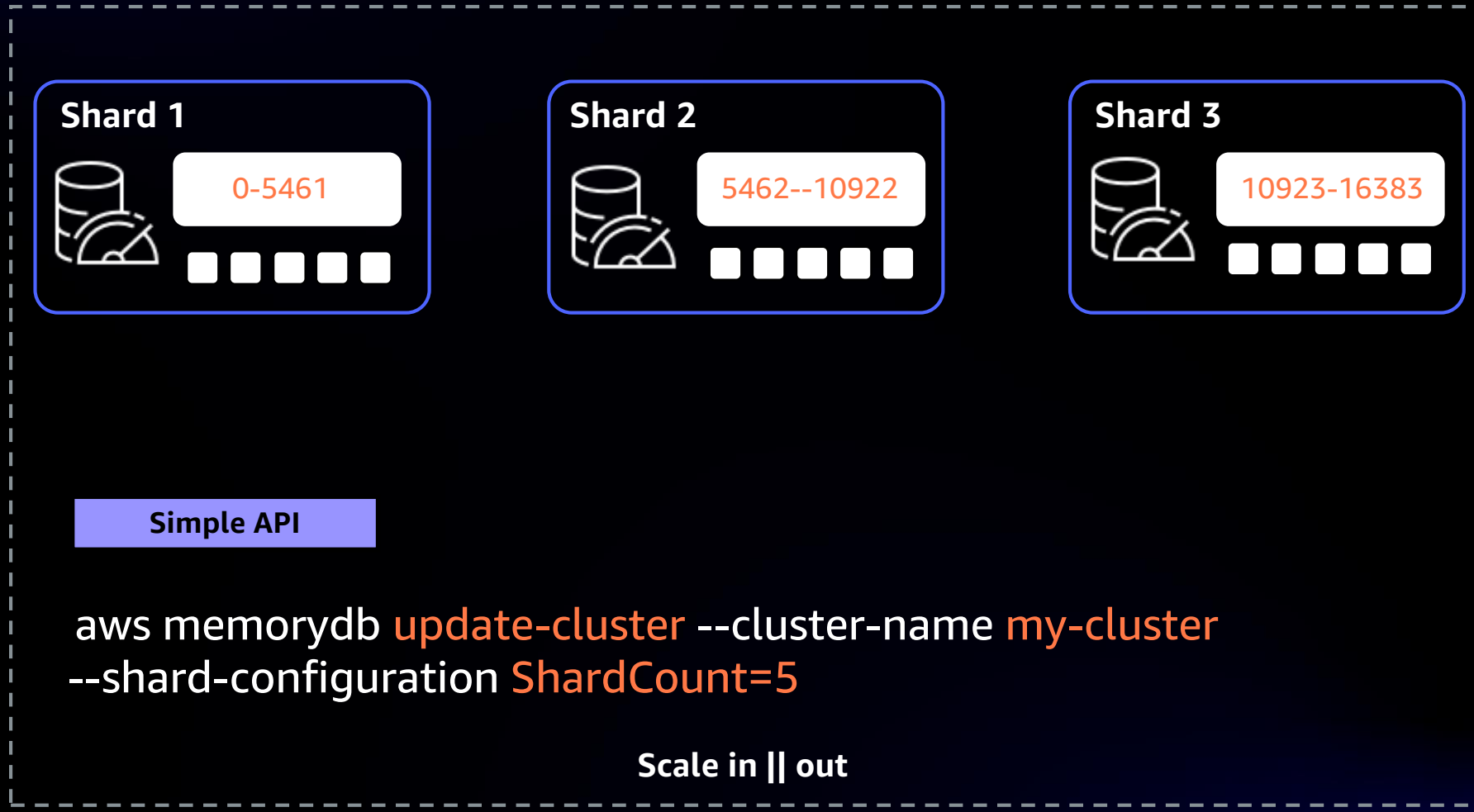
100K writes per second: Scale further with sharding

1.3 GB/second reads, 100 MB/second writes

Based on internal read workload, and write workload tests of r6g.16xlarge node

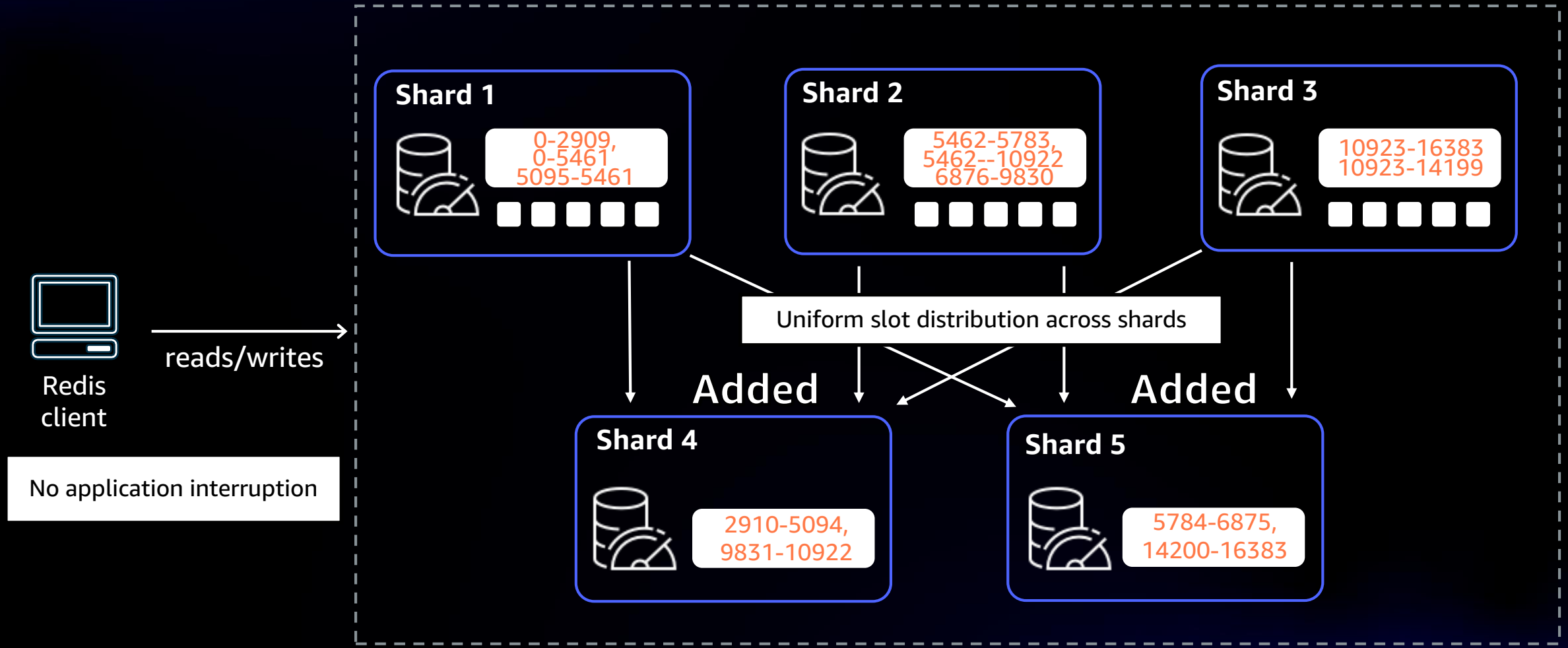
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

HA and scalability: Online re-sharding



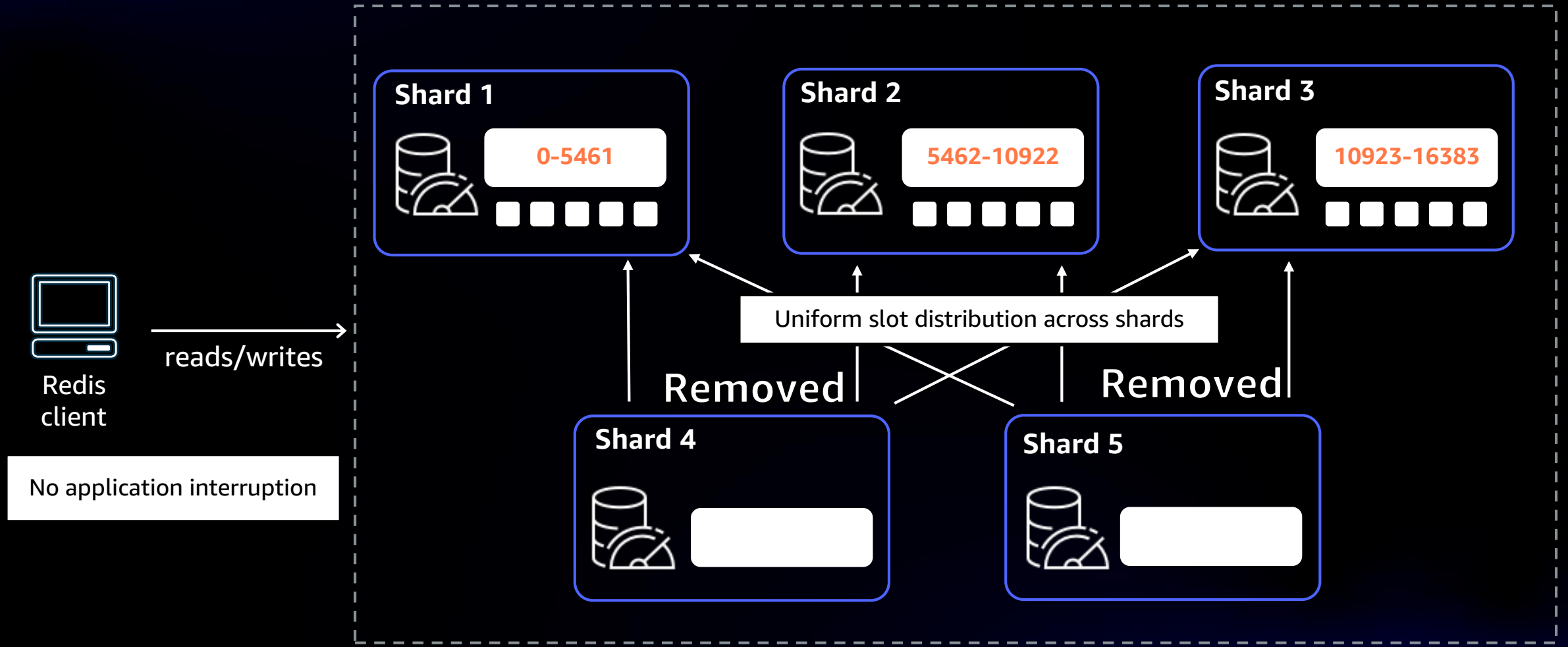
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Scalability: Online scale-out



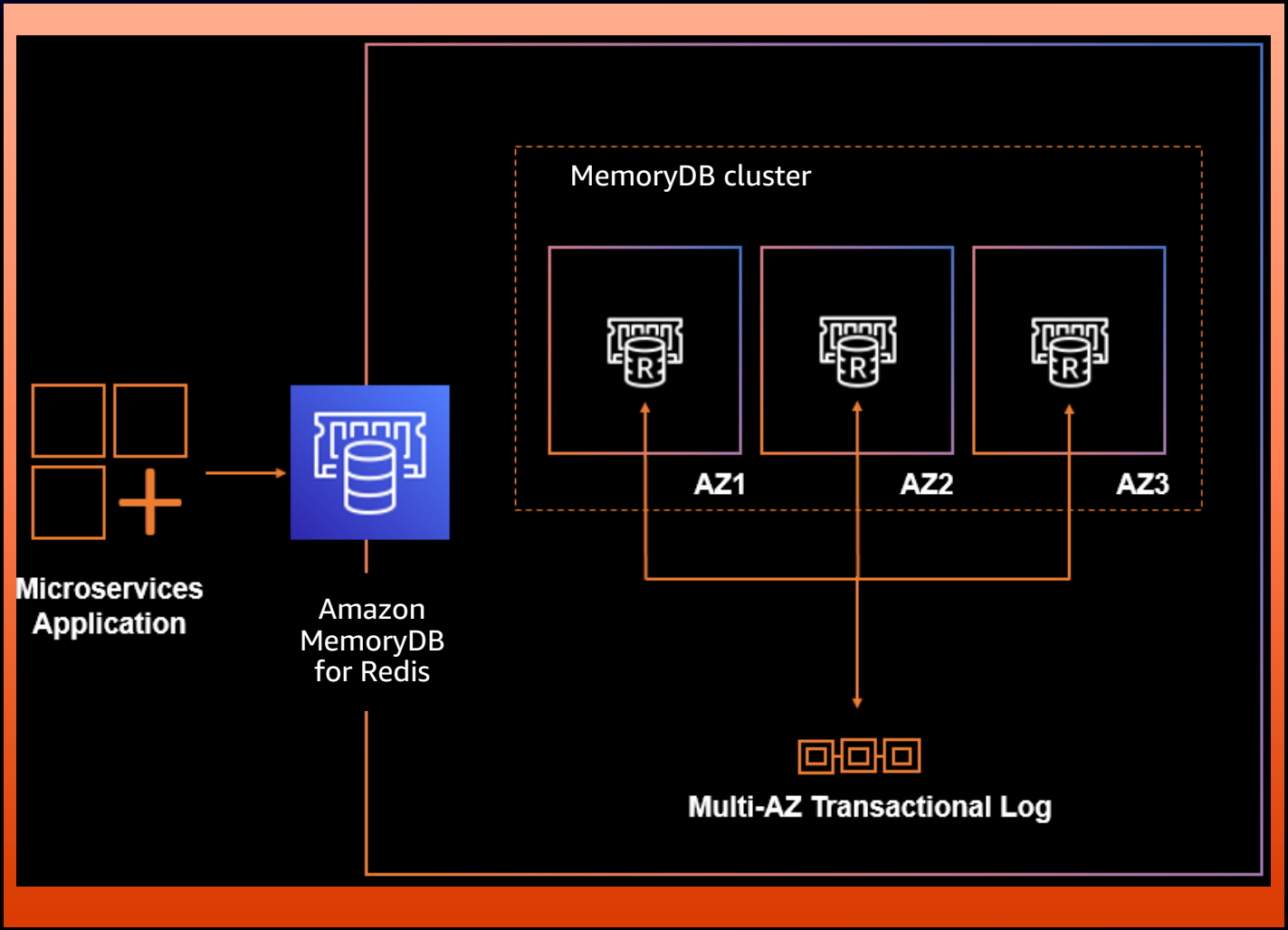
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Scalability: Online scale-in



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

Durable and highly available



- All writes committed to Multi-AZ transactional log
- Guaranteed delivery of writes to replicas

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

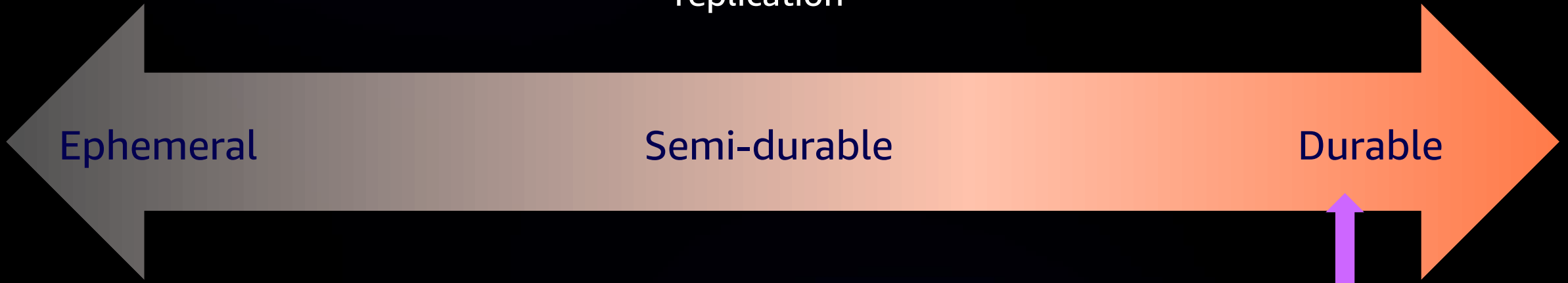
In-memory durability spectrum

No persistence

Snapshots

Asynchronous replication

Multi-AZ transaction log





ElastiCache for Memcached



ElastiCache for Redis

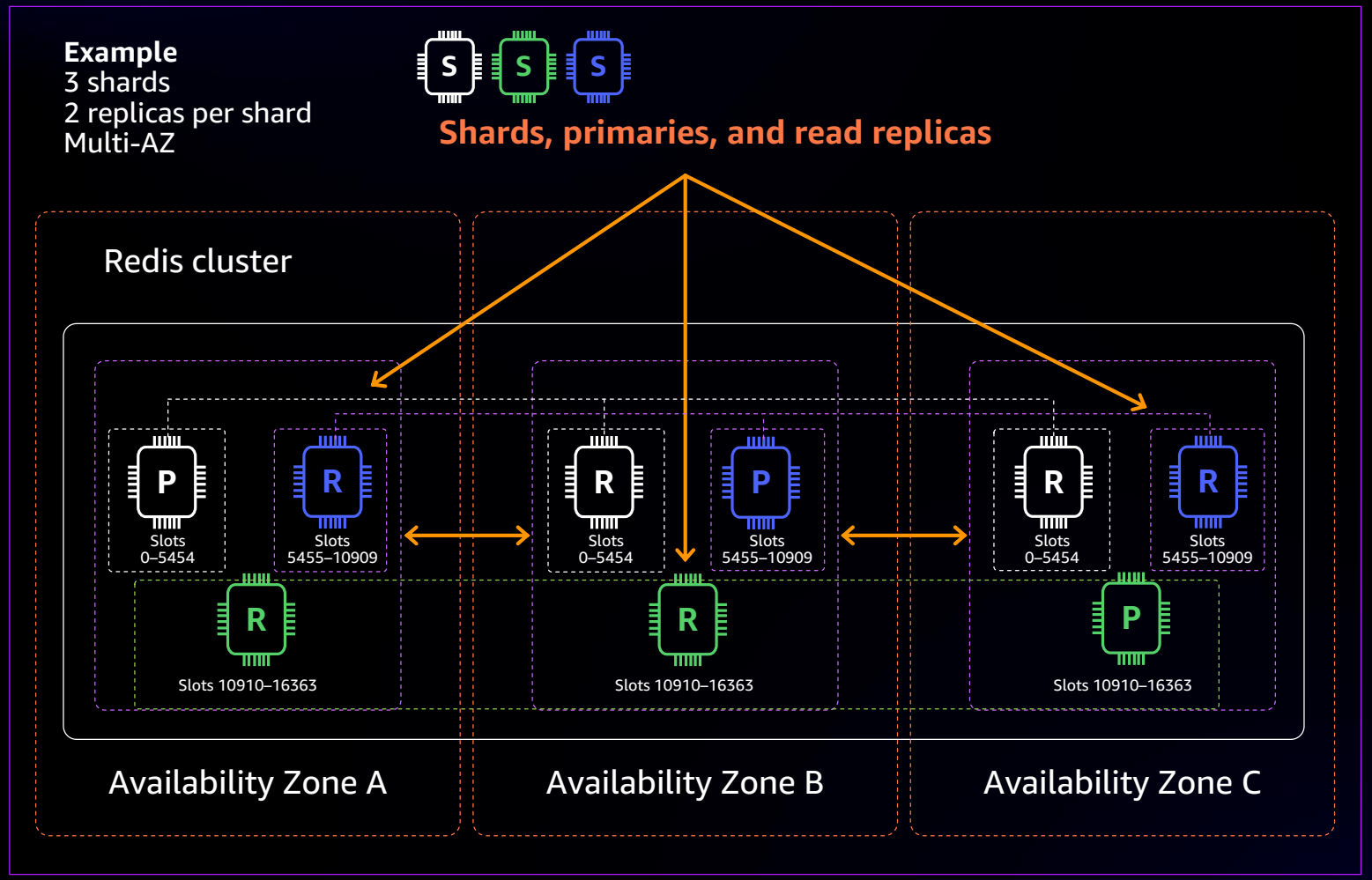


MemoryDB for Redis

aws

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

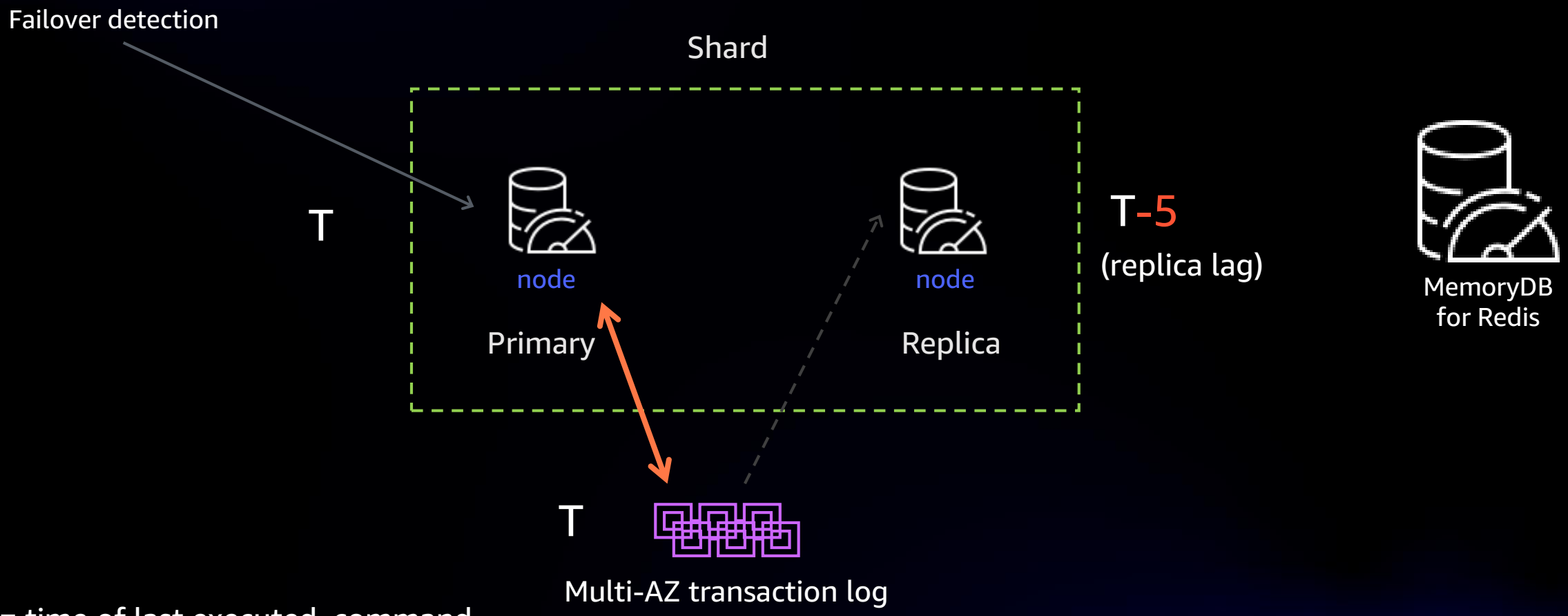
High availability: Multi-AZ and replication



- Add shards to scale reads/writes, increase in-memory capacity
- Add replicas to scale reads, increase availability

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

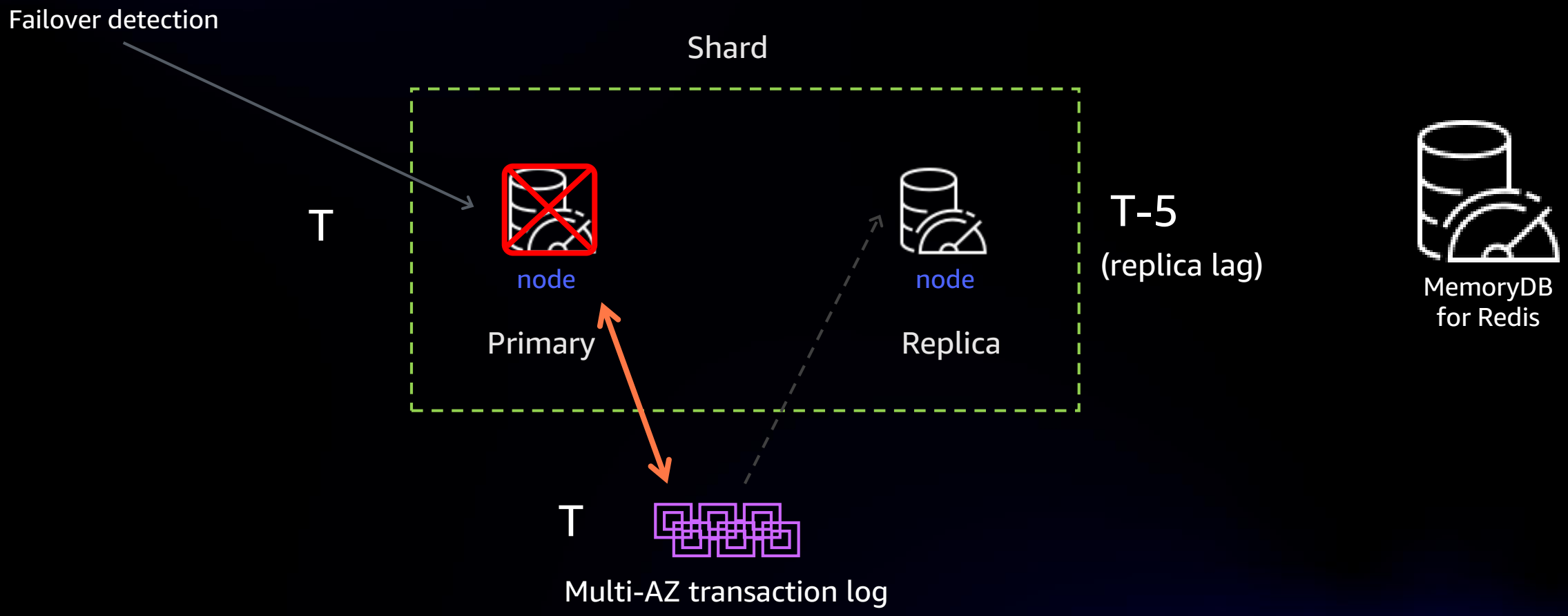
High availability architecture



T = time of last executed command

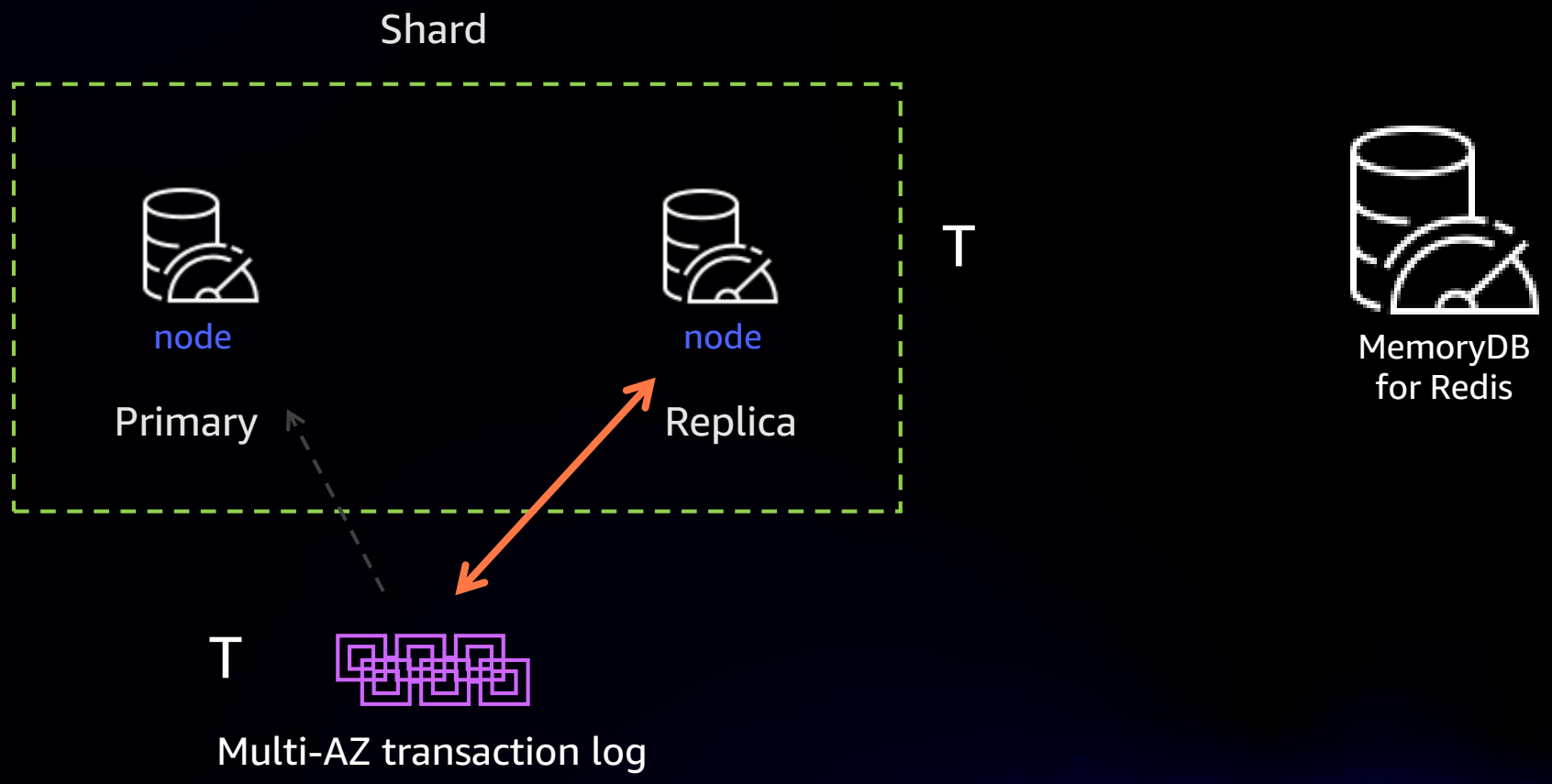
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

High availability architecture



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

High availability architecture



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

Flexible: Redis data structures and API

Data type	Description	Example use cases	Associated commands
String	Sequence of bytes	Fast data access	APPEND,GET,SET,INCR,DECR,GETSET…
List*	A list of strings	Many	LSET,LLEN,LPUSH,LPOP,LTRIM,RPOP…
Set*	Nonrepeating, unordered collection of strings	Cardinality	SADD,SCARD,SDIFF,SUNION,SINTER,SMEMBERS…
Sorted set*	Nonrepeating, ordered collection of strings	Leaderboards	ZADD,ZCARD,ZCOUNT,ZRANK,ZSCORE…
Hash*	Map of key/value pairs		HGET,HGETALL,HKEYS,HVALS,HMSET,HMGET…
Streams	Log data structure	Message queue	XADD,XRANGE,XREAD,XACK,XCLAIM,XLEN…
Geospatial	Longitude-/latitude-based entries	Maps, "nearby"	GEOADD,GEODIST,GEOPOS,GEORADIUS…
Bitmaps	Special usage of string type		GETBIT,BITCOUNT,SETBIT,SETRANGE,GETRANGE…
HyperLogLogs	Special usage of string type		PFADD,PFCOUNT,PFMERGE

* These data types are capable of storing over 4 billion sub-elements

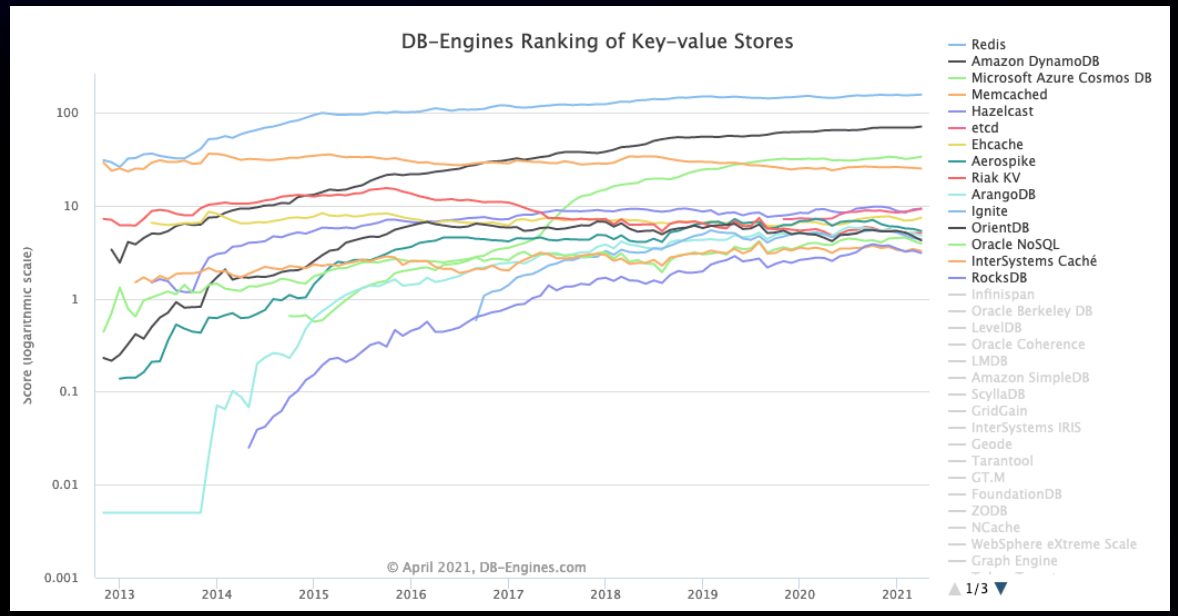
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Friendly: Ease of use

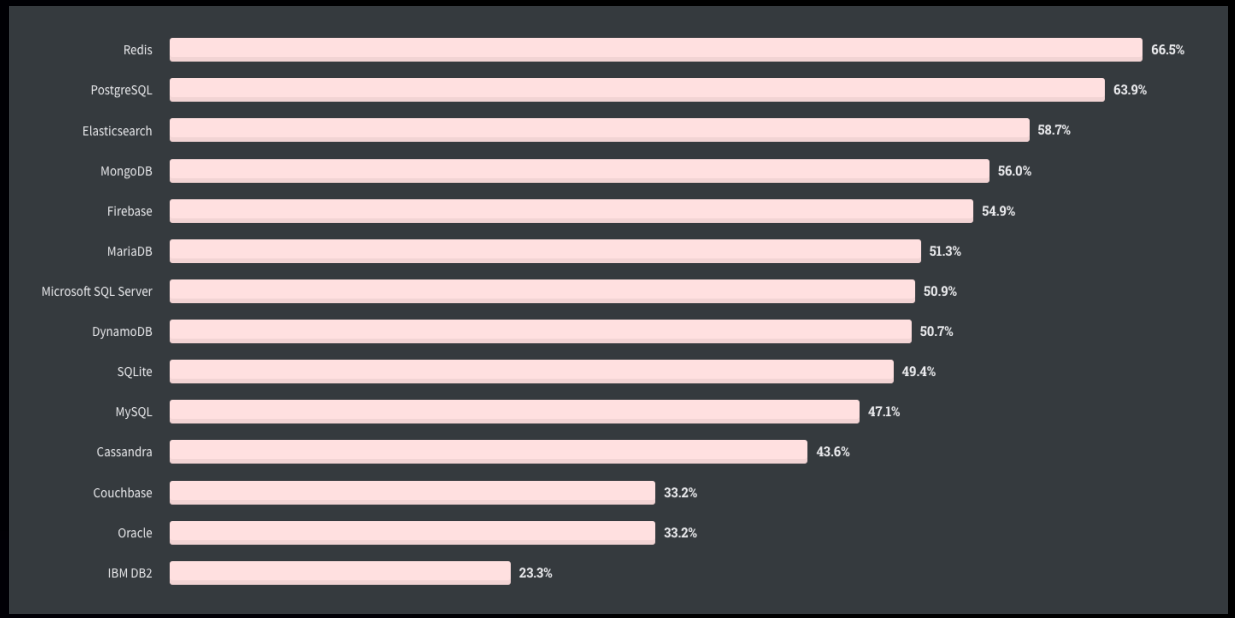
- Fully managed – leveraging our proven in-memory infrastructure
- Easily migrate existing data in Redis and ElastiCache to MemoryDB
- Support for AWS Graviton2 instance types for great price/performance
- Support for security features like Amazon VPC, encryption with AWS KMS, and authentication and authorization with Redis ACLs
- Improvements to open-source Redis clients for ease of use and getting started

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

Redis is the developer-friendly database



https://db-engines.com/en/ranking_trend/key-value+store



https://insights.stackoverflow.com/survey/2020#most-loved-dreaded-and-wanted

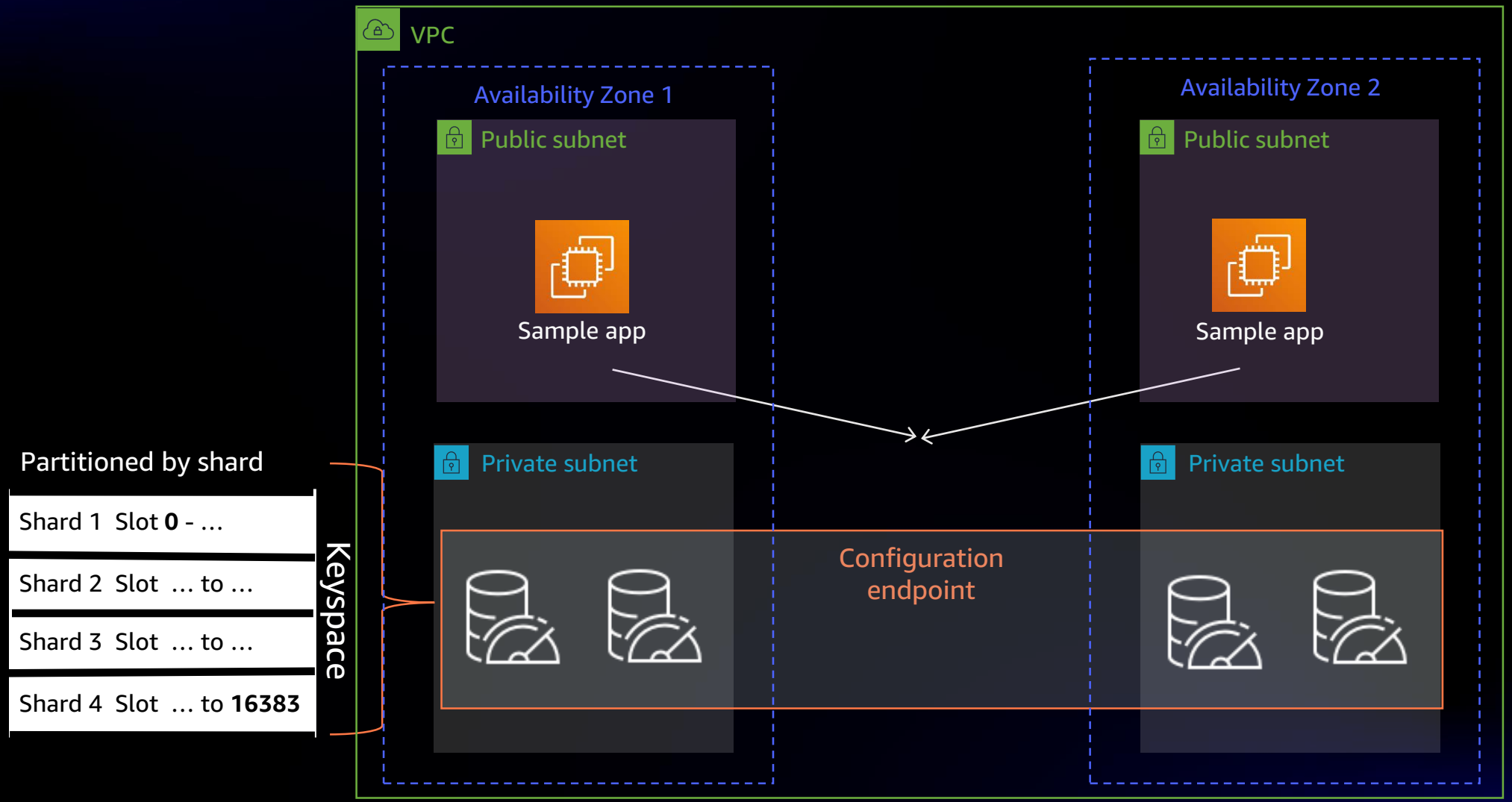
Redis has topped the DB-ENGINES rankings for KV-stores since 2013 (note log scale on Y-axis)

Redis has won the Stack Overflow “Most Loved” database award for the last 5 years in a row

aws

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

Getting connected to MemoryDB



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

Getting connected to MemoryDB

USE EXISTING OPEN-SOURCE REDIS CLUSTER-AWARE CLIENTS – PYTHON EXAMPLE

```python
import rediscluster
import certifi

MemoryDBEndpoint = "MEMORYDB ENDPOINT"
startup_nodes = [{"host": MemoryDBEndpoint, "port": "6379"}]

cluster = rediscluster.RedisCluster(startup_nodes=startup_nodes, username='zach',
          password='REDISAUTH12345678910', skip_full_coverage_check=True, decode_responses=True,
          ssl=True, ssl_ca_certs=certifi.where())

cluster.set("hello", "memorydb")

print(cluster.get("hello"))
```

Use open-source cluster-aware Redis client

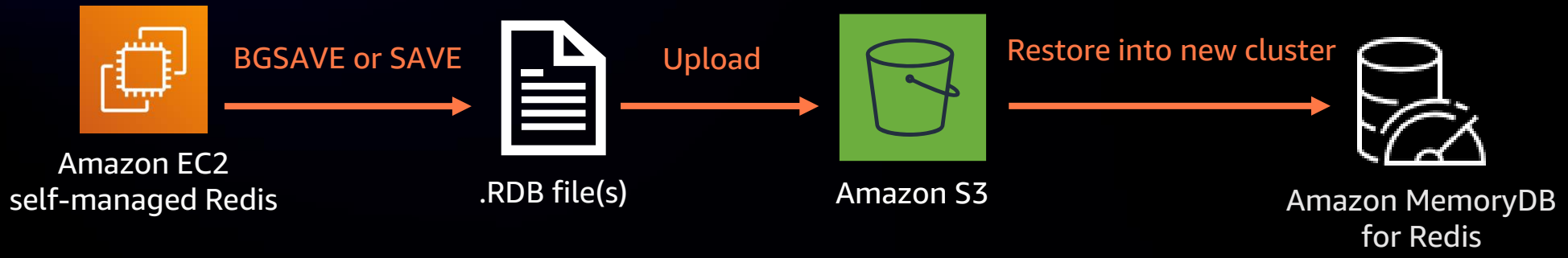
Authenticate with ACLs

Use open-source Redis 6.2 API commands

aws

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

Migrate from Redis: Backup/restore



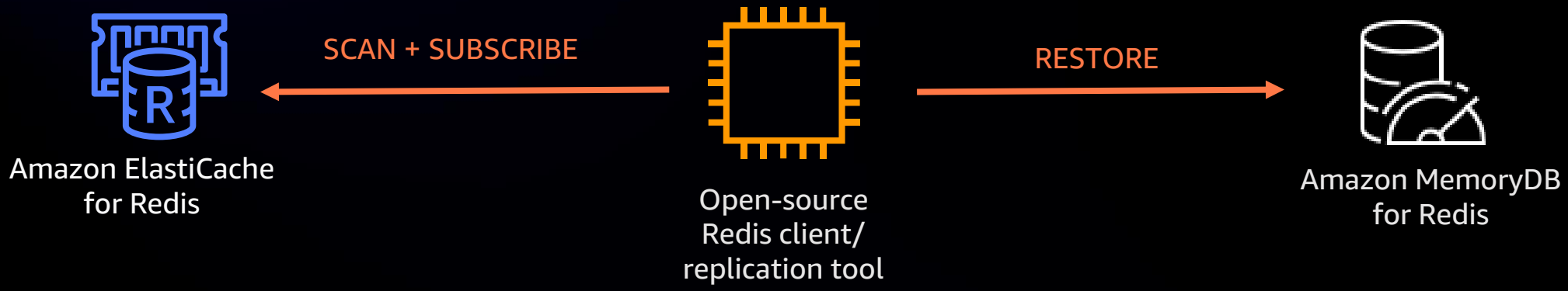
Recommendation: leverage planned maintenance window

1. Create a Redis backup using BGSAVE OR SAVE operations
2. Create an Amazon S3 bucket and folder
3. Upload your backup to Amazon S3
4. Grant MemoryDB read access to the .RDB file
5. Seed the MemoryDB cluster with the .RDB file data

aws

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

Logical replication: Open-source clients



Recommendation: minimal downtime scenarios

1. Create your MemoryDB cluster
2. Ensure client connectivity between your source Redis (ElastiCache or open source) and MemoryDB for Redis
3. ETL data using SCAN & SUBSCRIBE and restore keys into MemoryDB for Redis

aws

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

Security features



Encryption

- Encryption-in-transit
- Encryption for all data at rest

Authentication

- Create users with passwords, and access strings using Redis ACLs
- Only available with encryption-in-transit enabled
- Token and password rotation API

Authorization

- Ability to limit commands and keys using ACLs

Amazon VPC integration

- VPC security groups for authorized IPs, AWS resources
- VPC subnet groups for network isolation

aws

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

Security features: Amazon CloudWatch monitoring

IMPORTANT CLUSTER SECURITY METRICS IN CLOUDWATCH

AuthenticationFailures = Failed attempts to authenticate to Redis

KeyAuthorizationFailures = Failed attempts by users to access keys without permission

CommandAuthorizationFailures = Failed attempts by users to run commands without permission

aws

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

Security features: Authentication



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

Security features: Authorization

User	Authorized	Keyspace	Access string
 readapp	**GET** command	App keys	`on ~app:* -@all +get`
 marketing	All commands	Marketing keys	`on +@all ~mktg:* ~marketing:*`
 analytics	All read-only commands	All keys	`on ~* +@read`

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

Open-source Redis security commands

Run the ACL LOG command to learn more

```
> acl log
1)  1) "count"
    2) (integer) 1
    3) "reason"
    4) "command"
    5) "context"
    6) "toplevel"
    7) "object"
    8) "lrange"
    9) "username"
   10) ”dbuser"
   11) "age-seconds"
   12) "562.735999999999999"
   13) "client-info"
   14) "id=15 addr=172.31.43.79:60584 fd=11 name= age=25
       idle=0 flags=N db=0 sub=0 psub=0 multi=-1 qbuf=42
       qbuf-free=32726 obl=0 oll=0 omem=0 events=r
       cmd=lrange user=dbuser"
```

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

Choosing a fast NoSQL database service

MemoryDB

- Microsecond read and single-digit millisecond write latencies
- Redis API compatible

DynamoDB

- Key value store with single-digit millisecond latency at limitless scale
- Serverless and auto scaling

ElastiCache

- Redis and Memcached API compatible
- Microsecond read performance for transient data

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

Using MemoryDB for data tokenization as a service



Innovation that delivers.

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

What's in this session?

- Introduction to use case
- Why MemoryDB?
- HA implementation
- MemoryDB security
- Benchmarking tests
- What's next in our journey?

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

TSYS introduction

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

About TSYS, a Global Payments company

Market leader

#1 issuer in US, Canada, UK, Ireland, and China

#2 issuer in Western Europe

100+ countries

1,300+ financial institutions

Scale

750+ million accounts

50+ billion transactions/yearr

15,000+ TPS on real-time event processing on low-latency workloads

1M+ TPS throughput on high-volume batch workloads

Breadth

100+ vertical industries including banks, fintech, retail, and restaurants

140+ payment method integrations

Integrations with all major payment networks and channels across the world

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

Data tokenization

1. **Goal:** to reduce the footprint and transfer of sensitive data which comes under payment card industry requirements across different applications within TSYS by tokenizing the data elements and using the tokens for querying

2. **How:** replace account/personal identifiable data with a random number and store them in one central secured storage area
   - Eliminates need to handle PAN/PII at every distributed system
   - Makes PAN/PII security the data tokenizer problem

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

Data tokenization characteristics



Storage: 4+ billion tokens

Locality: Global

Performance: Ms latency

Request rate: 1,000s/sec

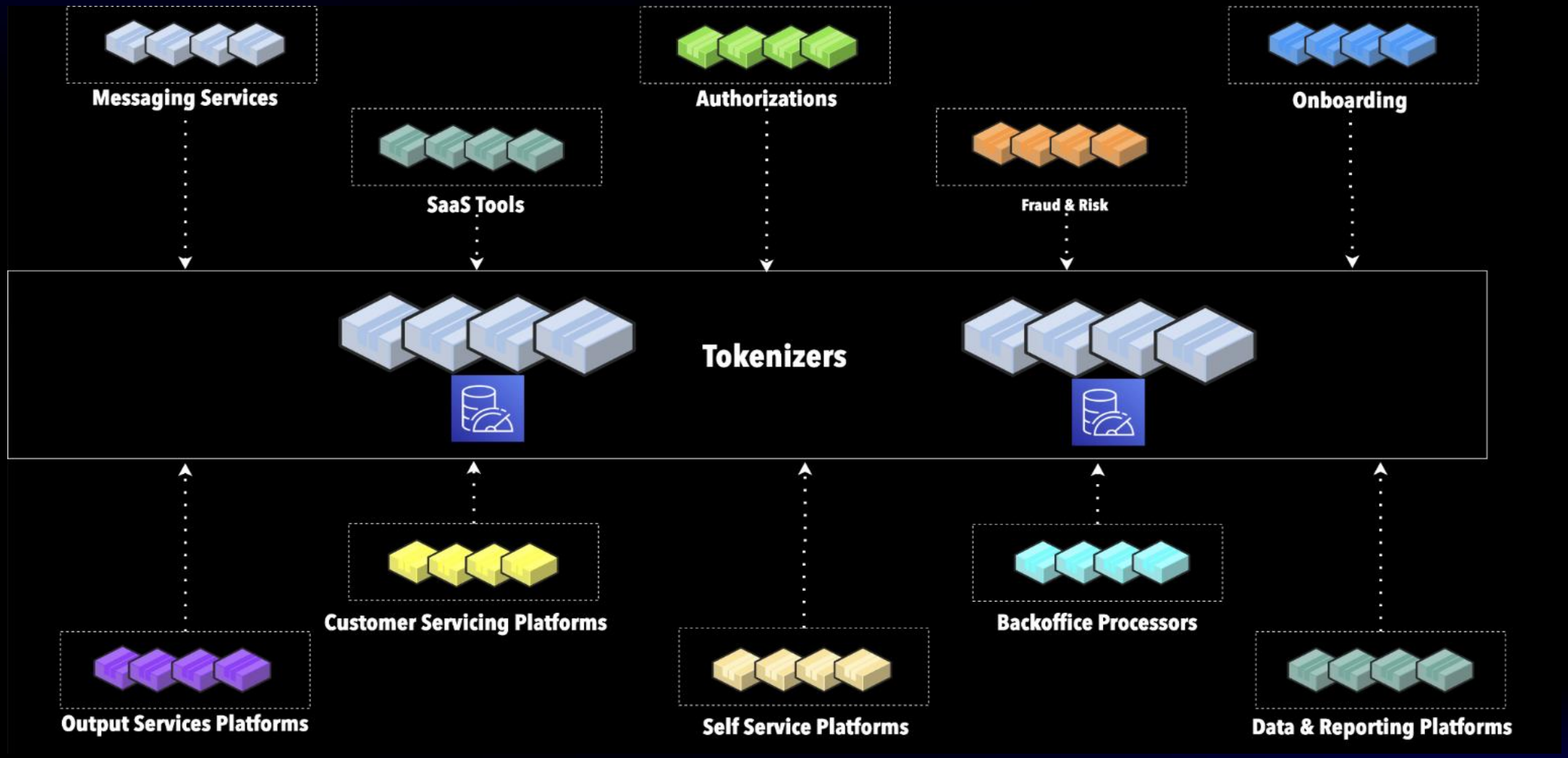
Access: GRPC, rest, stream

Availability: 24/7

Durability: Zero data loss

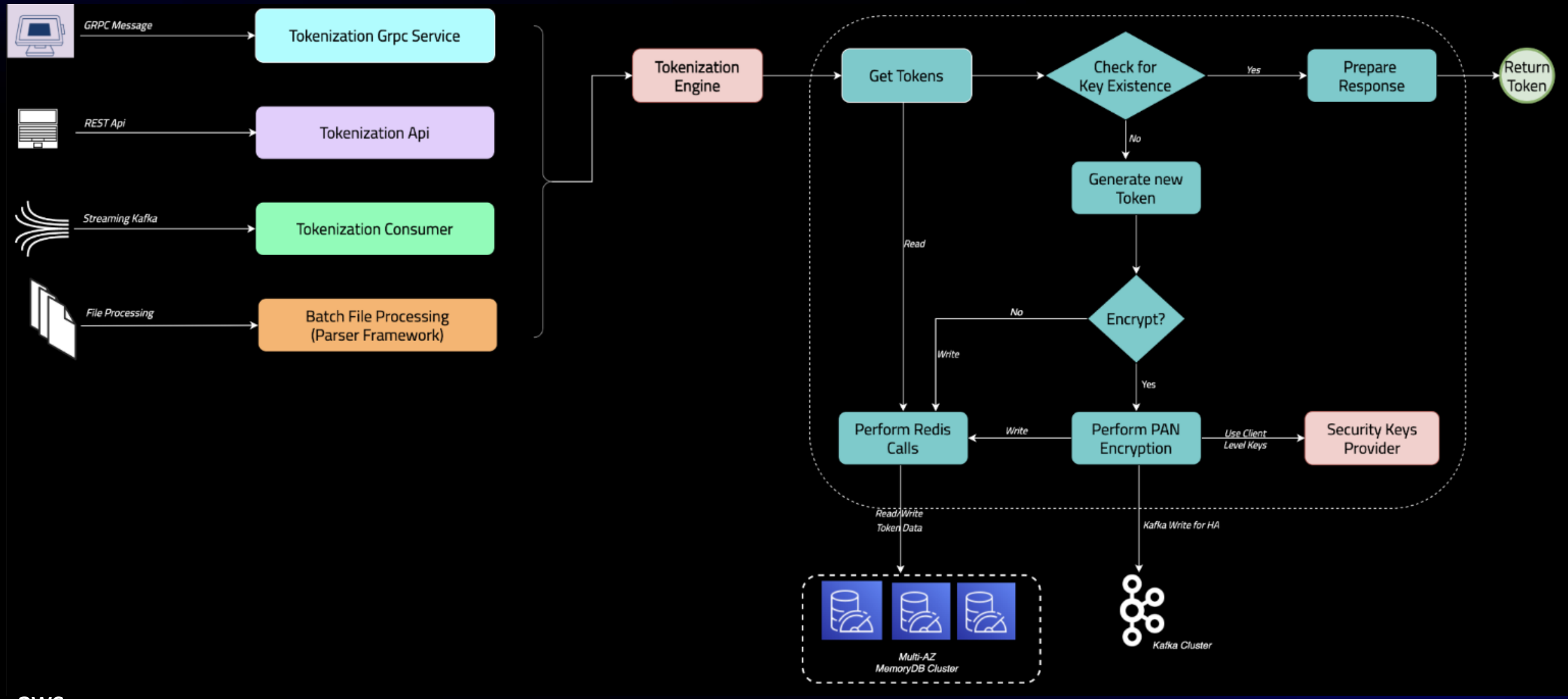
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Data tokenization touchpoints



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

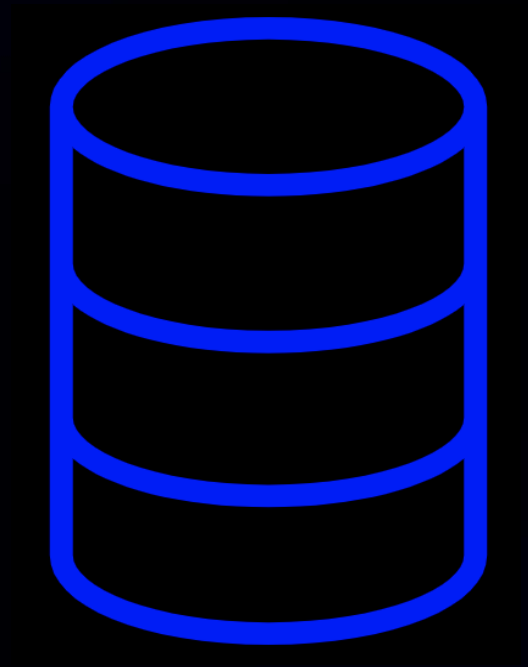
Data tokenization data flow



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

Token vault expectations

Storage
4+ billion tokens



Availability
24/7

Read IOPS
>=100,000

Write IOPS
>=20,000

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

Why MemoryDB?



Performance at scale

- Handles thousands of requests per second
- Provides single-digit millisecond write latency
- Provides zero data loss durability



No servers to manage

- Maintenance-free
- Managed scaling through API
- On-demand capacity

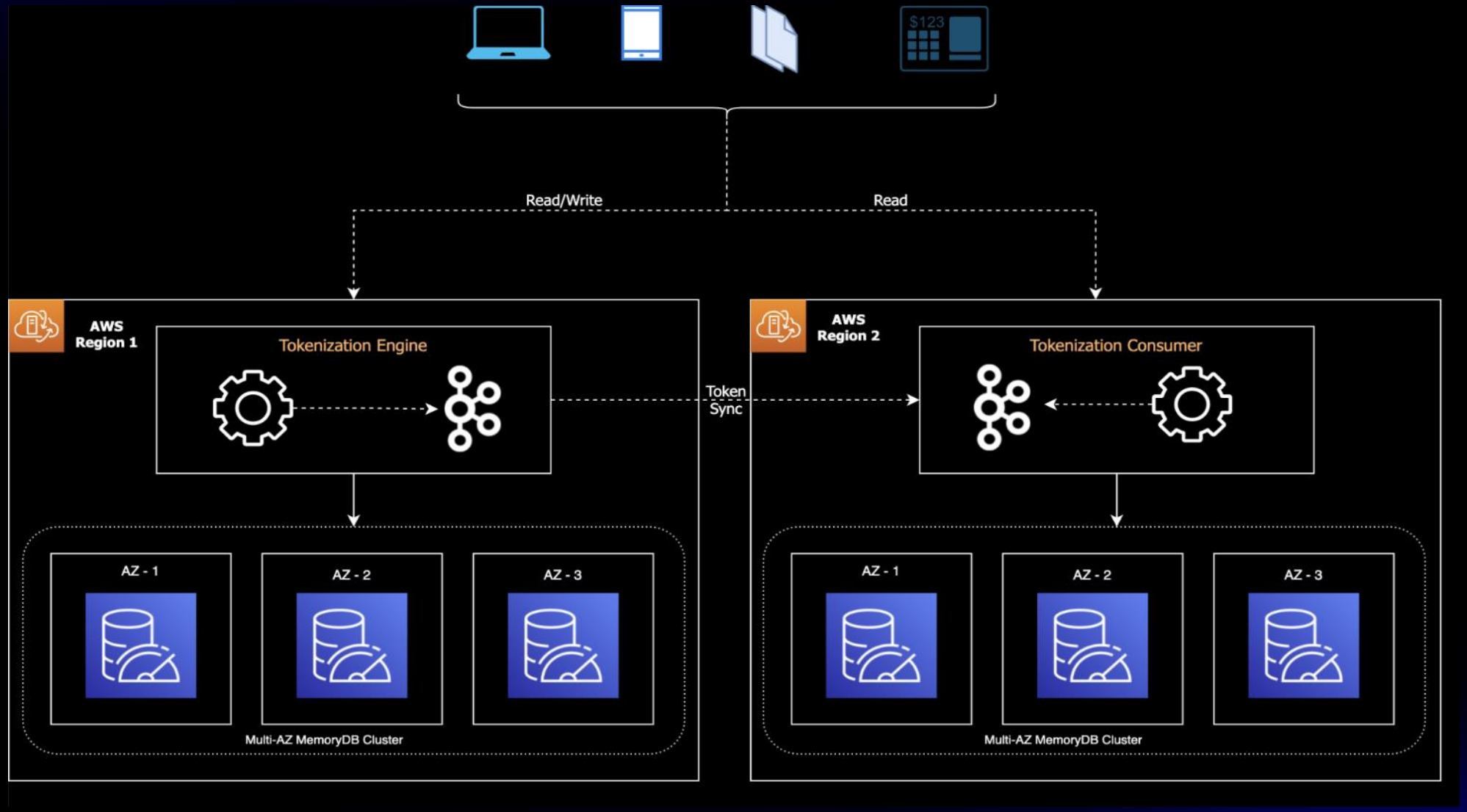


Enterprise application-ready

- Encryption at rest and transit
- On-demand backup/restore
- Access control using Redis ACLs

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

Data tokenization HA architecture



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

MemoryDB security

- Authentication/authorization
  - Redis ACL-based auth
    - Define users
    - Restrict commands users can execute
    - Restrict keys users can access
- Encryption
  - Data-at-rest encryption using CMK
  - Data-in-transit encryption using TLS

User name	Access string
memdbadmin	on ~* &* +@all
readonly	on ~* resetchannels -@all
appuser	on ~client:1234:* resetchannels -@all +@read +@write

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

MemoryDB scale testing

TSYS

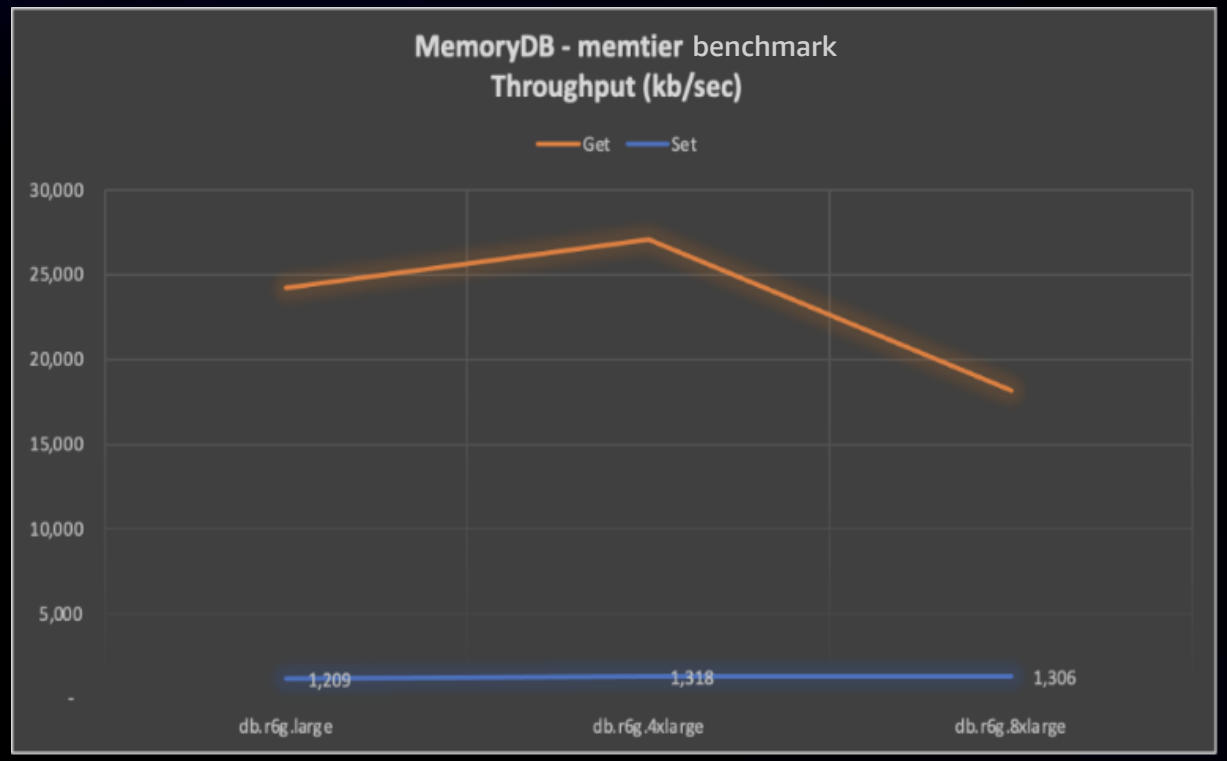
Test setup

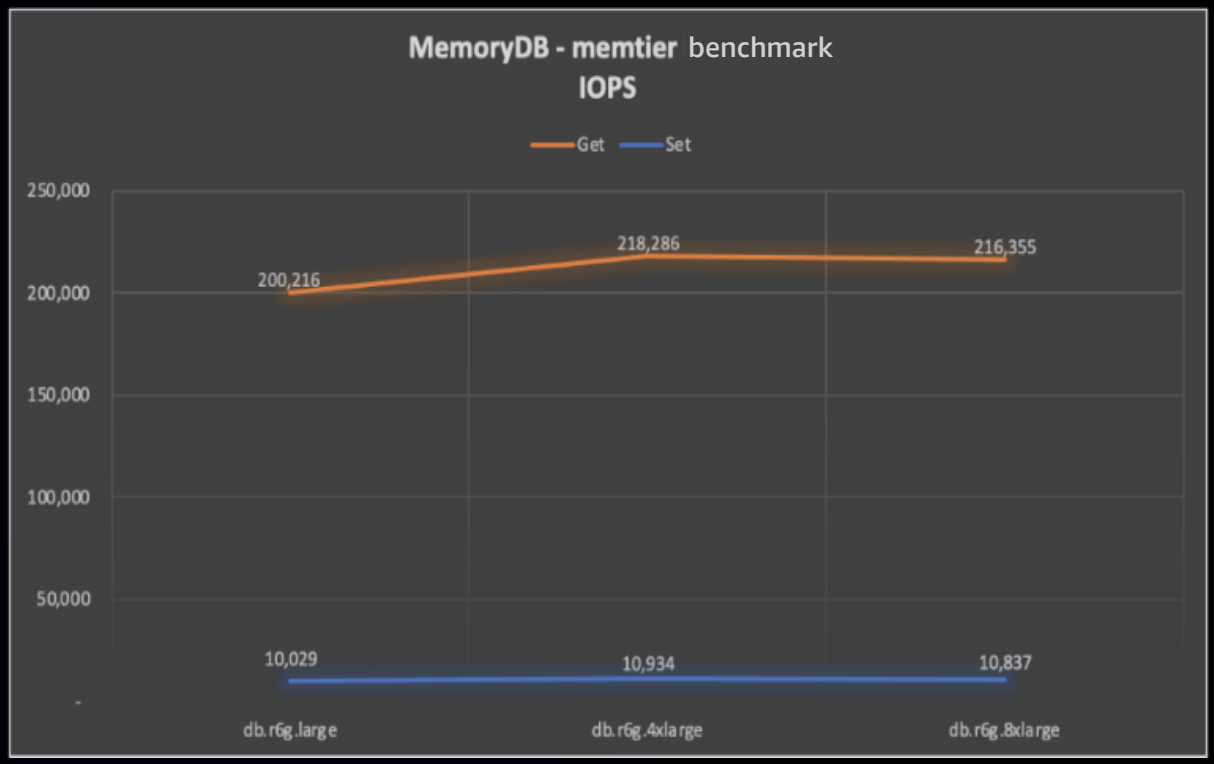
- Number of threads: 10
- Connections per thread: 20
- Requests per connection: 10,000
- Read-to-write ratio: 90:10

aws

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

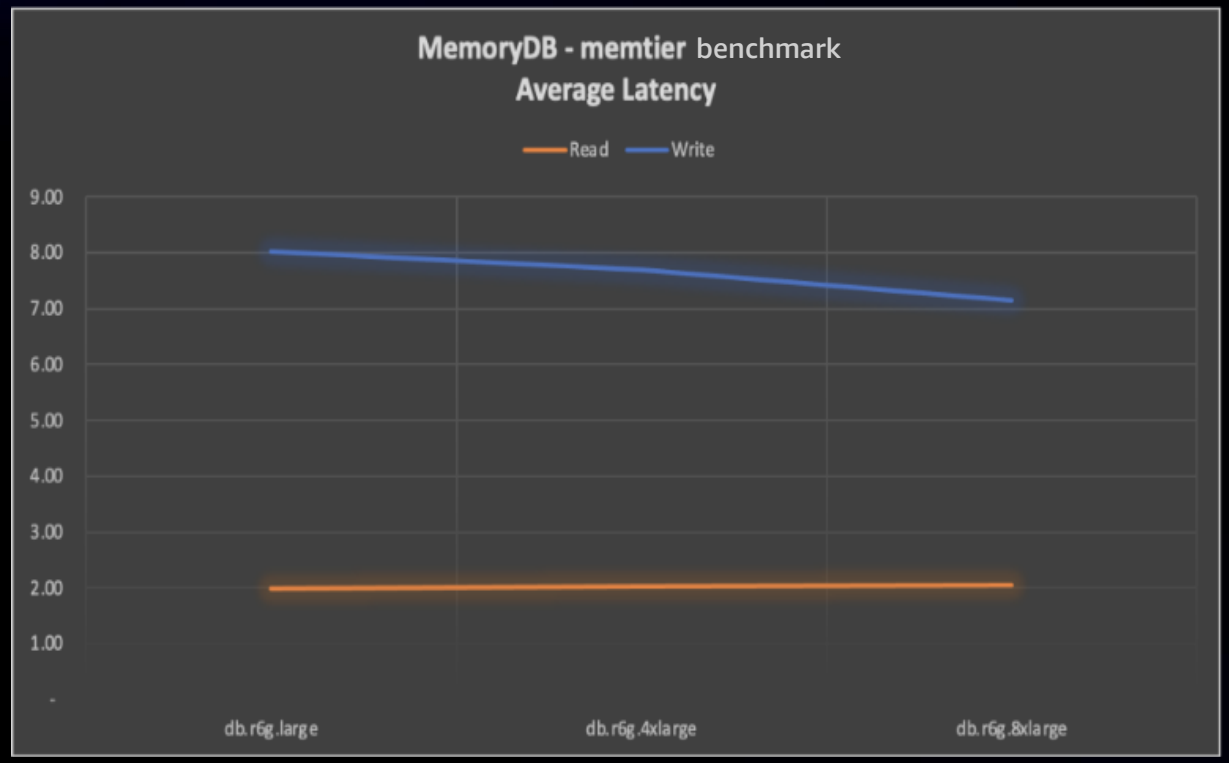
MemoryDB throughput results

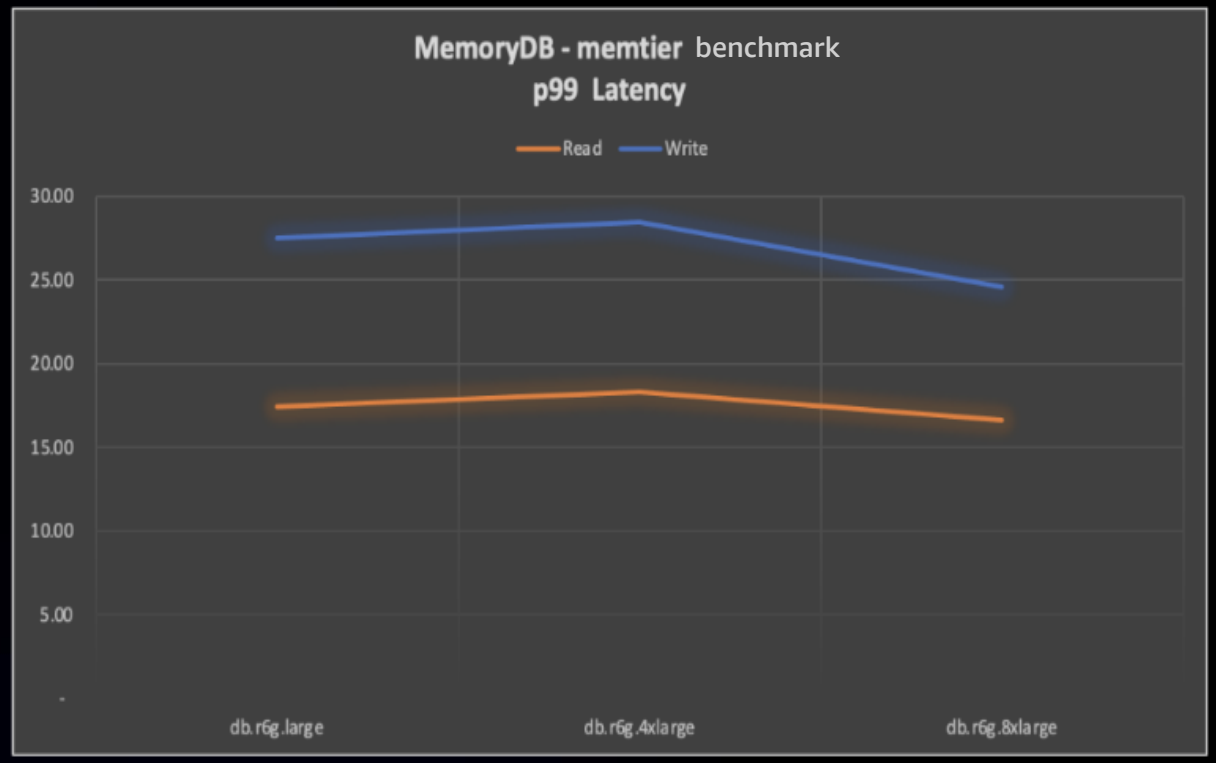




© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

MemoryDB latency results





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

What's next?

1. MemoryDB as data store for the program configurations
2. Keyspace notifications for notifying clients about the key changes
3. MemoryDB as cache for application state
4. MemoryDB as session state store for web and mobile applications

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

Thank you!

Jon Fritz

jonfritz@amazon.com

amazon.aws.com/memorydb

Mahesh Lagishetty

mlagishetty@tsys.com

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