Build data driven, high-performance, internet-scale applications

Julio Faerman
@faermanj
AWS Technical Evangelist
Characteristics of modern applications
Internet-scale and transactional

Users: 1M+
Data volume: TB–PB–EB
Locality: Global
Performance: Milliseconds–microseconds
Request Rate: Millions
Access: Mobile, IoT, devices
Scale: Up-out-in
Economics: Pay-as-you-go
Developer access: Instant API access

Ride hailing  Media streaming  Social media  Dating

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Modern applications require internet-scale performance

>20M requests per second  
>150M active users  
Data >1 PB
## Common data categories and use cases

<table>
<thead>
<tr>
<th>Relational</th>
<th>Key-value</th>
<th>Document</th>
<th>In-memory</th>
<th>Graph</th>
<th>Time-series</th>
<th>Ledger</th>
</tr>
</thead>
<tbody>
<tr>
<td>Referential integrity, ACID transactions, schema-on-write</td>
<td>High throughput, low-latency reads and writes, endless scale</td>
<td>Store documents and quickly access querying on any attribute</td>
<td>Query by key with microsecond latency</td>
<td>Quickly and easily create and navigate relationships between data</td>
<td>Collect, store, and process data sequenced by time</td>
<td>Complete, immutable, and verifiable history of all changes to application data</td>
</tr>
</tbody>
</table>

- **Relational**: Lift and shift, ERP, CRM, finance
- **Key-value**: Real-time bidding, shopping cart, social, product catalog, customer preferences
- **Document**: Content management, personalization, mobile
- **In-memory**: Leaderboards, real-time analytics, caching
- **Graph**: Fraud detection, social networking, recommendation engine
- **Time-series**: IoT applications, event tracking
- **Ledger**: Systems of record, supply chain, health care, registrations, financial
Purpose-built databases

- Relational: Amazon RDS, Aurora, MySQL, PostgreSQL
- Key-value: DynamoDB
- Document: DocumentDB
- In-memory: ElastiCache
- Graph: Neptune
- Time-series: Timestream
- Ledger: Quantum ElastiCache

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Internet-scale applications

with Amazon DynamoDB
Purpose-built databases for internet-scale apps

The world’s largest e-commerce business, Amazon.com, runs on nonrelational cloud databases because of their scale, performance, and maintenance benefits.

“A deep dive on how we were using our existing databases revealed that they were frequently not used for their relational capabilities.”

— Werner Vogels
CTO, Amazon
Needed a solution that scales and manage up to 8x more riders during peak times.

SOLUTION
DynamoDB stores GPS coordinates of all rides.
With AWS, Lyft saves on infrastructure costs and enables massive growth of ridesharing platform. There are now 23M people who use Lyft worldwide.
Amazon DynamoDB
Fast and flexible key value database service for any scale

**Performance at scale**
Consistent, single-digit millisecond response times at any scale; build applications with virtually unlimited throughput

**Serverless**
No hardware provisioning, software patching, or upgrades; scales up or down automatically; continuously backs up your data

**Comprehensive security**
Encrypts all data by default and fully integrates with AWS Identity and Access Management for robust security

**Global database for global users and apps**
Build global applications with fast access to local data by easily replicating tables across multiple AWS Regions

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.
DynamoDB transactions

Simplify your code by executing multiple, all-or-nothing actions within and across tables with a single API call
DynamoDB: Capacity managed for you

Provisioned

Govern max consumption

Auto Scaling

Set a minimum

On-Demand

No limit

Start at zero
DynamoDB Accelerator (DAX)

Features

• **Fully managed, highly available:** Handles all software management, fault tolerant, replication across multi-AZs within a Region

• **DynamoDB API compatible:** Seamlessly caches DynamoDB API calls, no application rewrites required

• **Write-through:** DAX handles caching for writes

• **Flexible:** Configure DAX for one table or many

• **Scalable:** Scales-out to any workload with up to 10 read replicas

• **Manageability:** Fully integrated AWS service: Amazon CloudWatch, Tagging for DynamoDB, AWS Console

• **Security:** Amazon VPC, AWS IAM, AWS CloudTrail, AWS Organizations
### DynamoDB

**Advancements over the last 21 months**

<table>
<thead>
<tr>
<th>February 2017</th>
<th>April 2017</th>
<th>April 2017</th>
<th>June 2017</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Time To Live (TTL)</strong></td>
<td><strong>VPC endpoints</strong></td>
<td><strong>DynamoDB Accelerator (DAX)</strong></td>
<td><strong>Auto scaling</strong></td>
</tr>
<tr>
<td>November 2017</td>
<td>November 2017</td>
<td>November 2017</td>
<td>March 2018</td>
</tr>
<tr>
<td><strong>Global tables</strong></td>
<td><strong>On-demand backup</strong></td>
<td><strong>Encryption at rest</strong></td>
<td><strong>Point-in-time recovery</strong></td>
</tr>
<tr>
<td>June 2018</td>
<td>August 2018</td>
<td>November 2018</td>
<td>November 2018</td>
</tr>
<tr>
<td><strong>99.999% SLA</strong></td>
<td><strong>Adaptive capacity</strong></td>
<td><strong>Transactions</strong></td>
<td><strong>On-demand</strong></td>
</tr>
</tbody>
</table>

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Amazon DynamoDB

A serverless database for applications that need high performance at any scale

DynamoDB
@dynamodb

Amazon DynamoDB is a serverless database for applications that need high performance at any scale. Follow us for the latest about DynamoDB!

Thomas Henson @henson_tm · Jan 6
Learn the top 11 CLI commands every DynamoDB Engineer should know bit.ly/2KY9065 AWS DynamoDB NoSQL @dynamodb

11 Fundamental DynamoDB CLI Commands - Tho...
Ready to manage DynamoDB from the AWS CLI? Step by Step through the 11 Fundamental commands with examples in this post every AWS Dynamodb Engineer ...
## SQL vs NoSQL

<table>
<thead>
<tr>
<th></th>
<th>SQL</th>
<th>NoSQL</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Optimized for storage</strong></td>
<td>Optimized for compute</td>
<td></td>
</tr>
<tr>
<td>Normalized/relational</td>
<td>Denormalized/hierarchical</td>
<td></td>
</tr>
<tr>
<td>Ad hoc queries</td>
<td>Instantiated views</td>
<td></td>
</tr>
<tr>
<td>Scale vertically</td>
<td>Scale horizontally</td>
<td></td>
</tr>
<tr>
<td>Good for OLAP</td>
<td>Built for OLTP at scale</td>
<td></td>
</tr>
</tbody>
</table>
Scalable and relational

with Amazon Aurora
Old world commercial relational databases

- Very expensive
- Proprietary
- Lock-in
- Punitive licensing
- You’ve got mail
Moving to open database engines

MySQL

PostgreSQL

MariaDB

Enterprise-grade performance and reliability
Amazon Aurora

MySQL and PostgreSQL compatible relational database built for the cloud
Performance and availability of commercial-grade databases at 1/10th the cost

Performance and scalability
5x throughput of standard MySQL and 3x of standard PostgreSQL; scale-out up to 15 read replicas

Availability and durability
Fault-tolerant, self-healing storage; six copies of data across three AZs; continuous backup to S3

Highly secure
Network isolation, encryption at rest/transit

Fully managed
Managed by RDS: no hardware provisioning, software patching, setup, configuration, or backups
Aurora: Scale-out, Distributed architecture

- Push Log applicator to Storage
  - “The log is the database”

- 4/6 Write Quorum & Local tracking
  - Write performance
  - Read scale out
  - AZ + 1 failure tolerance
  - Instant database redo recovery

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Write and read throughput

Aurora MySQL is 5x faster than MySQL

Write Throughput

Read Throughput

Using Sysbench with 250 tables and 200,000 rows per table on R4.16XL
Read scale out

**MySQL Read Scaling**

- **MySQL Master**
  - 70% Write
  - 30% Read
  - Data Volume

- **MySQL Replica**
  - 70% Write
  - 30% New Reads
  - Data Volume

**Amazon Aurora Read Scaling**

- **Aurora Master**
  - 70% Write
  - 30% Read
  - Shared Multi-AZ Storage

- **Aurora Replica**
  - 70% Write
  - 30% New Reads
  - 100% New Reads

- **PAGE CACHE UPDATE**

**Logical using complete changes**

**Same write workload**

**Independent storage**

**Physical using delta changes**

**NO writes on replica**

**Shared storage**
Aurora Multi-Master Architecture

- Orange Master
  - SQL Transactions
  - Caching

- Blue Master
  - SQL Transactions
  - Caching

Cluster Services
- Membership
- Replication
- Heartbeat
- Metadata

Decoupled System
- Optimistic Conflict Resolution
- Microservices Architecture
- No Pessimistic Locking
- No Global Ordering
- No Global Commit-Co-ordination

Shared Storage Volume

AZ1
- 1
- 2
- 3

AZ2
- 1
- 2
- 3

AZ3
- 1
- 2
- 3
Driving down query latency – Parallel Query

- Parallel, Distributed processing
- Push-down processing closer to data
- Reduces buffer pool pollution
“AZ+1” failure tolerance

Why?
- In a large fleet, always some failures
- AZ failures have "shared fate"

How?
- 6 copies, 2 copies per AZ
- 2/3 quorum will not work
Database backtrack

Backtrack brings the database to a point in time without requiring restore from backups

- Backtracking from an unintentional DML or DDL operation
- Backtrack is not destructive. You can backtrack multiple times to find the right point in time
Global replication
Faster disaster recovery and enhanced data locality
Performance Insights

Dashboard showing database load
- Easy – e.g. drag and drop
- Powerful – drill down using zoom in

Identifies source of bottlenecks
- Sort by top SQL
- Slice by host, user, wait events

Adjustable time frame
- Hour, day, week, month
- Up to 2 years of data; 7 days free
Aurora Serverless

Responds to your application load automatically

Scale capacity up and down in < 10 seconds

New instance has warm buffer pool

Multi-tenant proxy is highly available
How does it work . . .

>70% CPU utilization or >90% max connections
Introducing Web Service Data API

Access your database from Lambda applications

SQL statements packaged as HTTP requests

Connection pooling managed behind proxy

Aurora Serverless
Real-time applications

with Amazon ElastiCache
Internet-scale Apps Need Low Latency and High Concurrency

Users: 1M+
Data volume: TB-PB-EB
Locality: Global
Performance: Milliseconds -Microseconds
Request Rate: Millions
Access: Mobile, IoT, Devices
Scale: Up-Out-In
Economics: Pay as you go
Developer access: Instant API access

Gaming leaderboards
Financial trading
Social media
Ride hailing
Dating
Media streaming
Session stores
Introducing Amazon ElastiCache

Fully-managed, Redis or Memcached compatible, low-latency, in-memory data store

- **Extreme Performance**: In-memory data store and cache for sub-millisecond response times
- **Fully Managed**: AWS manages all hardware and software setup, configuration, monitoring
- **Easily Scalable**: Read scaling with replicas. Write and memory scaling with sharding. Non-disruptive scaling

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.
What’s New: Redis

Redis 5.0

- Redis Streams
- SortedSets now have LIST capabilities (POP and BLOCK)
- HyperLogLogs has an optimized algorithm
- Speed Improvements (Jemalloc additions, etc.)
- Active Defragmentation
- Added In-line HELP command for redis-cli
- Native TLS Integration Redis (ElastiCache)

Memcached 1.5.10

- Automated Slab rebalancing
- LRU crawler to background-reclaim memory
- Faster hash table lookups with murmur3 algorithm
New: Amazon ElastiCache 250 node support

When 9.5 TiB is not enough!

Example 1:

• Assume 125 shards made of 1 Primary + 1 Replica = 250 nodes
• Assume R5.24xlarge (635.61 GiB)
• Cluster memory 635.61 GiB X 125 = ~80 TiB = ~88 TB

Example 2:

• Assume 250 shards made of 1 Primary + 0 Replica = 250
• Assume R5.24xlarge (635.61 GiB)
• Cluster memory 635.61 GiB X 250 = ~159 TiB = ~170 TB
Redis Overview

- Fast
  <1ms latency for most commands

- Open source

- Easy to learn

- Highly available
  Replication

- Atomic operations
  Supports transactions

- In-memory
  key-value store

- Powerful
  ~200 commands, Lua scripting, Geospatial, Pub/Sub

- Various data structures
  Strings, lists, hashes, sets, sorted sets, bitmaps, streams, and HyperLogLogs

- Backup/Restore
  Enables snapshotting
Cluster sizing best practices

• **In-Memory Storage**
  - Recommended: Memory needed + 25% reserved memory (for Redis) + some room for growth (optional 10%)
  - Optimize using eviction policies and **TTLs**
  - Scale up or out when before reaching max-memory using CloudWatch alarms
  - Use **memory optimized nodes** for cost effectiveness (R5 support)

• **Performance**
  - Benchmark operations using Redis Benchmark tool
    - For more READIOPS—Add replicas
    - For more WRITEIOPS—Add shards (scale out)
    - For more network —Use network optimized instances and scale out
  - Use pipelining for bulk reads/writes
  - Consider **Big(O) time complexity** for data structure commands

• **Cluster Isolation (apps sharing key space)—Choose a strategy that works for your workload**
  - Identify what kind of isolation is needed based on the workload and environment
  - Isolation: No Isolation $ | Isolation by Purpose $$ | Full Isolation $$$
GE’s Predix Platform Powered by ElastiCache Redis

Using ElastiCache Redis with Open Service Broker, Predix Platform from GE Digital allows developers to easily create Redis clusters with standard, pre-configured parameters, sizing and network security.

Developers build container-based stateless applications on AWS and ElastiCache is used to manage session state for these applications.

The architecture makes is easy and simple for developers to build applications.

GE is the world’s largest digital industrial company. We use [ElastiCache Redis] to make it super easy and simple for developers to use Amazon services. Amazon ElastiCache team implemented the Redis AUTH feature in four regions in two months enabling application level security.”

– Amulya Sharma
Senior Staff Software Engineer
GE Digital

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Expedia’s Real-time Analytics with ElastiCache

Expedia is a leader in the $1 trillion travel industry, with an extensive portfolio that includes some of the world’s most trusted travel brands.

Expedia’s real-time analytics application collects data for its “test & learn” experiments on Expedia sites. The analytics application processes ~200 million messages daily.

"With ElastiCache Redis as caching layer, the write throughput on DynamoDB has been set to 3500, down from 35000, reducing the cost by 6x."

– Kuldeep Chowhan
Engineering Manager, Expedia
Time-series applications

with Amazon Timestream
A sequence of data points recorded over a time interval.
Amazon Timestream (Preview)

Fast, scalable, and fully managed time series database

1,000x faster at 1/10th the cost of relational databases

Trillions of daily events

Analytics optimized for time series data

Serverless

Collect fast moving time-series data from multiple sources at the rate of millions of inserts per second

Capable of processing trillions of events daily; the adaptive query processing engine maintains steady, predictable performance

Built-in analytics for interpolation, smoothing, and approximation to identify trends, patterns, and anomalies

No servers to manage; time-consuming tasks such as hardware provisioning, software patching, setup, & configuration done for you
Summary
Our approach

- Architect services ground-up for the cloud and for the explosion of data
- Offer a portfolio of purpose-built services, optimized for your workloads
- Help you innovate faster through managed services
- Provide services that help you migrate existing apps and databases to the cloud
# When to Use Which Services

<table>
<thead>
<tr>
<th>Situation</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Existing application</td>
<td>Use your existing engine on RDS</td>
</tr>
<tr>
<td>• MySQL</td>
<td>→ Amazon Aurora, RDS for MySQL</td>
</tr>
<tr>
<td>• PostgreSQL</td>
<td>→ Amazon Aurora, RDS for PostgreSQL</td>
</tr>
<tr>
<td>• MariaDB</td>
<td>→ Amazon Aurora, RDS for MariaDB</td>
</tr>
<tr>
<td>• Oracle</td>
<td>→ Use SCT to determine complexity → Amazon Aurora, RDS for Oracle</td>
</tr>
<tr>
<td>• SQL Server</td>
<td>→ Use SCT to determine complexity → Amazon Aurora, RDS for SQL Server</td>
</tr>
<tr>
<td>New application</td>
<td>• If you can avoid relational features → DynamoDB</td>
</tr>
<tr>
<td></td>
<td>• If you need relational features → Amazon Aurora</td>
</tr>
<tr>
<td>In-memory store/cache</td>
<td>• Amazon ElastiCache</td>
</tr>
<tr>
<td>Time series data</td>
<td>• Amazon Timestream</td>
</tr>
<tr>
<td>Track every application change, crypto verifiable.</td>
<td>• Amazon Quantum Ledger Database (QLDB)</td>
</tr>
<tr>
<td>Have a central trust authority</td>
<td></td>
</tr>
<tr>
<td>Don’t have a trusted central authority</td>
<td>• Amazon Managed Blockchain</td>
</tr>
<tr>
<td>Data Warehouse &amp; BI</td>
<td>• Amazon Redshift, Amazon Redshift Spectrum, and Amazon QuickSight</td>
</tr>
<tr>
<td>Adhoc analysis of data in S3</td>
<td>• Amazon Athena and Amazon QuickSight</td>
</tr>
<tr>
<td>Apache Spark, Hadoop, HBase (needle in a haystack type queries)</td>
<td>• Amazon EMR</td>
</tr>
<tr>
<td>Log analytics, operational monitoring, &amp; search</td>
<td>• Amazon Elasticsearch Service and Amazon Kinesis</td>
</tr>
</tbody>
</table>
Thank you!

Julio Faerman
@faermanj
Please complete the session survey.