Careem

MINUTES TO SECONDS

How Careem uses volatile storage to power their dispatching algorithms

WHAT TO EXPECT?

- Ingest fast moving data with in-memory storage
- Storing volatile data for efficient lookups
- Real-time decision making with sub millisecond lookups
- Recommended practices for maximum utilization of resources

It comes from the Arabic word Kareem – meaning **generosity**

LET'S BE CAREEM (GENEROUS)

The reason we exist

To simplify and improve the lives of people,
and build an awesome organization that inspires

BASICS – GROUND CONDITIONS

- Infrastructure
- High speeds in some cities
- Far away exits
- Social norms

BASICS - CAPTAINS

- Central to our brand
- Where did the name come from?
    - Person in command
    - Leader of a team
- Entrepreneurs

MARKETPLACE

- Reliability
- Match quality
- Tracking

BASICS - CONNECTIVITY

- Low-end Android devices
- Bad GPS sensors
- Extreme temperatures – 46℃ to 56℃
- Limited bandwidth

METRICS

- ETA
- Time needed to make the best match (time to match)
- Age of captain location [ping]
- Ratio of requests matched

REQUIREMENTS

- Find the best captain in the minimum amount of time
- Ability to provide an upfront ETA (promised ETA)
- Lowest possible delta between promised and actual ETAs
- Ability to look up any captain’s location and status for tracking

Ping as fast as possible

TAKE 1

Captain app

Pings (json) over REST api

Stored in a MySQL table

SCALE

19
CITIES

450,000
CUSTOMERS

9,000
CAPTAINS

SCALE

945,000
ETA Requests

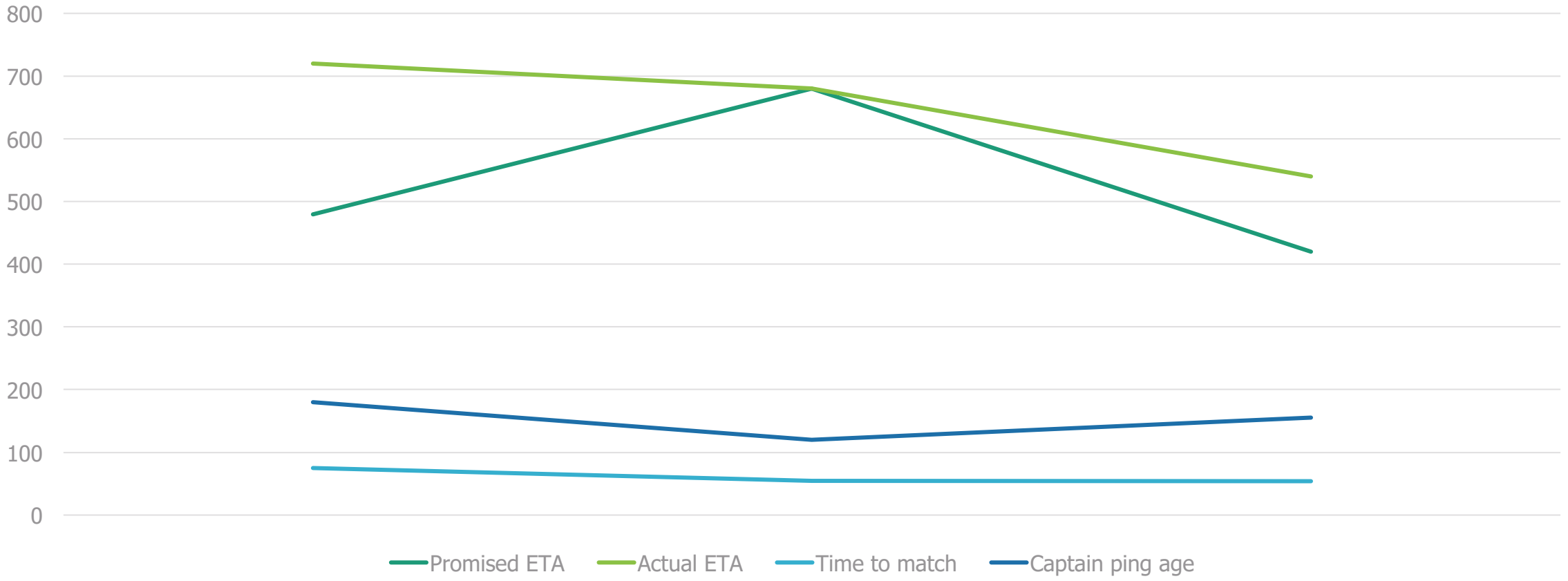
13 M
PINGS

16 M
LOOKUPS

ISSUES: TAKE 1

- Continuous deadlocks and performance issues
  https://bugs.mysql.com/bug.php?id=48652
- MySQL 5.6 - no geo-spatial support
- Finding nearby cars was not trivial
- For stability, we had to reduce the ping frequency to 60s
- Affected our ETAs and customer experience
- Cost impact – had to use over provisioned resources

PERFORMANCE



PERFORMANCE

40%

RELIABILITY

95%

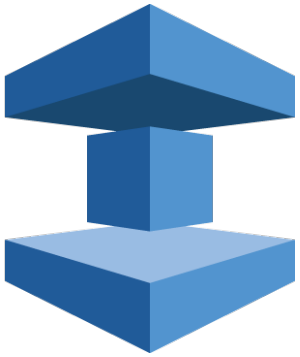
UPTIME

LEARNINGS: TAKE 1

- Data is volatile – only the most recent version is relevant and the continuous frequency rebuilds the data
- Locks
- Need a mechanism to support µs lookups
- Easy, efficient and super fast nearby-car lookups
- Ability to alter object schema at will
- Need a buffer to deal with traffic spikes
- Need historical time series data against only 2 anchors - captains and booking
- Need a mechanism of representing coordinates as a scalar value
- Speed is king

TOOLS

ElastiCache for Redis



Managed In-Memory Nosql Service

SQS

Queue for Captain Apps to Send Pings to

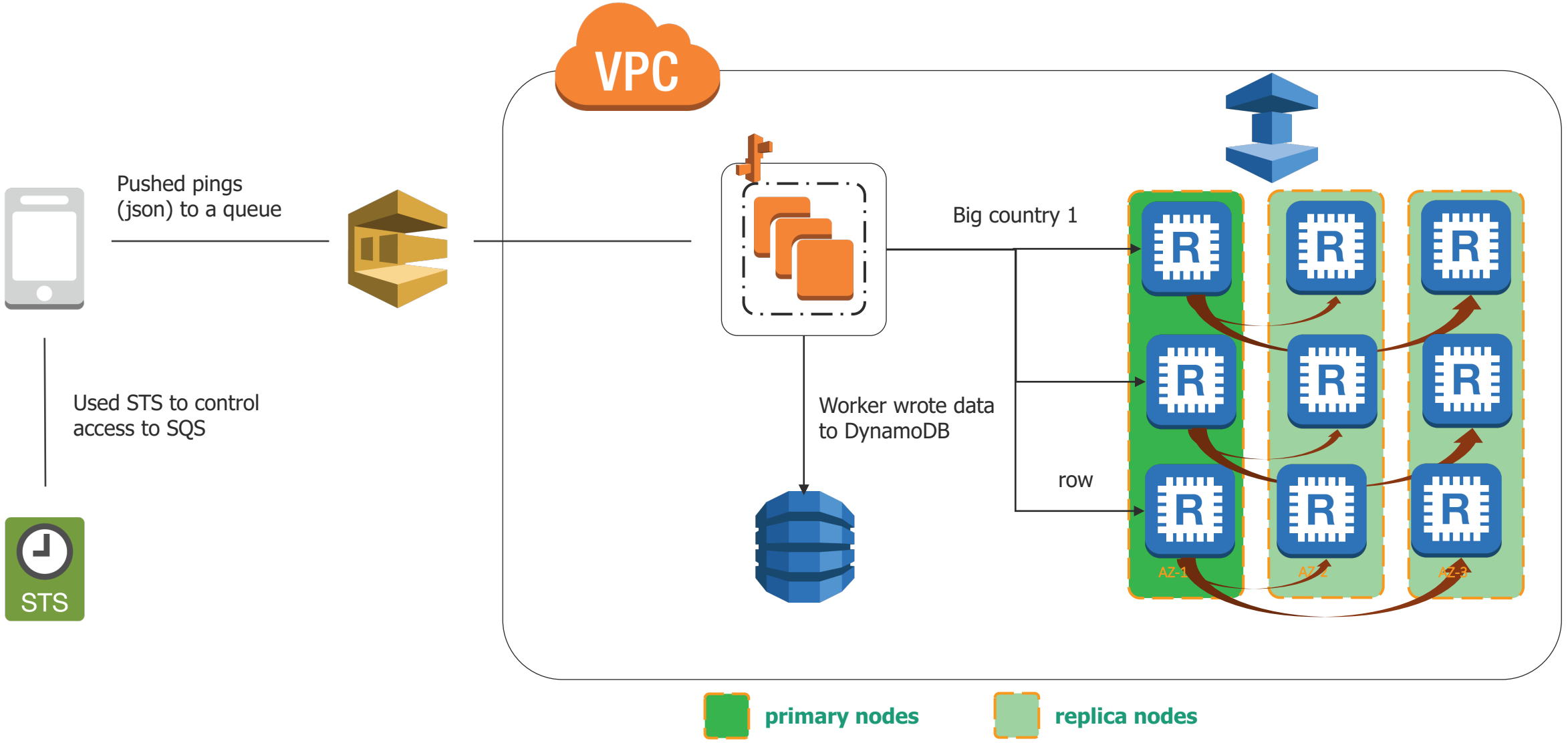
DynamoDB

NoSQL persistence data store for storing historical data against captains and bookings

WHY REDIS?

- Single threaded lock free architecture
- Rich data structure set
  - String
  - Sorted Sets
- Data structures provide built-in operations that process data optimally at the database level rather than the application level
- Pipelining
- Primary – replica configuration – for scaling out reads and fail-over support

TAKE 2


VPC
Pushed pings
(json) to a queue
Used STS to control
access to SQS
STS
Big country 1
Worker wrote data
to DynamoDB
row
R
AZ-1
AZ-2
AZ-3
primary nodes
replica nodes

DATA STRUCTURES AT WORK: CAPTAIN INDEXING

Key	Value
captain:234	{”captainId”: 234, ….. ”lastUpdated”:1506349247000}
captain:89	{”captainId”: 89, ….. ”lastUpdated”:1506349202000}
captain:236	{”captainId”: 236, ….. ”lastUpdated”:1506349247000}
captain:78	{”captainId”: 78, ….. ”lastUpdated”:1506349143000}

DATA STRUCTURES AT WORK: GEO INDEXING

Key: geohash:aabbc:product:12

Value	Score
captain:234	1506349247000
captain:89	**1506349202000**
captain:78	1506349143000

Key: geohash:aabbd:product:12

Value	Score
captain:236	1506349247000
captain:77	1506349143000

DATA STRUCTURES AT WORK: GEO INDEXING

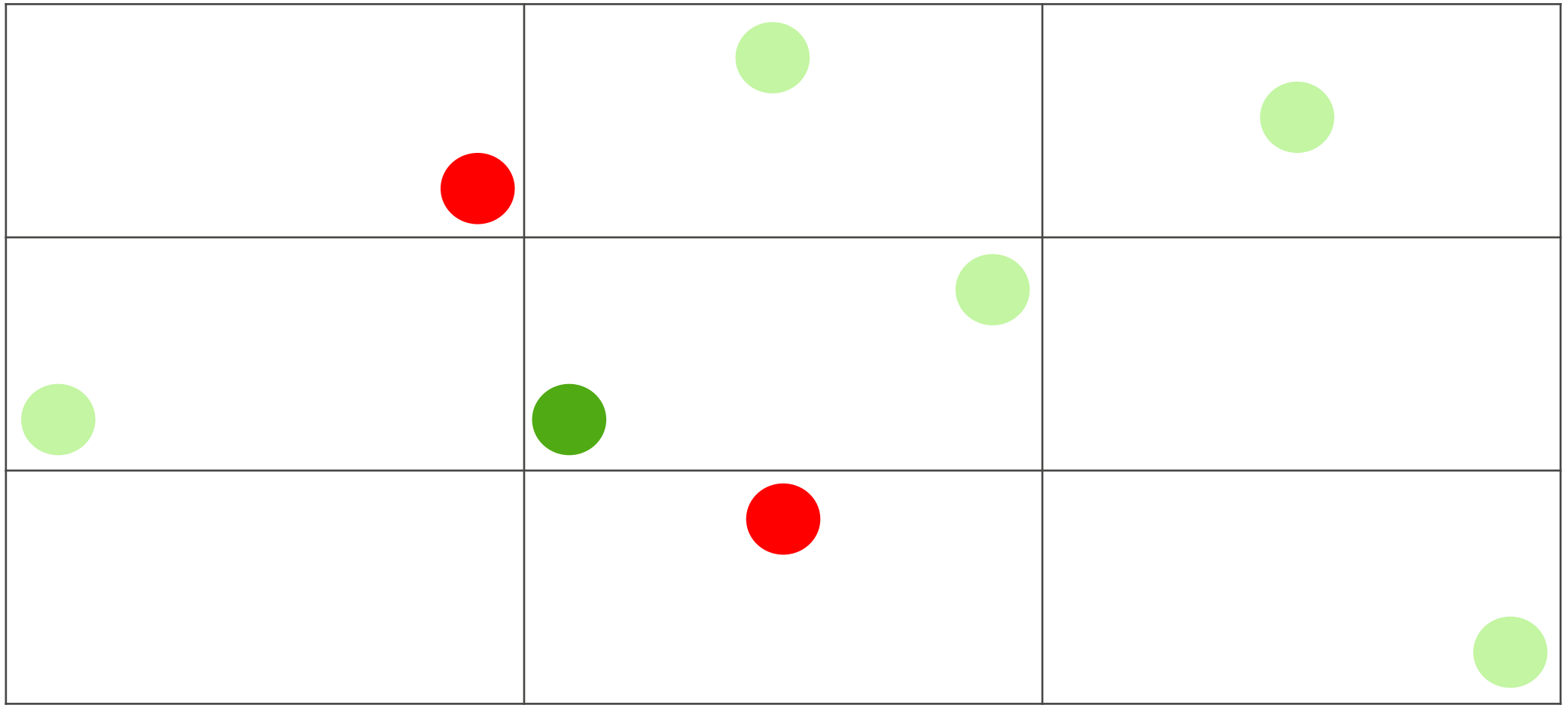
Key: geohash:aabbc:product:12

Value	Score
captain:234	1506349247000
captain:78	1506349143000

Key: geohash:aabbd:product:12

Value	Score
captain:89	1506349414000
captain:236	1506349247000
captain:77	1506349143000

GEO LOOKUPS – NEARBY CAPTAINS



SCALE

47
CITIES

6M
CUSTOMERS

250,000
CAPTAINS

PREVIOUS VALUES		
~~19~~	~~450,000~~	~~9,000~~

SCALE

4.1 M
ETA Requests

80 M
PINGS

26 M
LOOKUPS

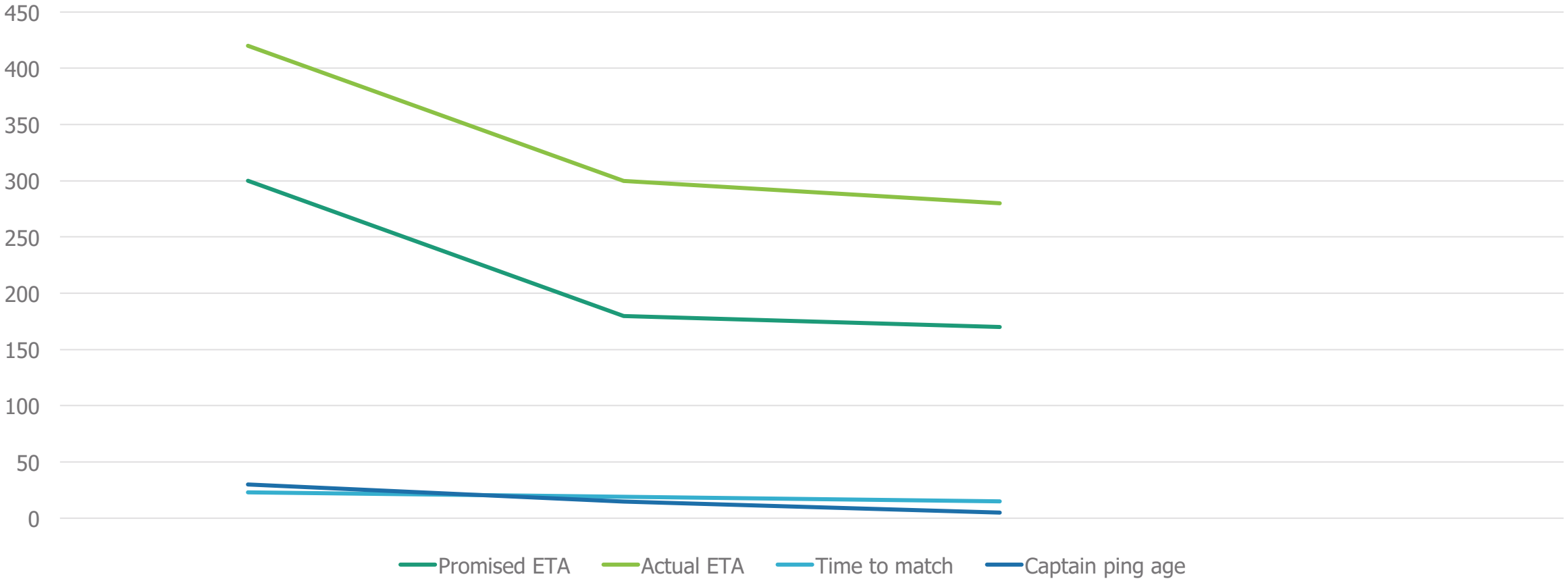
PREVIOUS VALUES

~~945,000~~

~~13 M~~

~~16 M~~

PERFORMANCE



PERFORMANCE

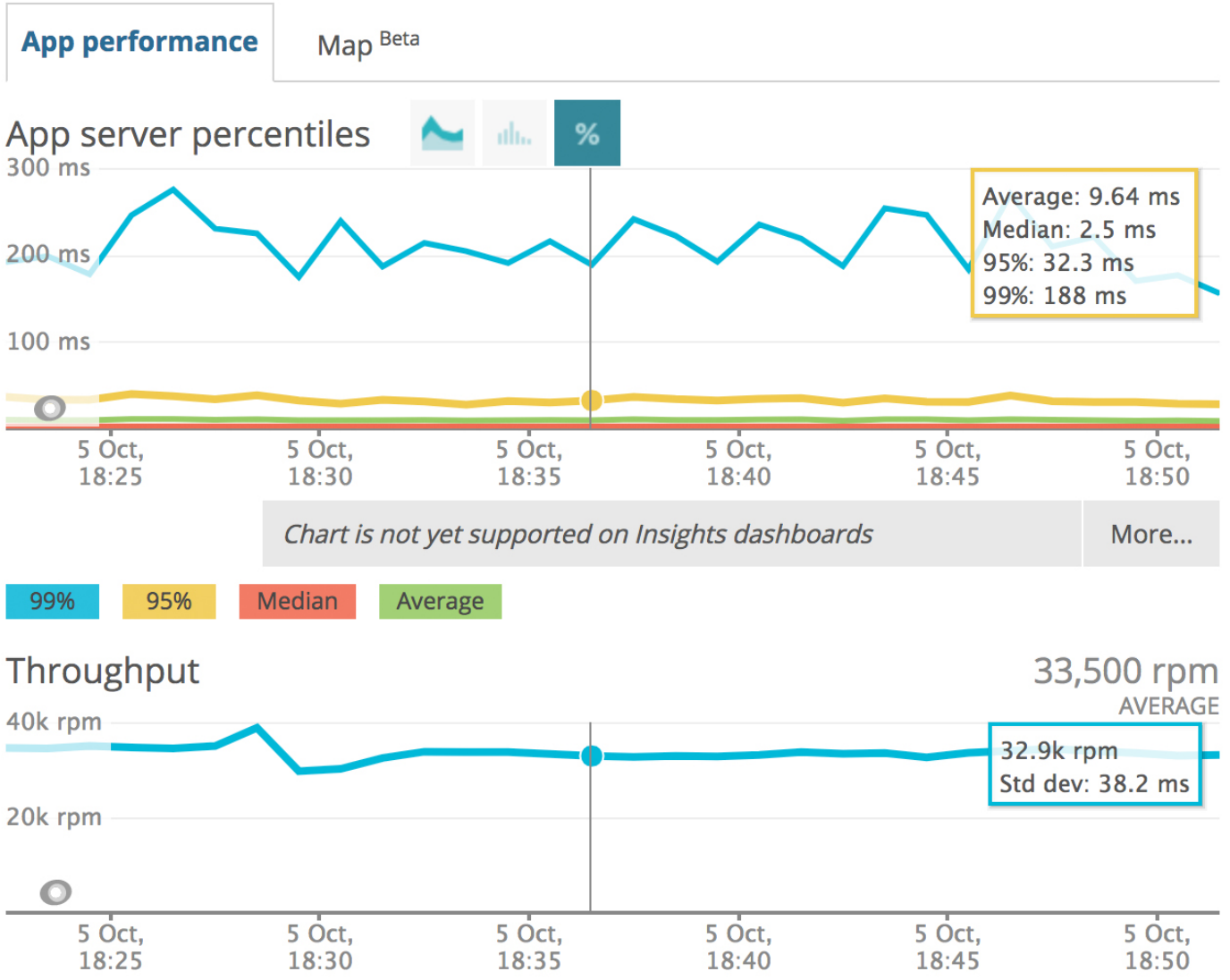
86%

RELIABILITY

99.99%

UPTIME

PERFORMANCE - LOOKUPS



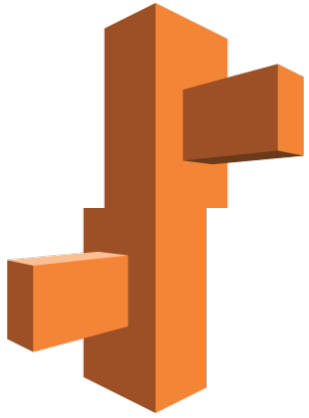
SUMMARY

- We were able to increase the frequency of pings to 4 per minute (every 15s)
- Improved our ETAs and customer experience as more pings meant that we were dispatching very close to captain’s locations
- Side benefit: granular level captain tracking for customers (in ride)
- Average time to match reduced from 2 minutes to 15 seconds

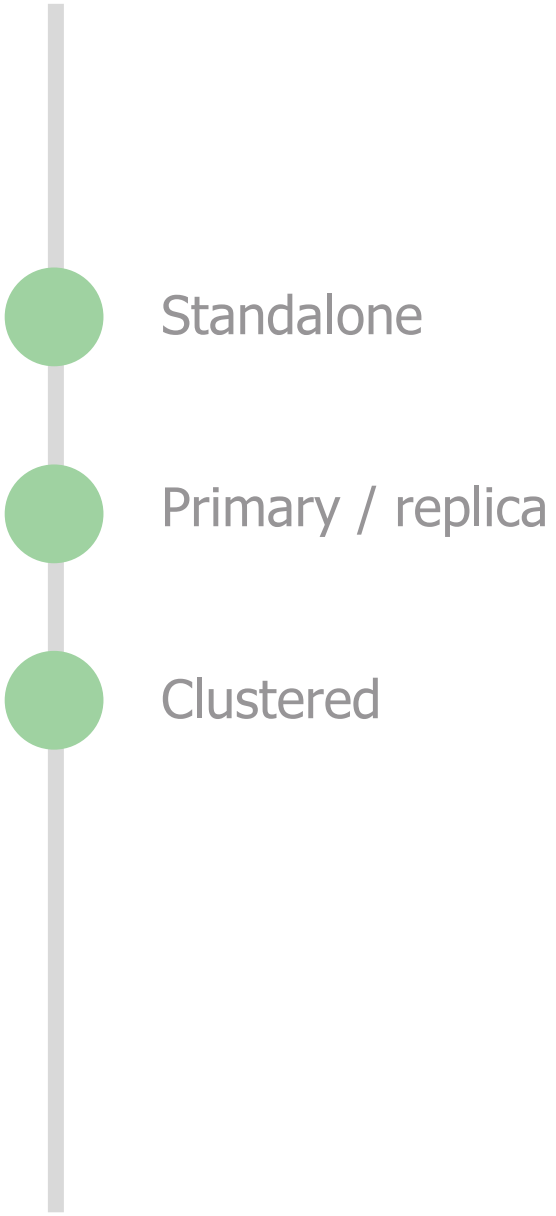
Careem

UNDER THE HOOD

OUR TECH STACK



REDIS – DIFFERENT MODES

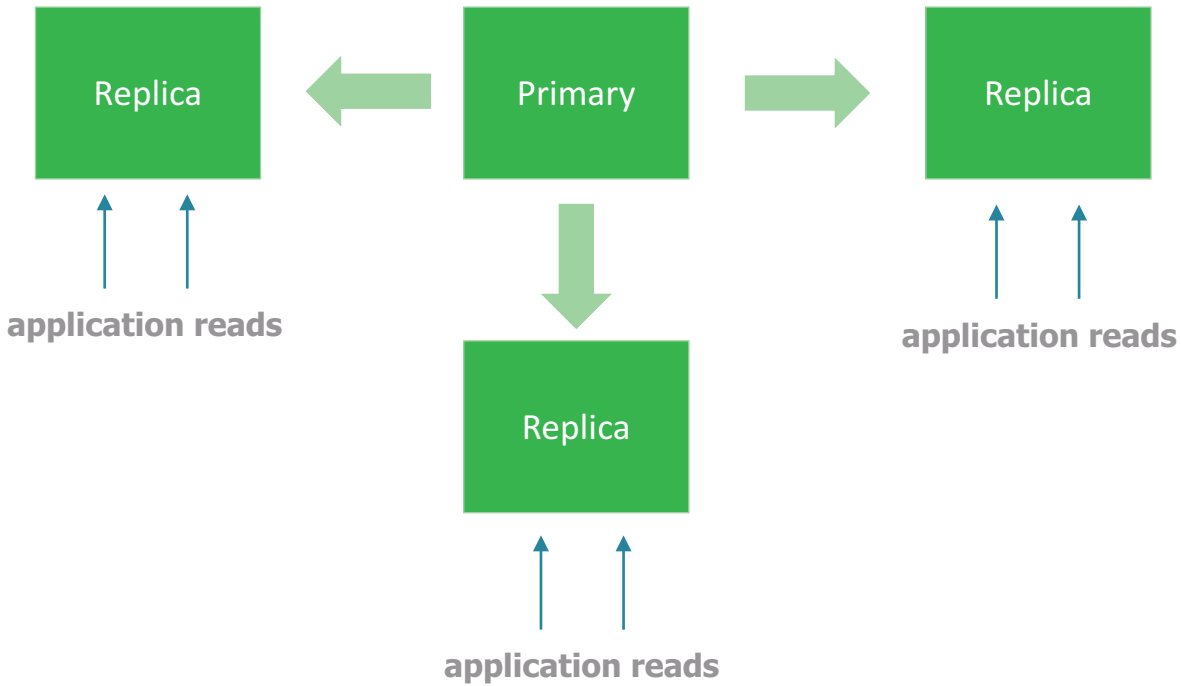


HYGIENE

- Always have multiple slaves in your cluster
- Configure back ups with a replica
- For writes never connect directly to the writable node directly, use the primary endpoint
- Set aside 30% memory

SCALING

- Read scale out
- Choose the right client library



Careem

FURTHER SCALING

FASTER PINGS BY GREATER NUMBER OF CAPTAINS

- We wanted to increase the frequency of pings further (every 5s)
- Even more captains on the network
- Even more customers reading assigned captain’s data
- Same performance demand
- More balanced read and writes

SCALE

80
CITIES

15 M
CUSTOMERS

450,000
CAPTAINS

PREVIOUS VALUES		
~~48~~	~~6 M~~	~~250,000~~

SCALE

15 M
ETA Requests

140 M
PINGS

48 M
LOOKUPS

PREVIOUS VALUES

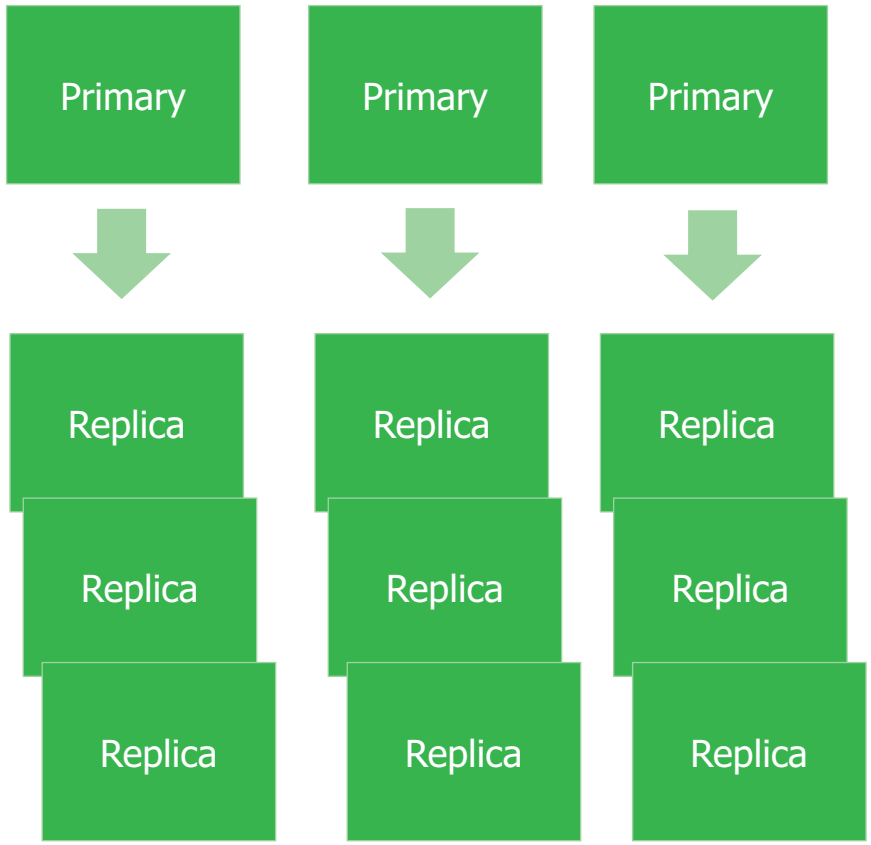
~~4.1 M~~

~~80 M~~

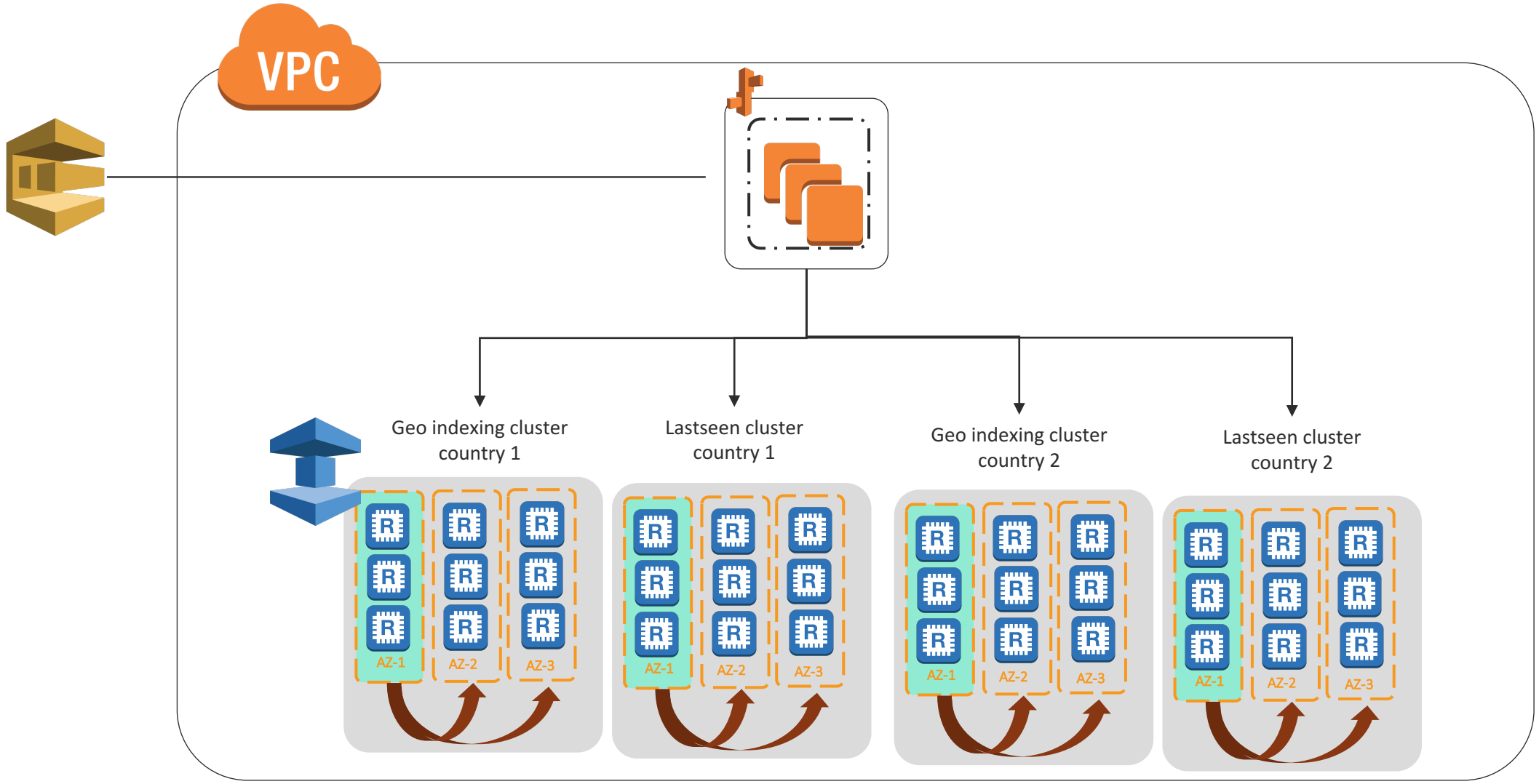
~~26 M~~

SHARDING

- Cluster mode with Redis 3.0
- Multiple primaries with each having its own set of replicas
- Hash slots
- CRC
- Application agnostic



TAKE 3


VPC
Geo indexing cluster country 1
Lastseen cluster country 1
Geo indexing cluster country 2
Lastseen cluster country 2
AZ-1
AZ-2
AZ-3

PERFORMANCE