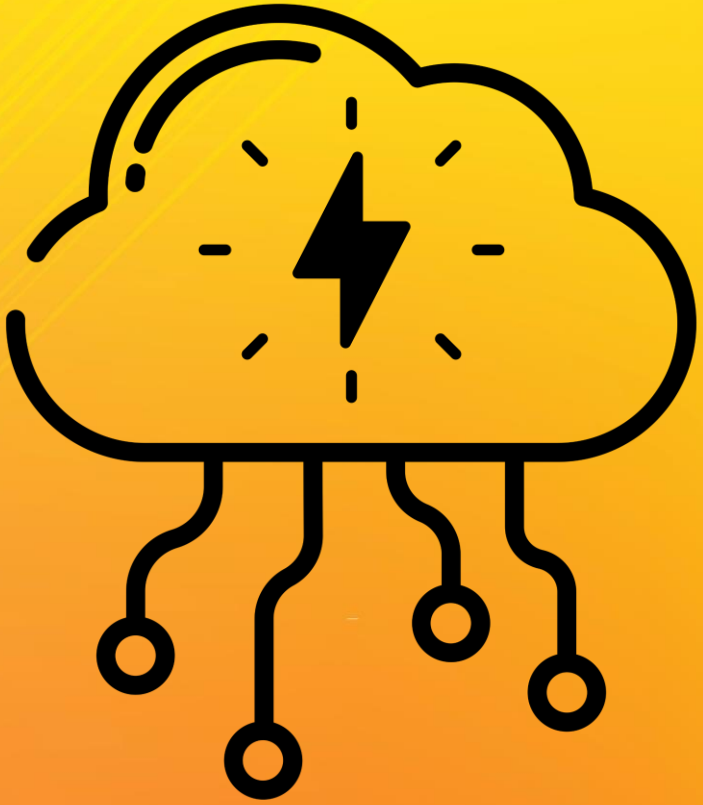
Caching in Cloud

Why cache?

Improved Performance

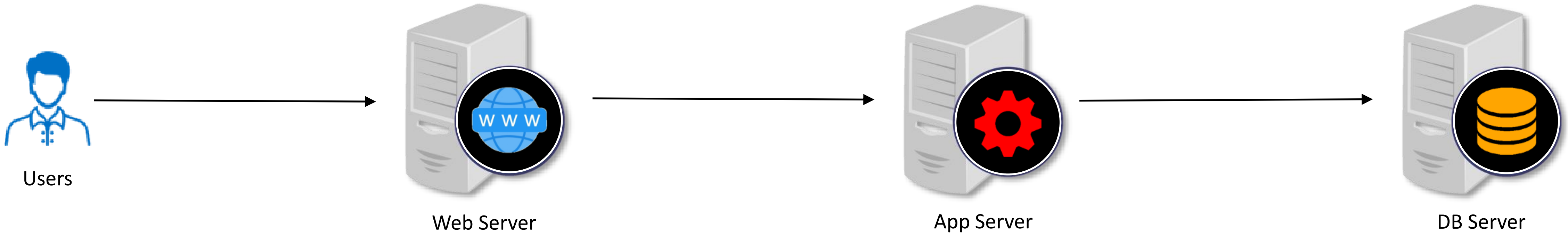
Additional Security

Cost Savings



Caching in AWS

A Typical Web Application



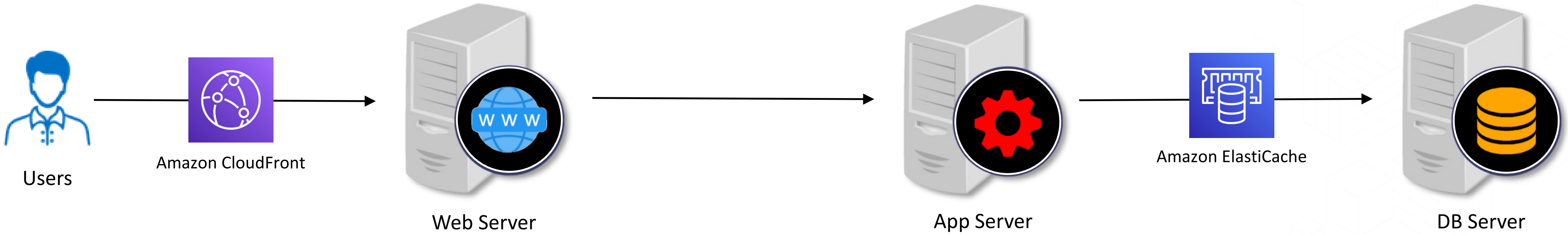
Users

Web Server

App Server

DB Server

Caching on AWS



Users

Amazon CloudFront

Web Server

App Server

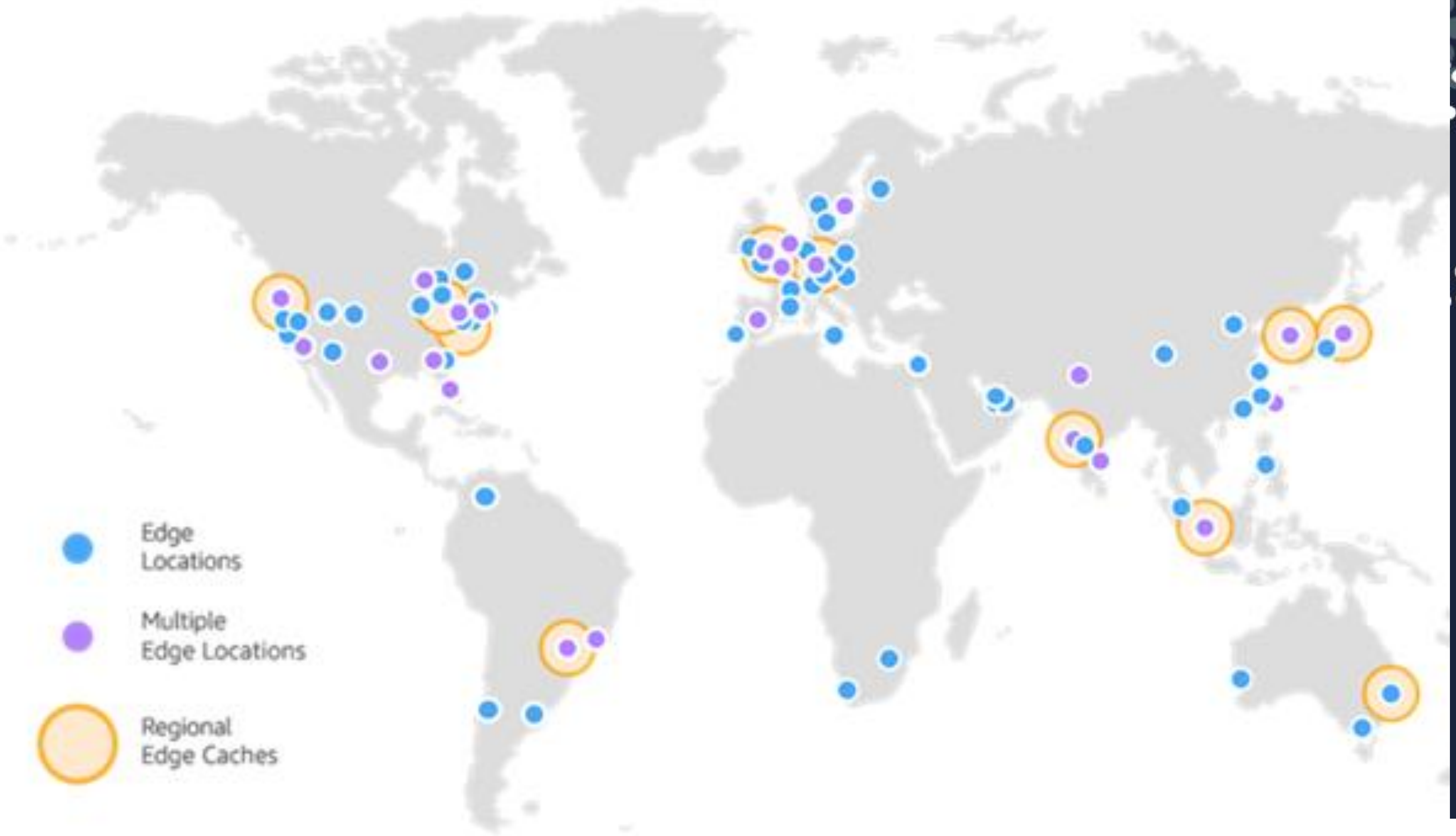
Amazon ElastiCache

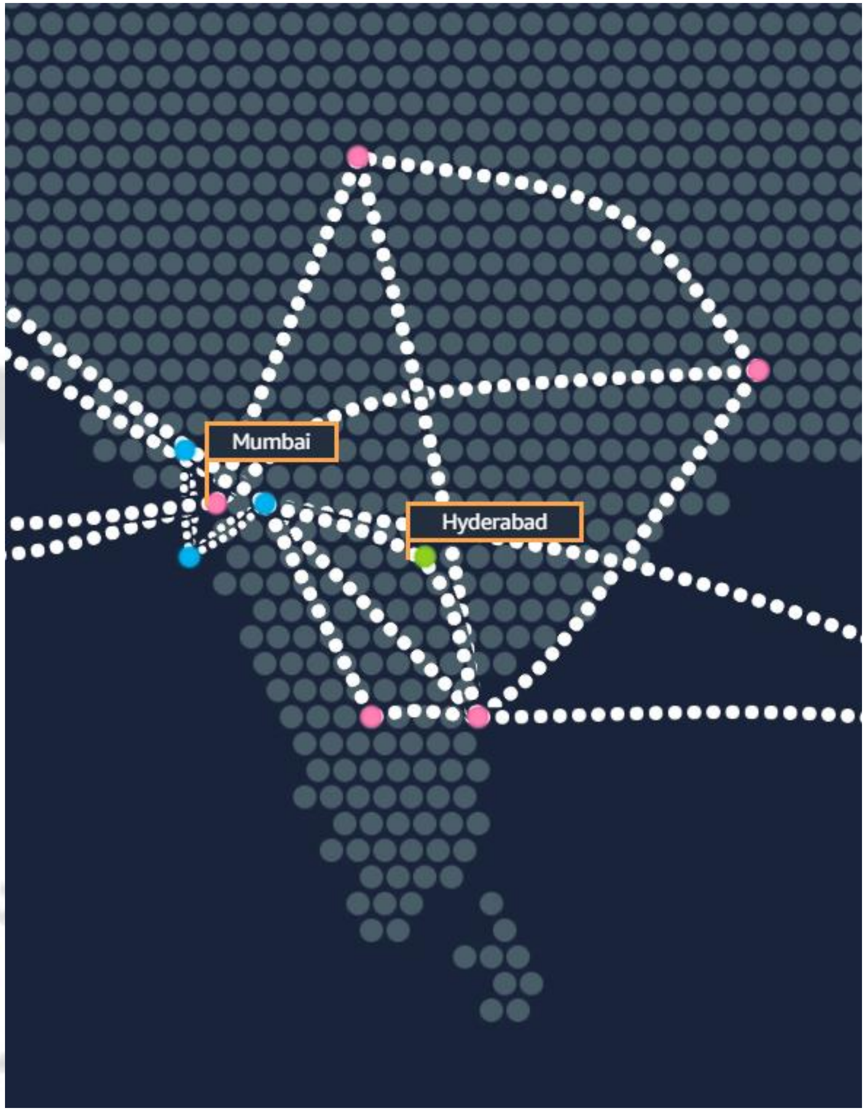
DB Server

Amazon CloudFront

Amazon CloudFront – Global Content Delivery Network

- Uses Edge Locations for caching





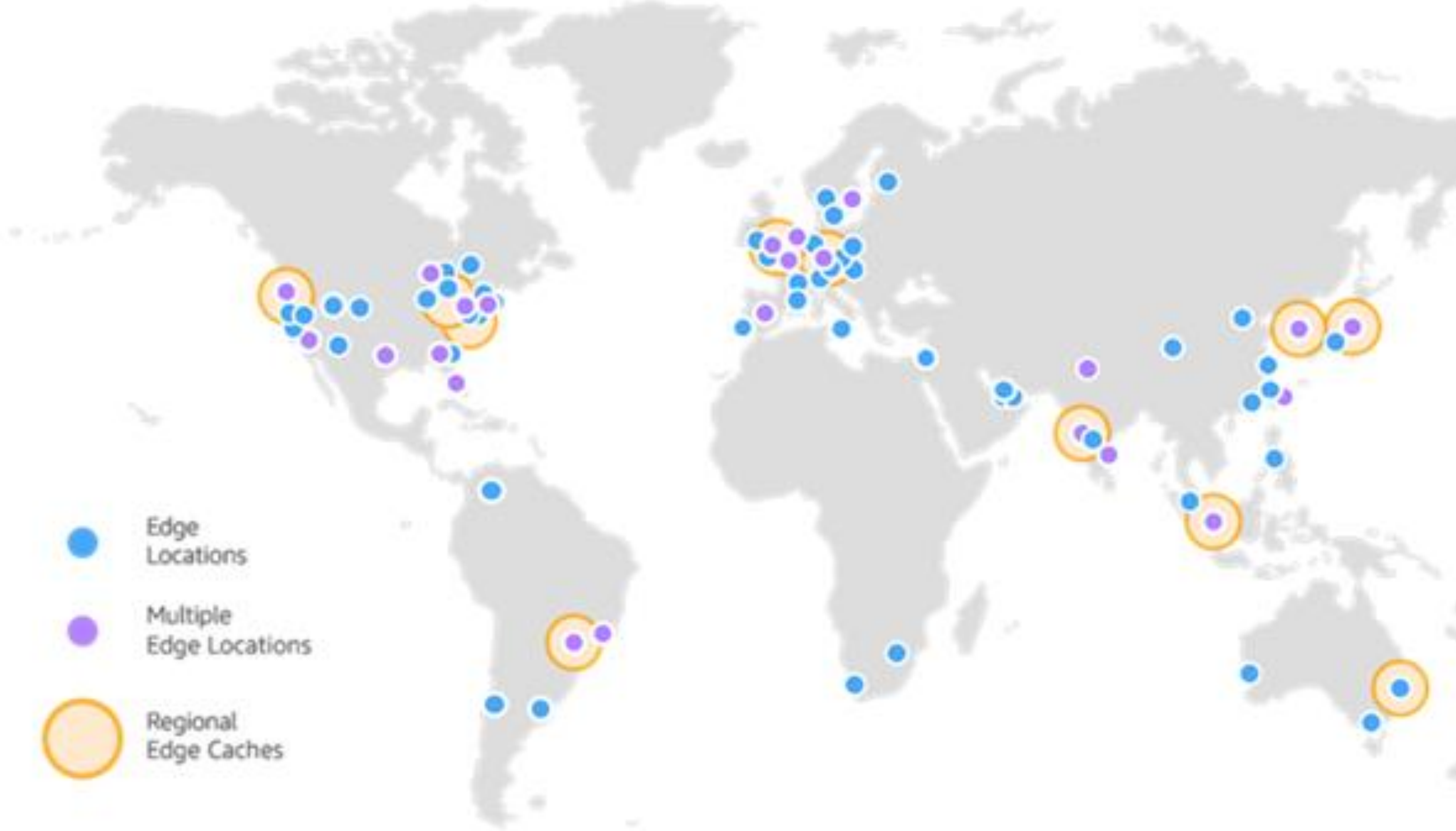
Region vs. Edge Locations (Analogy)

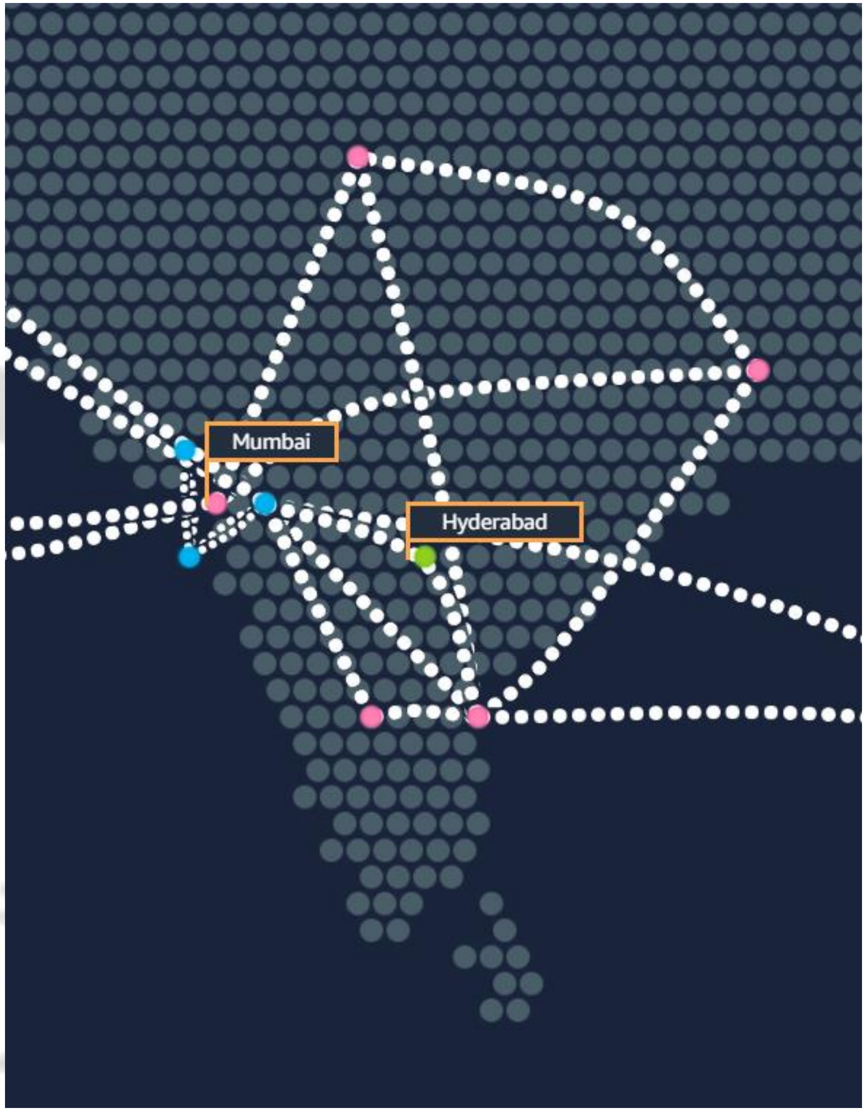
Large-format stores
(Everything under one roof)

Small-format stores
(everyday essentials)

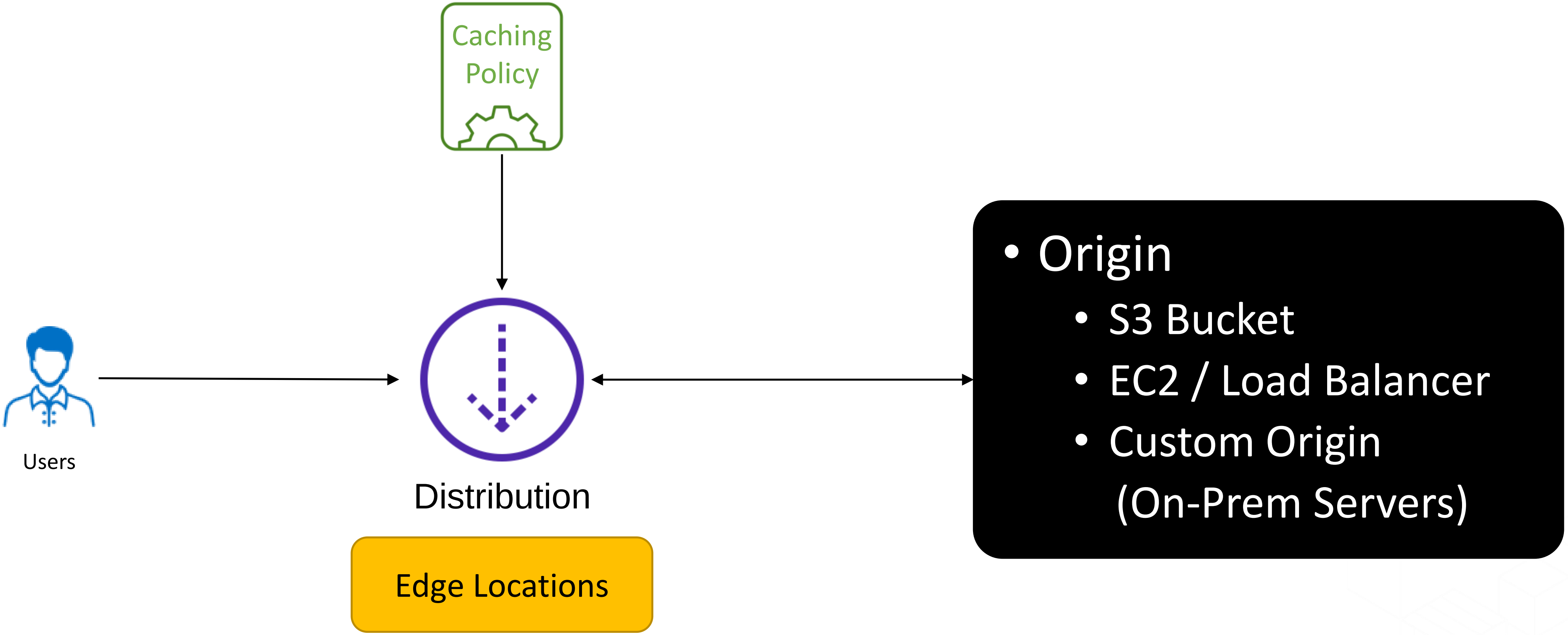
Amazon CloudFront

- Uses Edge Locations for caching
- DDoS Protection
- Lambda@Edge





Components



Caching Policy

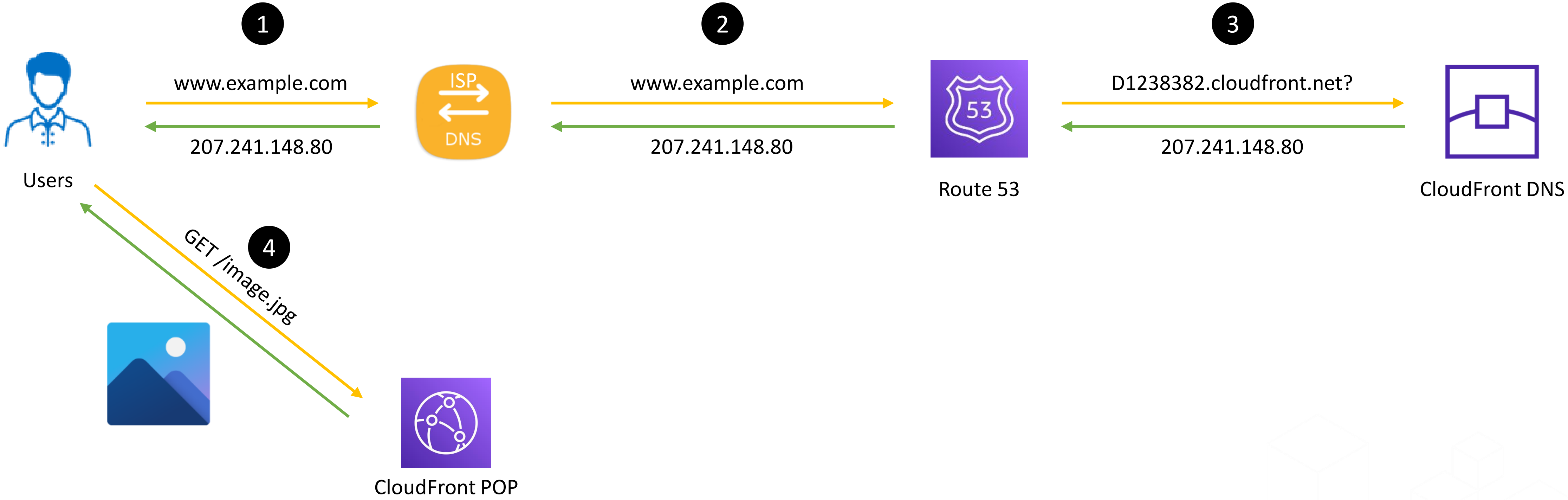
Users

Distribution

Edge Locations

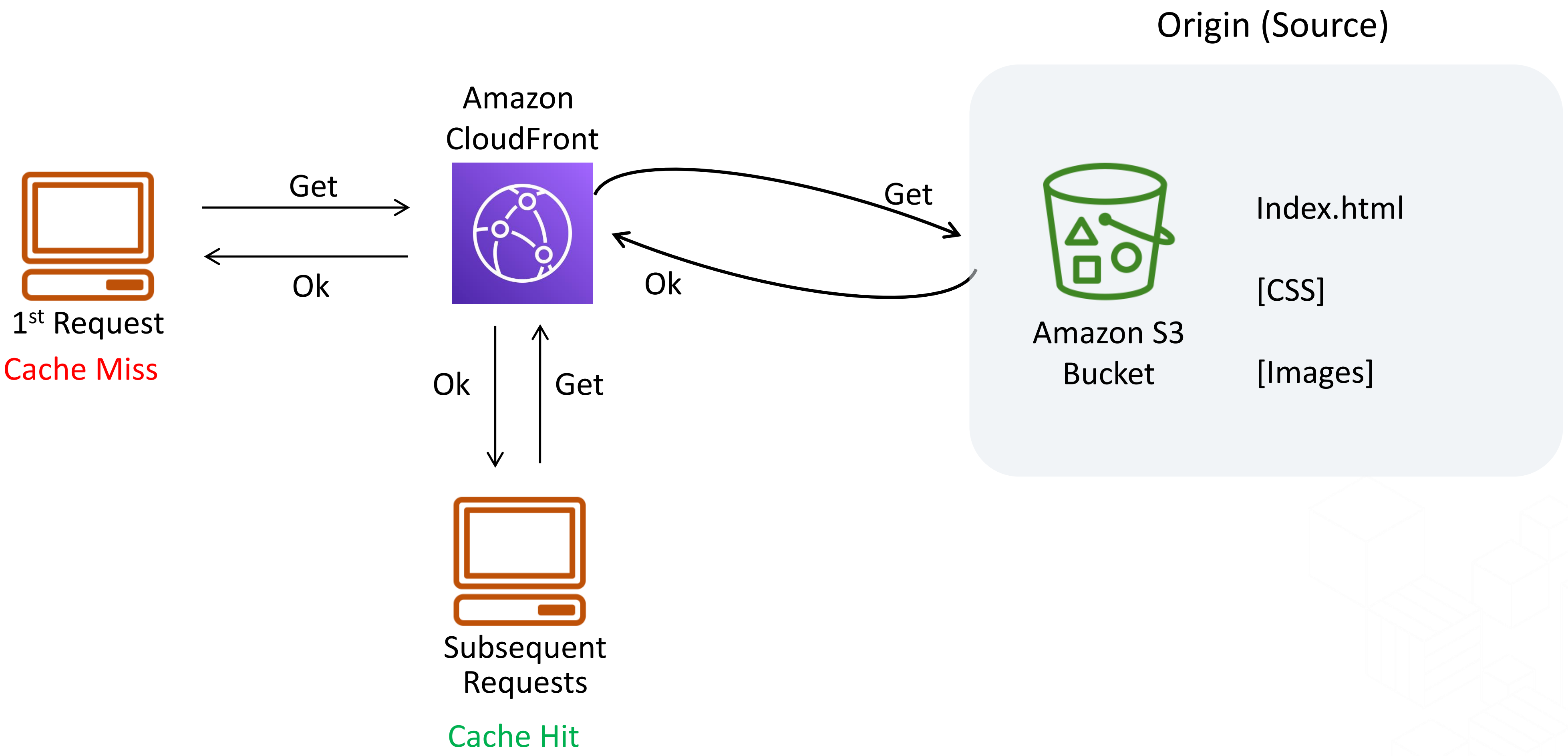
- Origin
  - S3 Bucket
  - EC2 / Load Balancer
  - Custom Origin (On-Prem Servers)

HTTP Request – www.example.com/image.jpg



Amazon CloudFront

How Amazon CloudFront works?



Database Caching

Database Caching

Cache Endpoint
Hostname**:**Port + Credentials

Memory (RAM)

Cache

Microsecond Latency

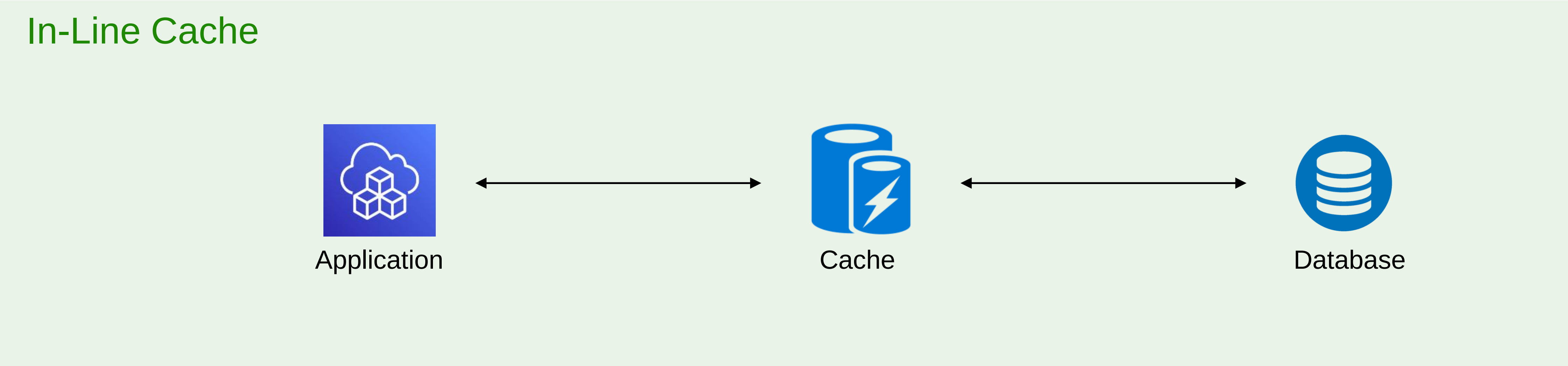
Database Endpoint
Hostname**:**Port + Credentials

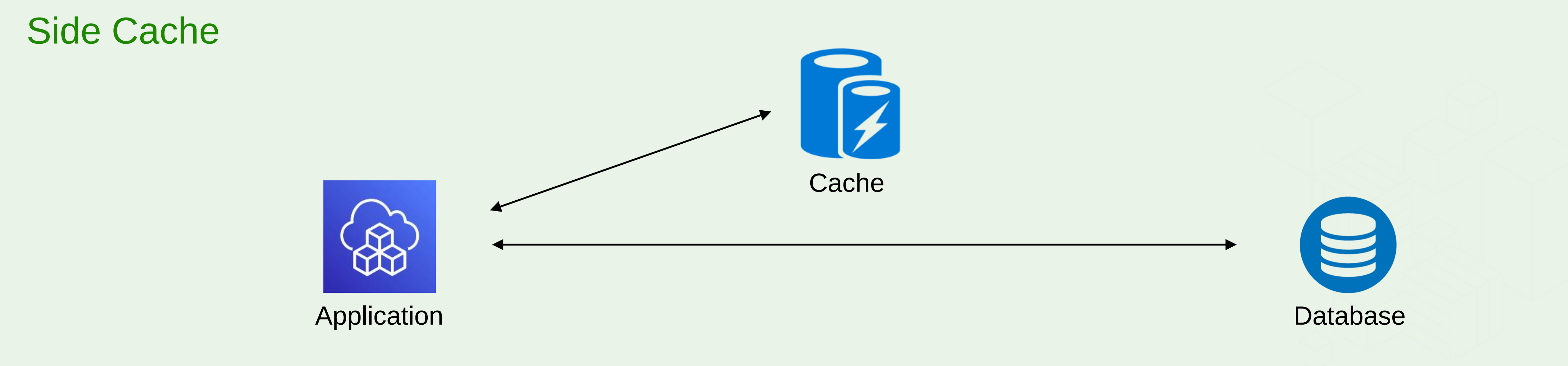
Disk

Database

Millisecond Latency

In-Line Cache vs Side Cache





Database Caching

- Open source caching engine
  - Memcached
  - Redis

Amazon Elasticache

Amazon Elasticache

Fully Managed

Extreme Performance

Scalable

Secure & Compliant

Amazon Elasticache – Engine choice

- Amazon Elasticache for Memcached
- Amazon Elasticache for Redis

ElastiCache for Memcached

Memcached
Cluster

ElastiCache for Redis

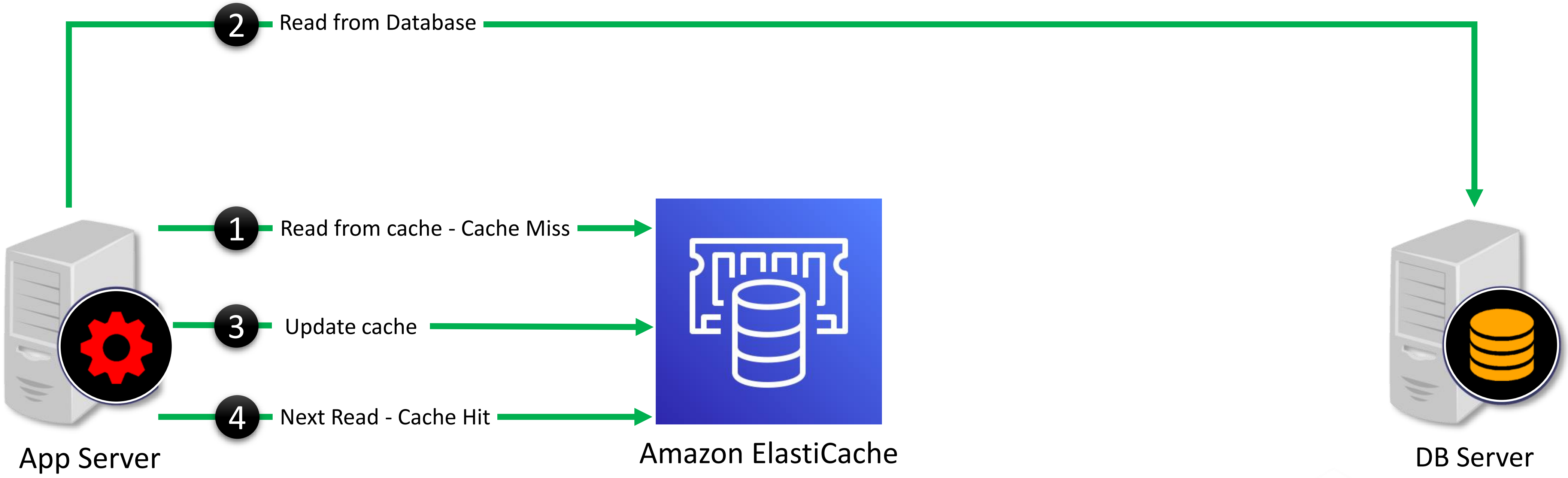
redis
Cluster

Memcached vs. Redis

	Memcached	Redis
Primary benefit	Simplicity	Rich set of features
Advanced data structures	Not supported	Supported – strings, lists, sets, sorted sets, hashes, bit arrays, and hyperloglogs.
High availability (Replication)	Not supported	Supported
Backup and Restore (Data Persistence)	Not supported	Supported
Pub/Sub capabilities	Not supported	Supported
Transactions	Not supported	Supported

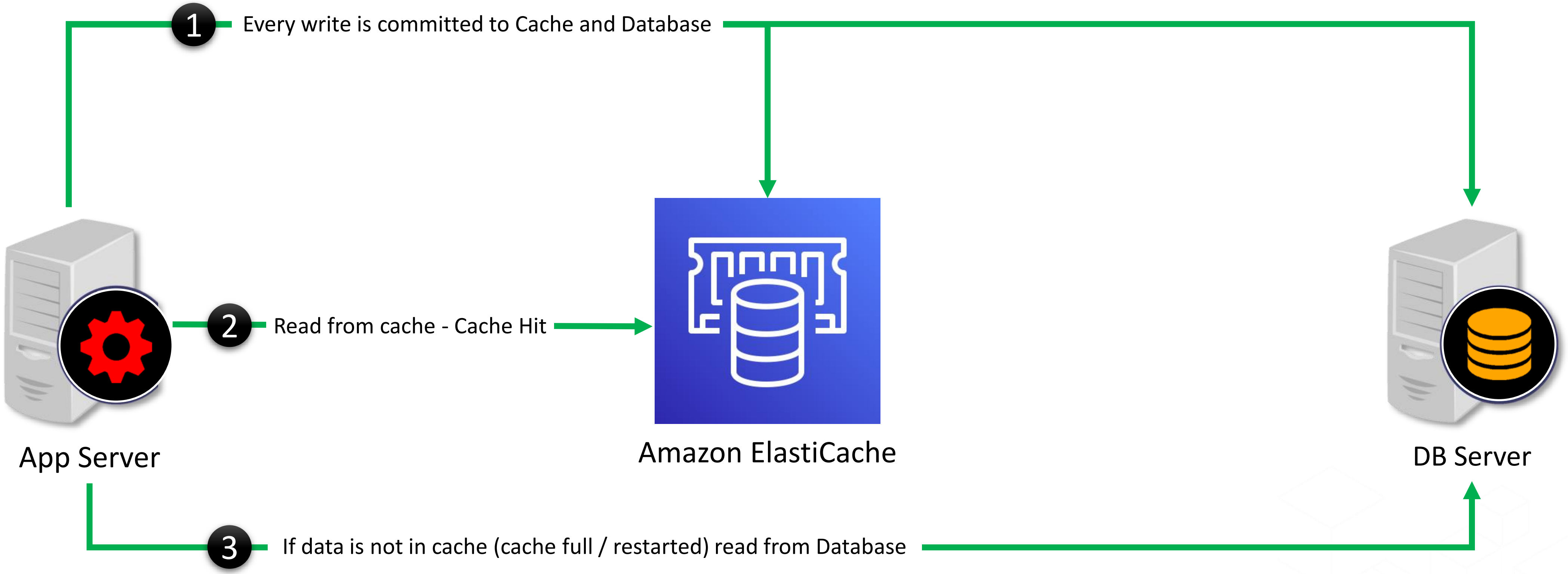
Caching Strategies

Lazy Loading Strategy



Advantage	Disadvantage
Only requested data is cached	Cache miss penalty. Each cache miss results in 3 trips
Node failures are not fatal	Application may receive stale data

Write Through Strategy



1 Every write is committed to Cache and Database

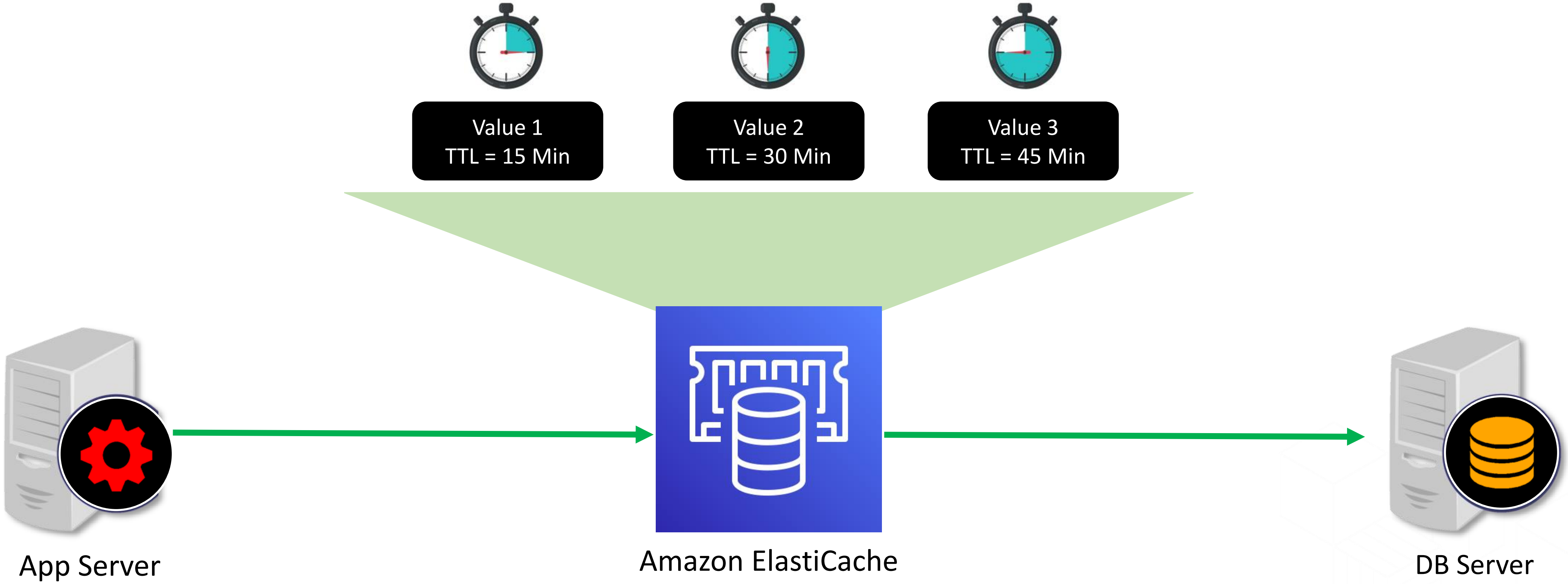
2 Read from cache - Cache Hit

3 If data is not in cache (cache full / restarted) read from Database

Advantage
The data in the cache is never stale
Write latency better tolerated by customers vs. read latency

Disadvantage
Write penalty - Every write involves two trips
Unused data in cache

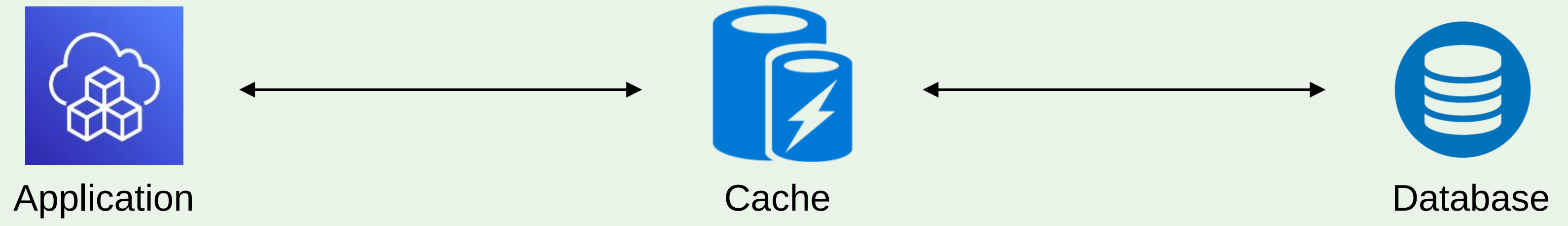
Cache Expiry - Time-to-Live (TTL)



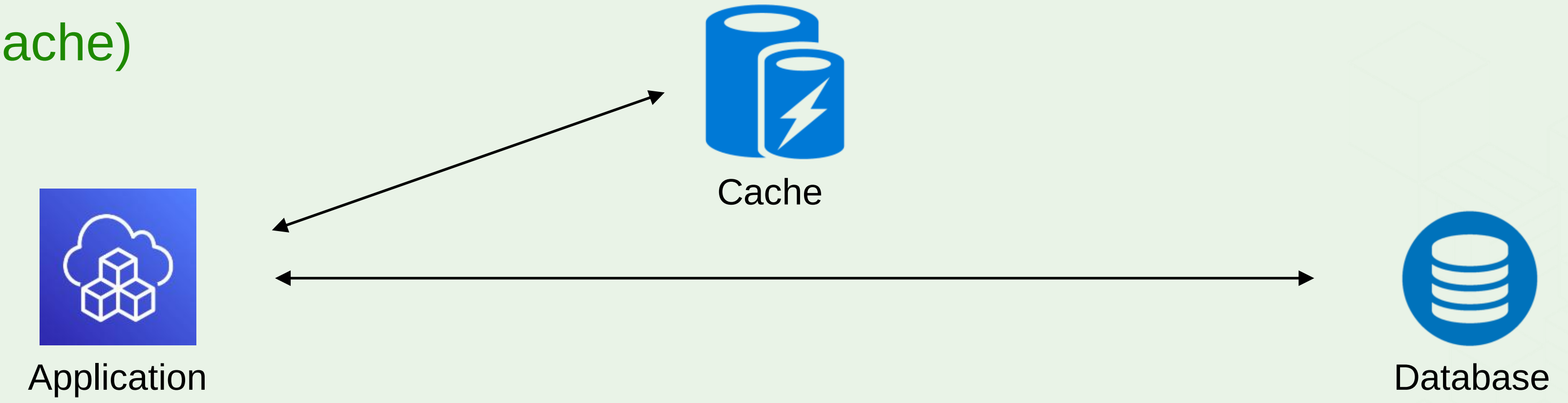
Amazon DynamoDB Accelerator (DAX)

In-Line Cache vs Side Cache

In-Line Cache
(DynamoDB Accelerator)

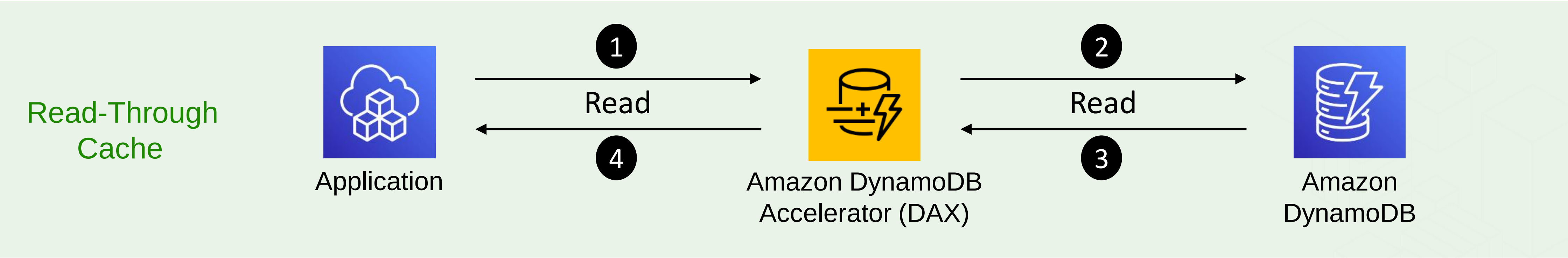


Side Cache
(Amazon Elasticache)

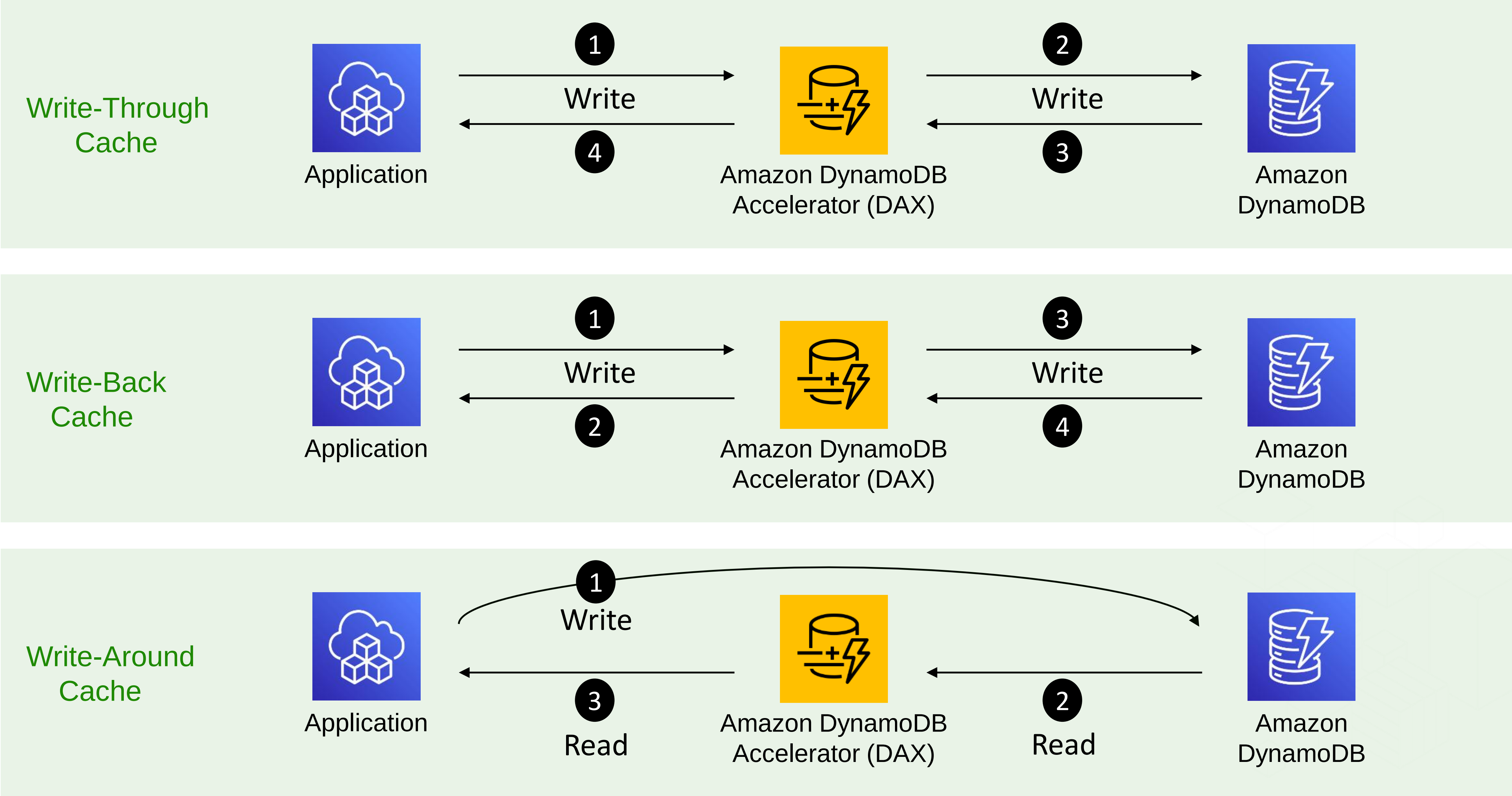


Amazon DynamoDB Accelerator (DAX)

- Fully managed, highly available in-memory cache for Amazon DynamoDB
- Response times in microseconds (instead of milliseconds)
- API compatible with DynamoDB - you can simply point your existing DynamoDB application at the DAX endpoint, no need to rewrite the application.
- Security – Amazon VPC, AWS IAM, CloudTrail, AWS Organizations



Write Operations



Write-Through Cache

Application

1 Write

2 Write

3

4

Amazon DynamoDB Accelerator (DAX)

Amazon DynamoDB

Write-Back Cache

Application

1 Write

2

3 Write

4

Amazon DynamoDB Accelerator (DAX)

Amazon DynamoDB

Write-Around Cache

Application

1 Write

2 Read

3 Read

Amazon DynamoDB Accelerator (DAX)

Amazon DynamoDB