Tips and Tricks for Building a Scalable Cloud Service

HackArizona 2015

Alex Landau

Amazon

Amazon

- Office in downtown Tempe near ASU (boooooooo)
- Most teams work for **Marketplace** (third party sellers)
- ~80 developers and growing fast
- **We're hiring.**

amazon.com®

The cunning plan…

- What? Why?
- Learn by doing
- Implications

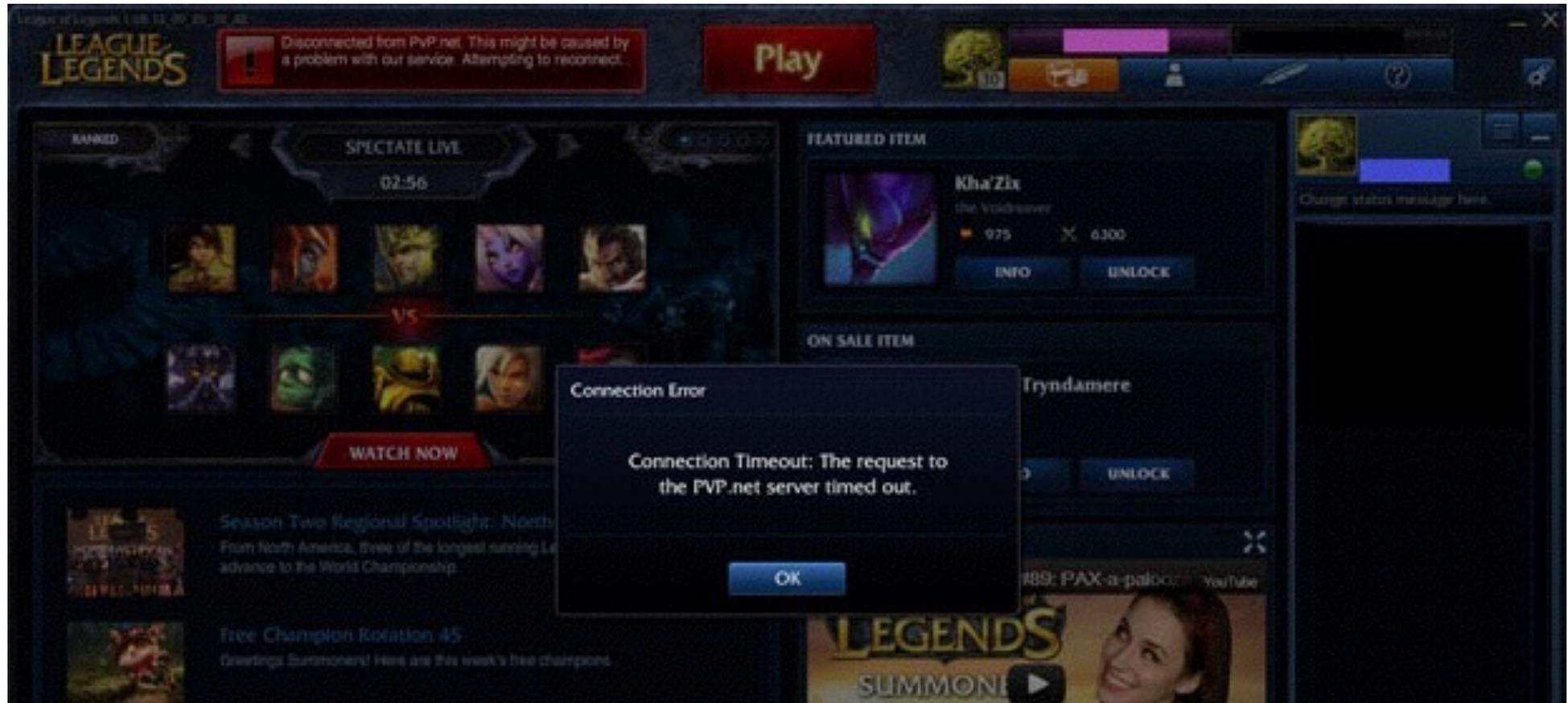# What does it mean to be Scalable?

- **Available**

reddit is under heavy load right now, sorry. Try again in a few minutes.

amazon.com

What does it mean to be Scalable?

- Available
- **Really, really fast**

What does it mean to be Scalable?

- Available
- Really, really fast
- **Reliable**

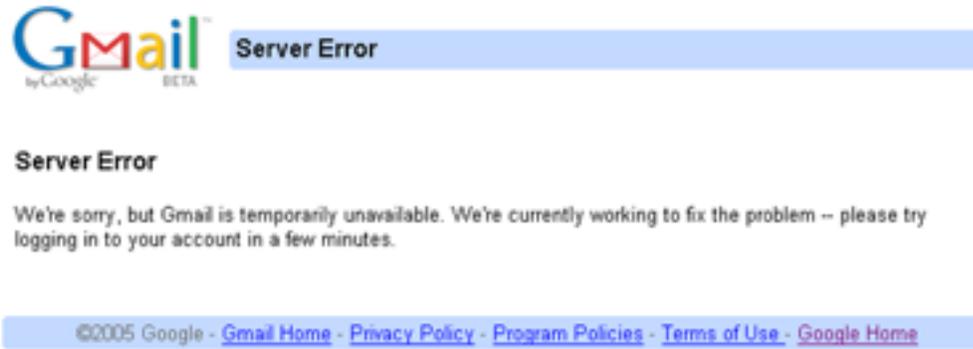
Error

Unable to start "ProtectedStorage" service on your computer due to the following error:

The operation completed successfully.

Standard Windows cryptography will be unavailable. Contact your system administrator.

OK

What does it mean to be Scalable?

- Available
- Really, really fast
- Reliable

- **With lots and lots of traffic.**

What is a lot of traffic?

- LA has ~374,000 cars per day on I-405 (http://tinyurl.com/adlsuol)

- Reddit has ~25 **million** upvotes and downvotes per day (http://www.reddit.com/about/)

- Dropbox hit 1 **billion** file uploads per day in 2013 (http://tinyurl.com/kskhn28)

- Amazon's S3 service peaks at **1.5 million requests** per second. (http://tinyurl.com/q6hnqvl)

amazon.com

Why do we care?

• **User experience should be positive. These things suck.**

So how do we build a service that scales?

The basics – our little service

User

Location

Checkin

ID, name, ….

ID, name, GPS coords, …

User was at **location** at **timestamp.**

amazon.com

The basics – our little service

User		
ID	Name	...

Location			
ID	Name	GPS Coords	...

Checkin			
ID	User ID	Location ID	Timestamp

amazon.com

The basics – our little service

- Record a checkin
- See a user's current location
- See the checkins for a user across a time range
- See the checkins at a location for a user across a time range

Attempt 1 – Direct database access
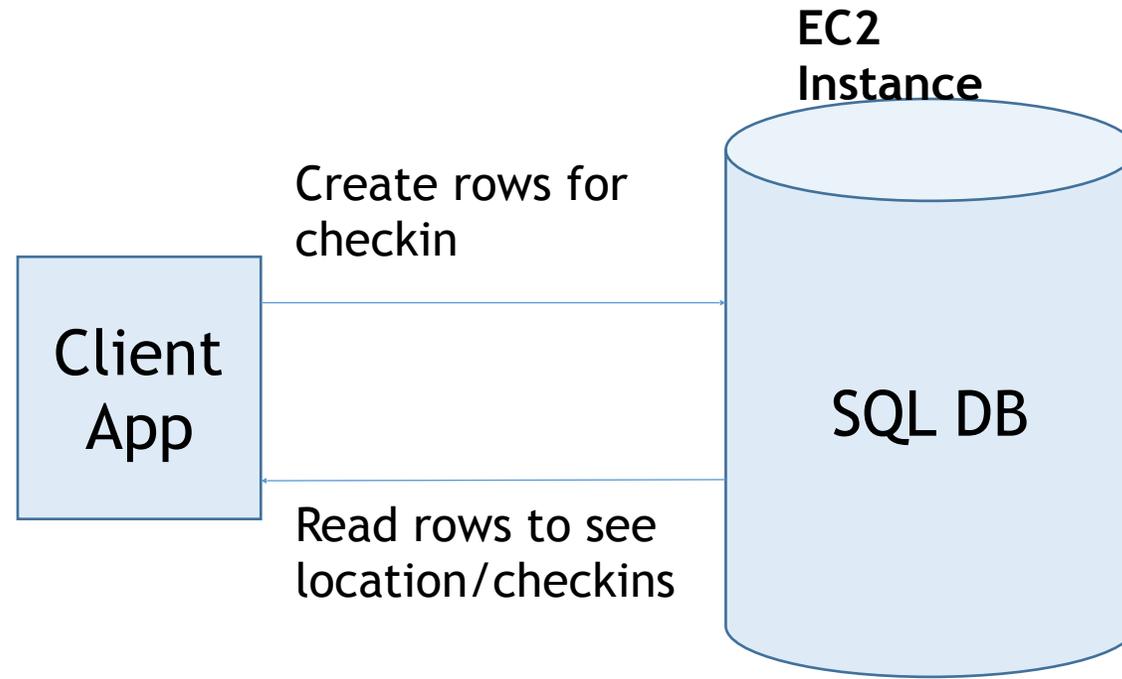
Problems

Attempt 2 – A service middleware

EC2 Instance

EC2 Instance

RecordCheckin

Client App

Web service

SQL DB

GetUserLocation,
GetCheckinsForUser,
GetCheckinsForLocation
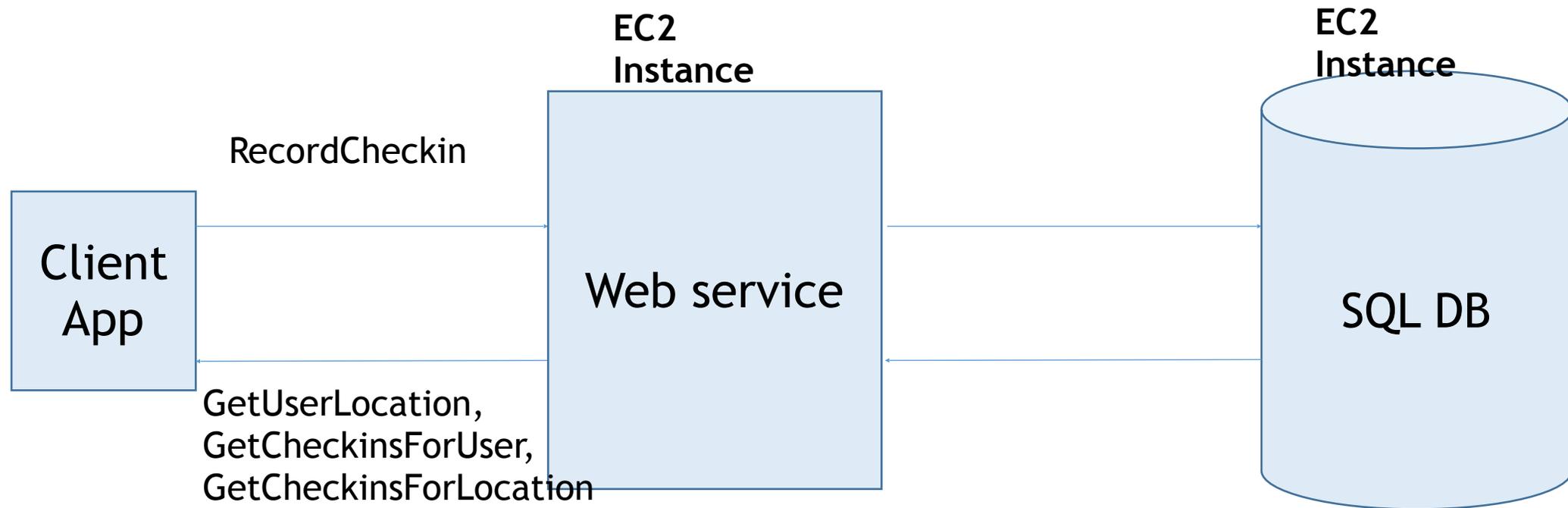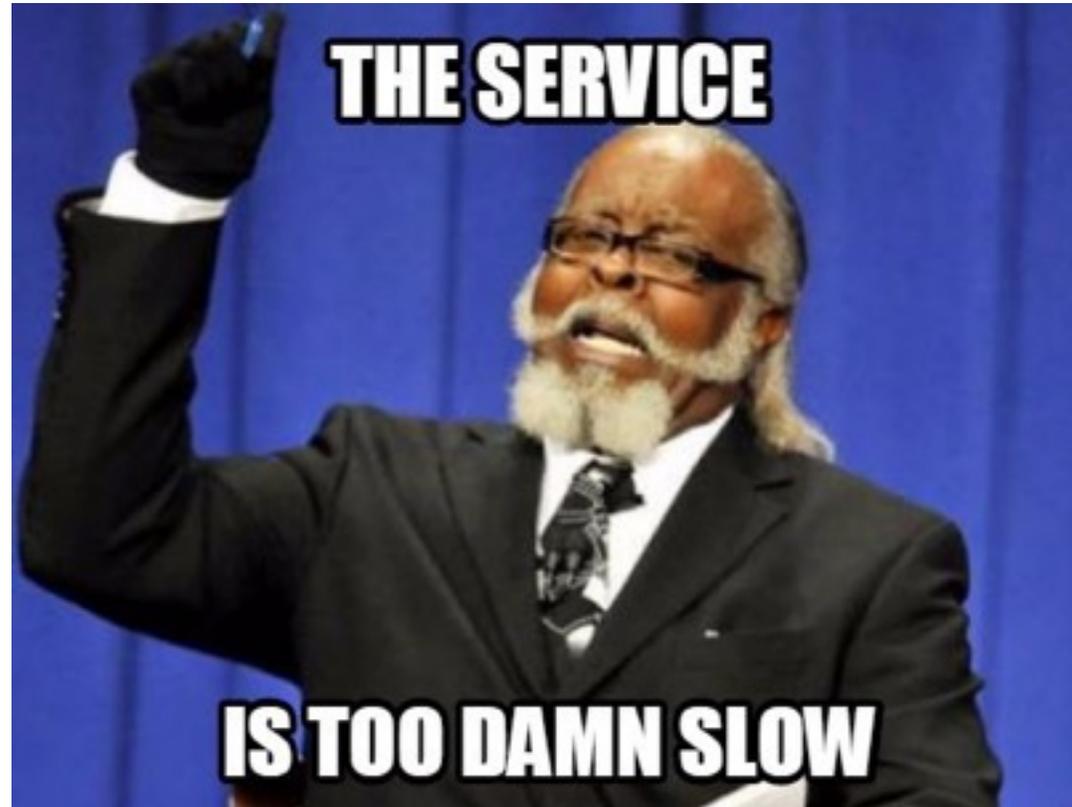
Problems

Asynchronous IO

Attempt 3 – Asynchronous IO
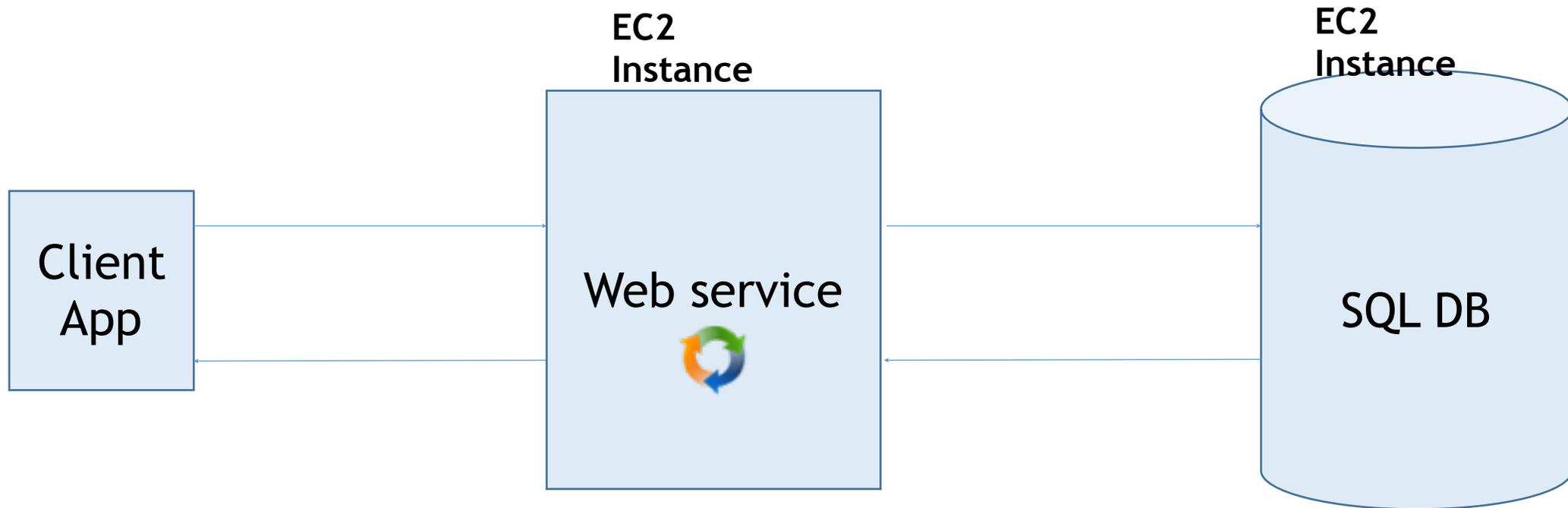
Problems
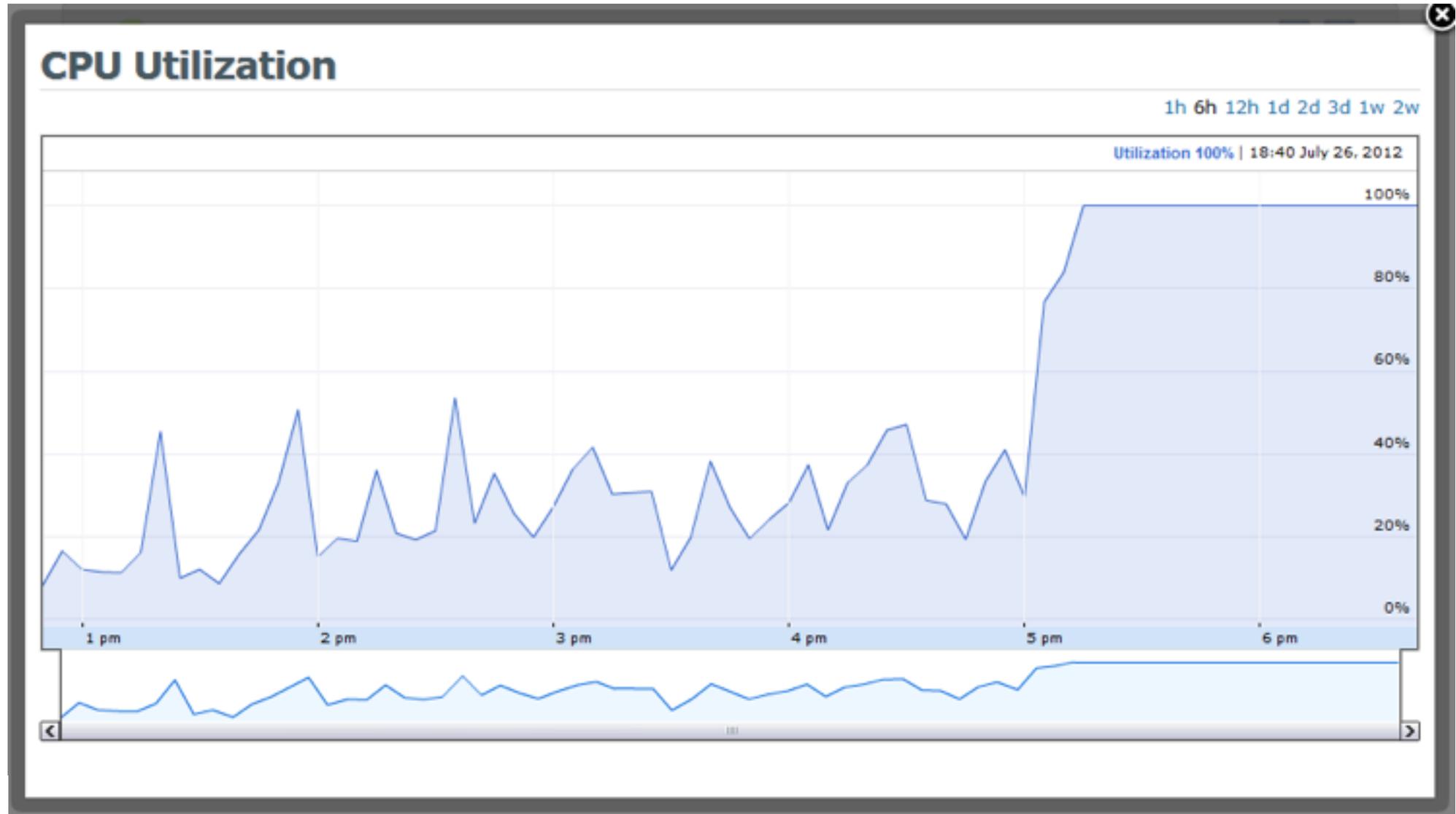
Attempt 4 – Load balancing

Client App

Elastic Load Balancer

Reverse Proxy/Load Balancer

EC2 Instances

Web service

Web service

Web service

Web service

EC2 Instance

SQL DB

Problems

Problems

EC2 Instances

Web service

Web service

Web service

Web service

Elastic Load Balancer

Client App

Reverse Proxy/Load Balancer

100 ms write

EC2 Instance

SQL DB
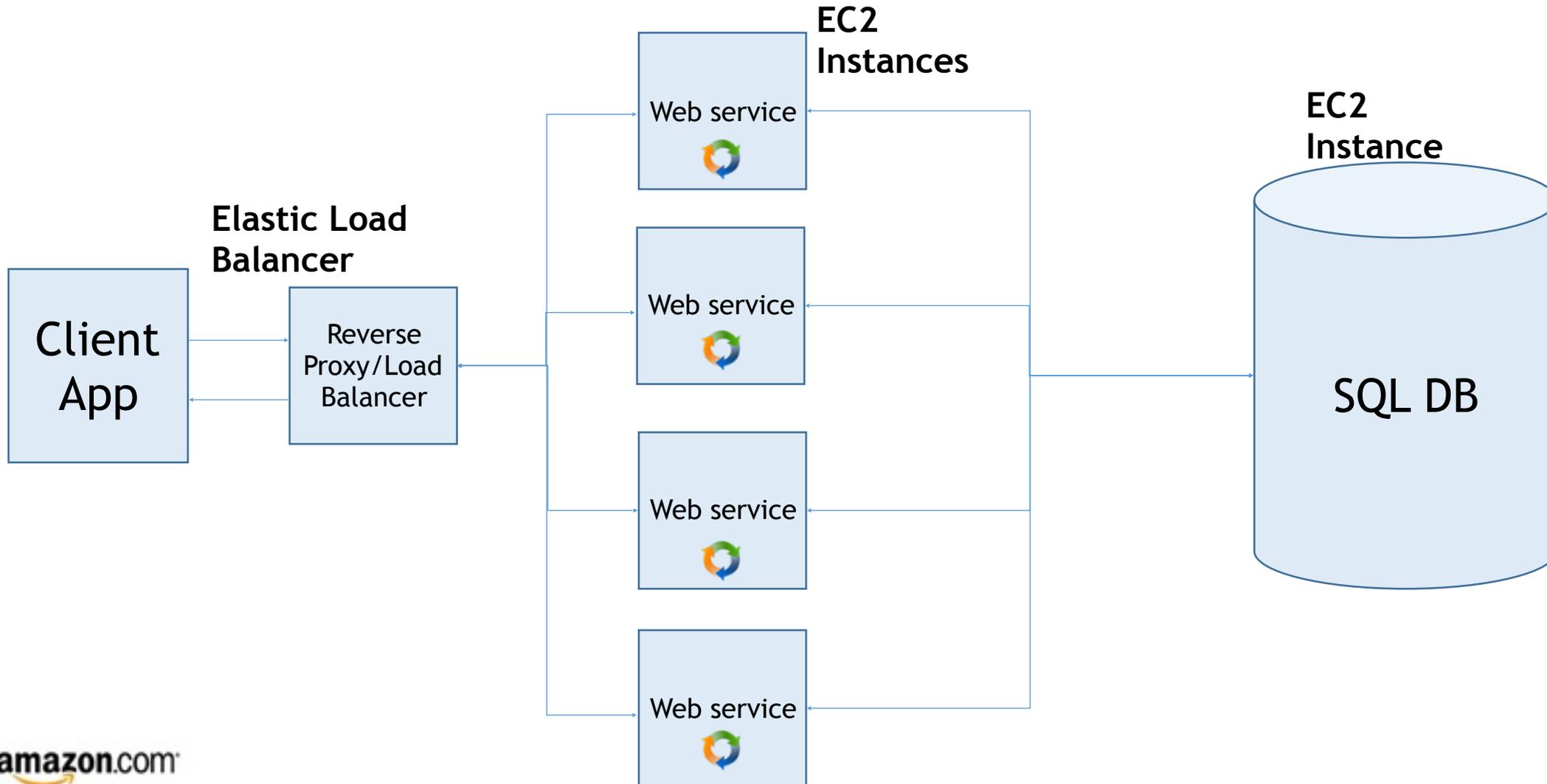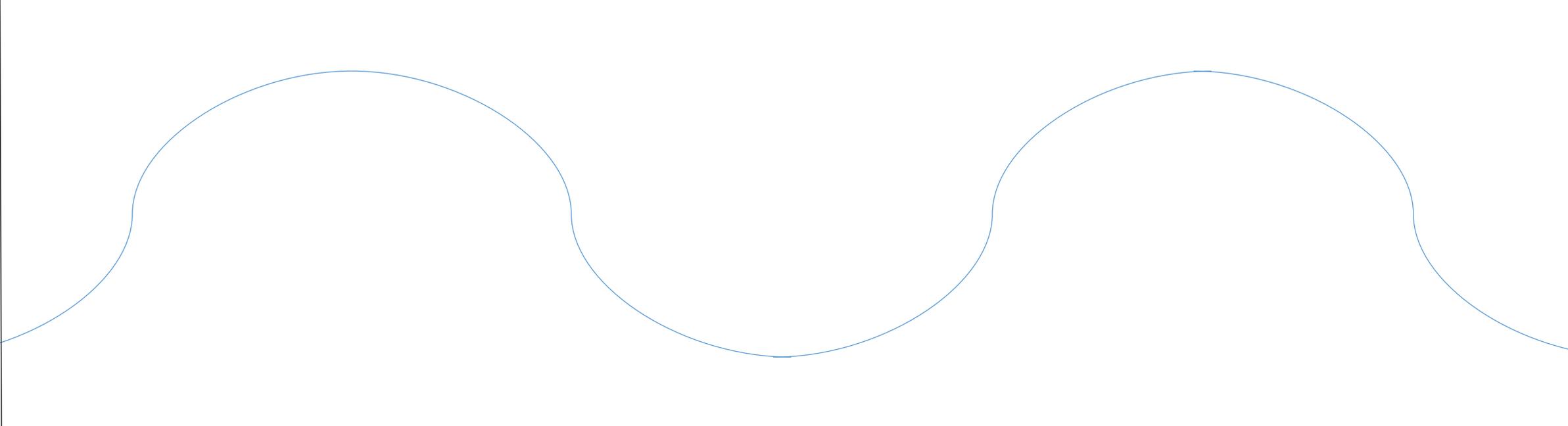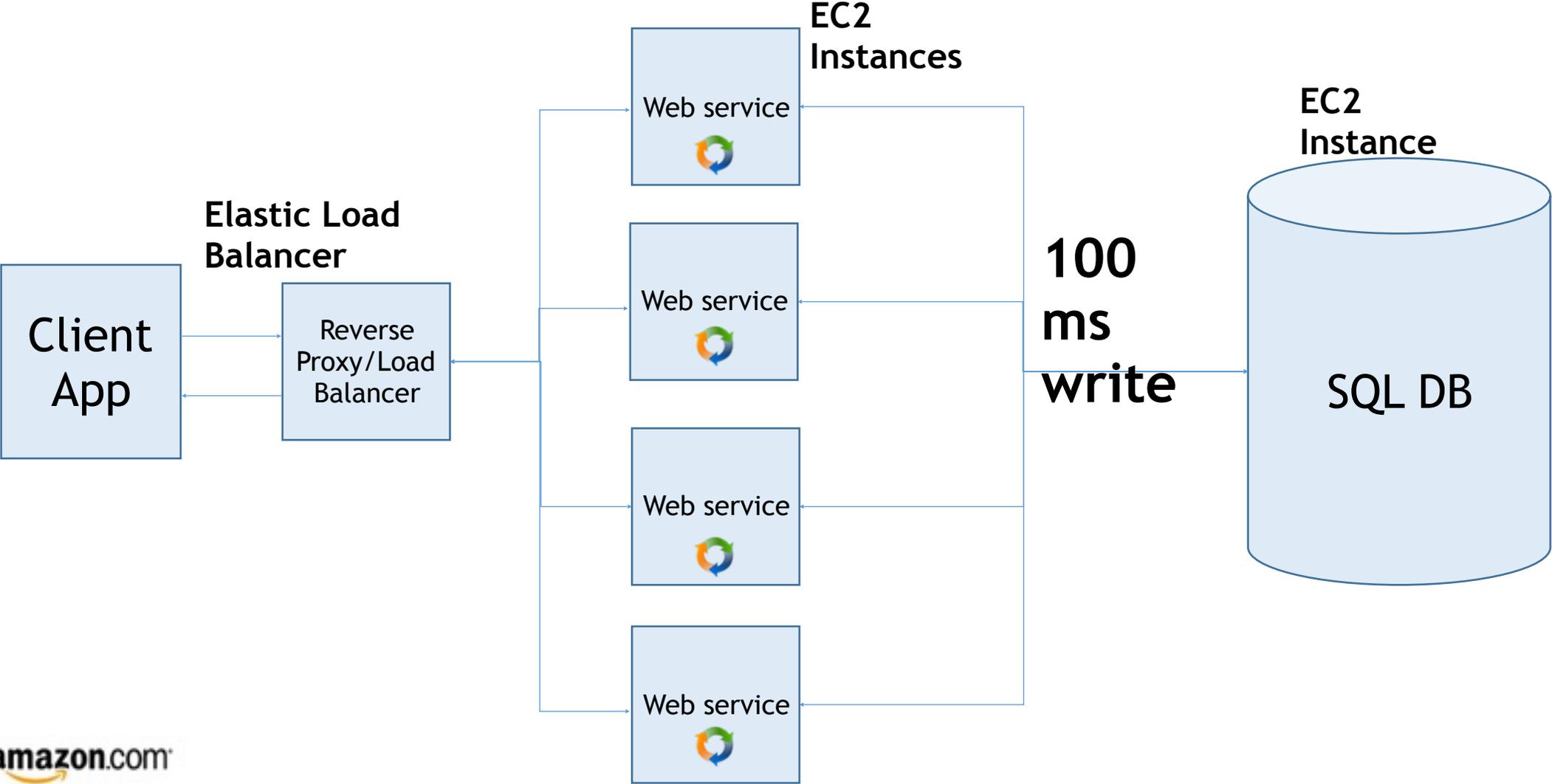
Attempt 5 – A queue in the cloud

EC2 Instances

Elastic Load Balancer

Web service

Web service

Web service

Web service

Client App

Reverse Proxy/Load Balancer

10 ms write

SNS/SQS

100 ms write

EC2 Instance

SQL DB

A queue in the cloud

Problems

THE DATABASE IS TOO DAMN SLOW
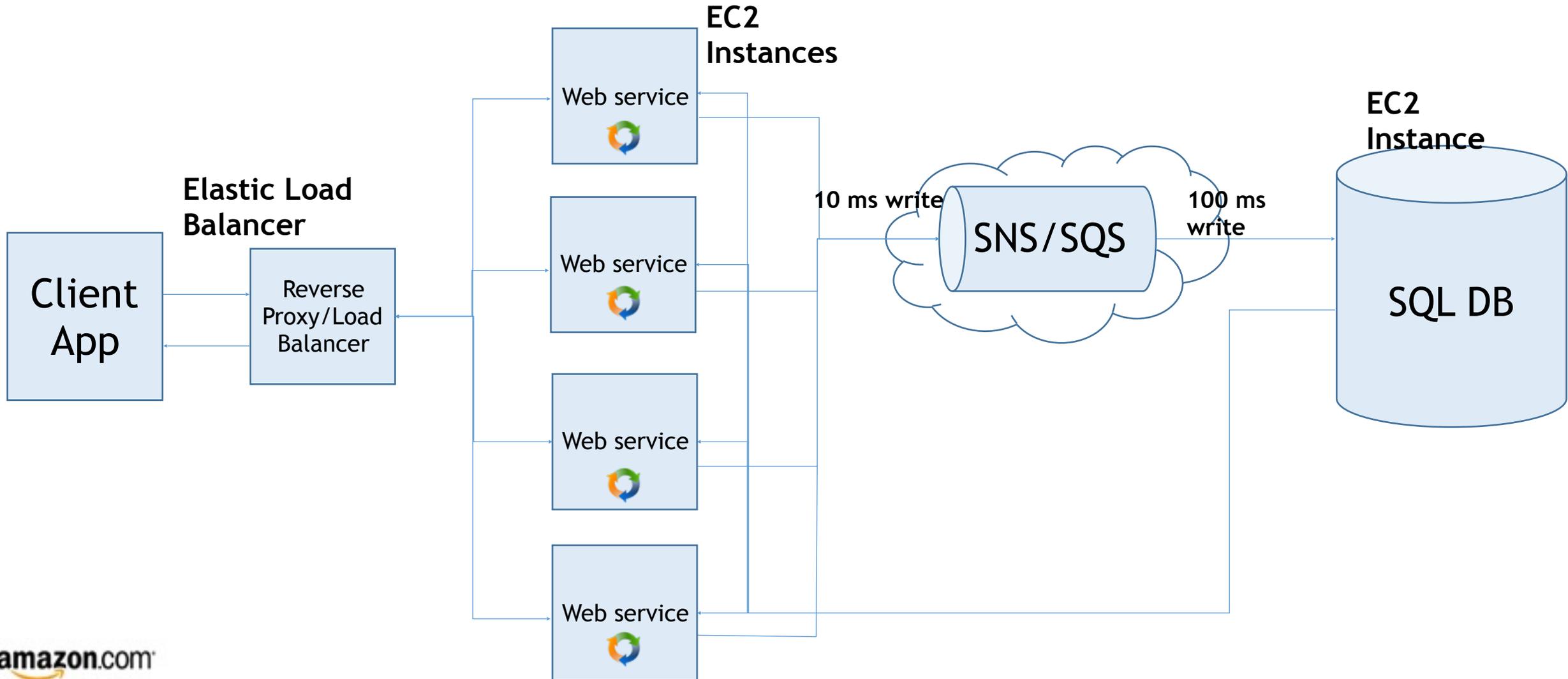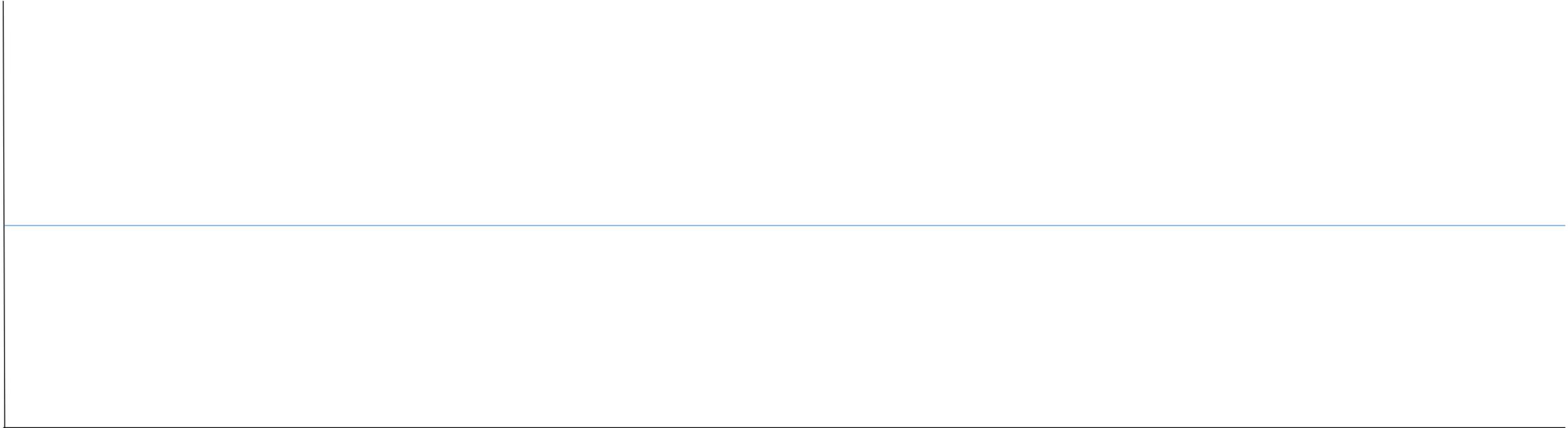
Amazon DynamoDB

- NoSQL
- **Fast, available, reliable**
- Automatically scalable
- ↑ Throughput == ↑ $$$

Attempt 6 – A database in the cloud

Client App → Elastic Load Balancer (Reverse Proxy/Load Balancer) → EC2 Instances (Web service) → SNS/SQS → DynamoDB

amazon.com
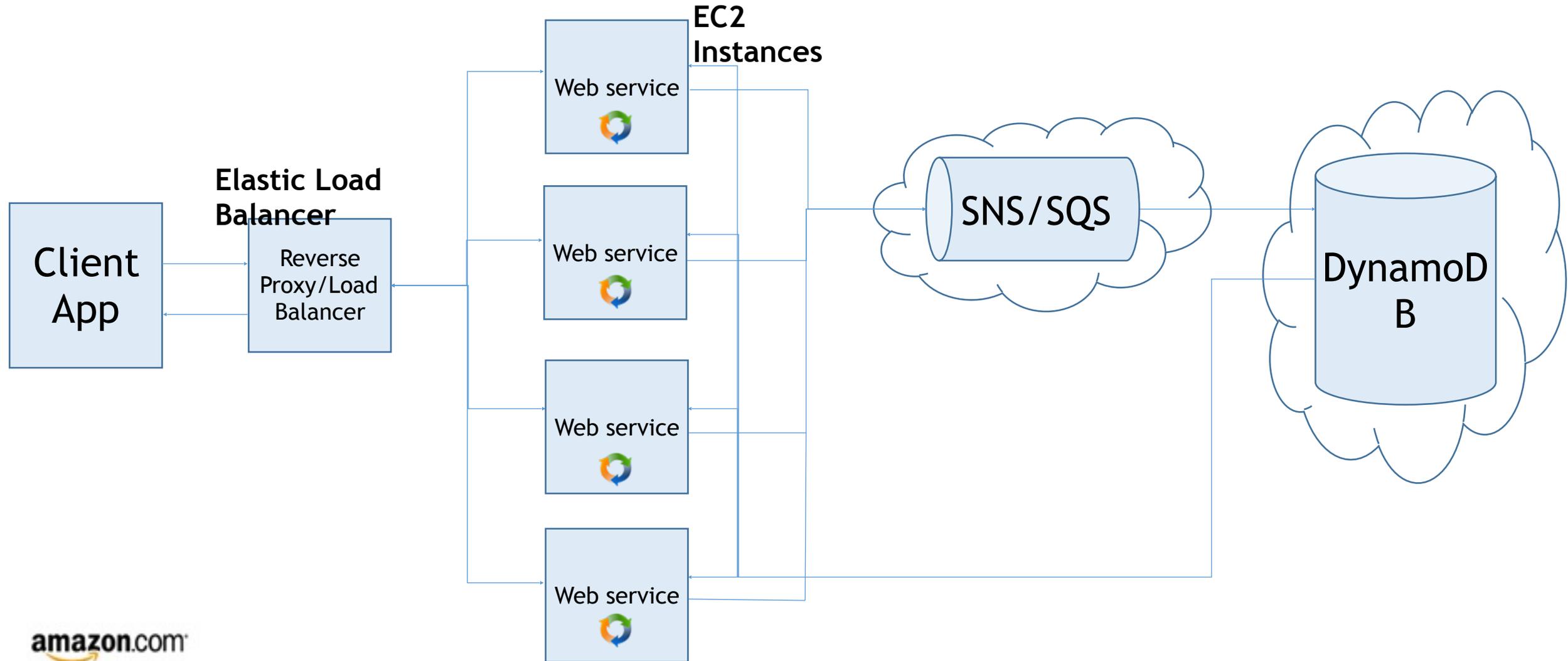
Even more improvements

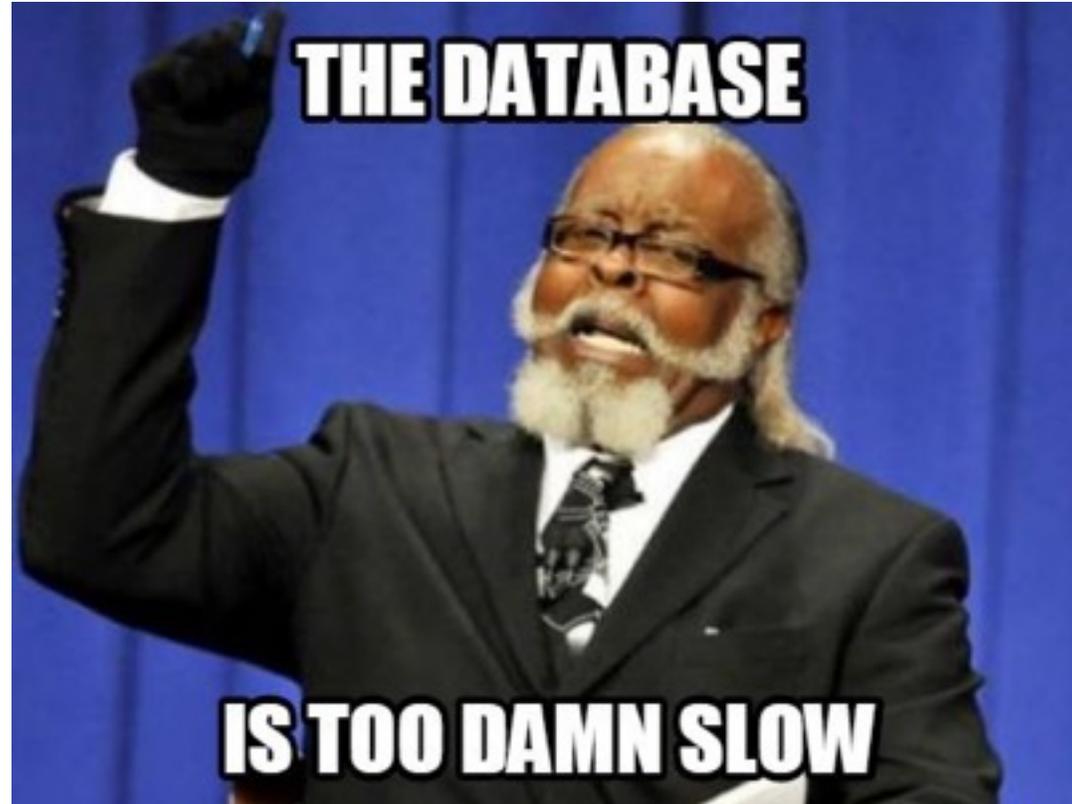- **CloudSearch** to speed up checkin reads
- Time-series database (**Druid**, **InfluxDB, Prometheus**) for checkins
- Friends lists, push notifications, metrics and data
- **Caching (ElastiCache)**

Summary

We went from this...

Summary

...to this.

EC2 Instances

Elastic Load Balancer

Client App → Reverse Proxy/Load Balancer → Web service / Web service / Web service / Web service → SNS/SQS → DynamoDB
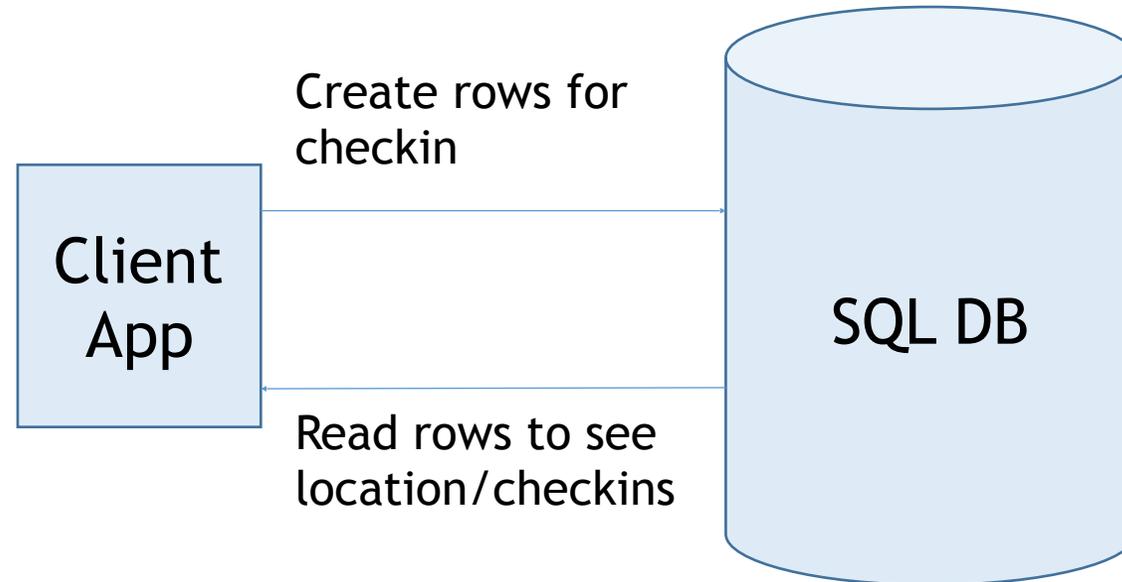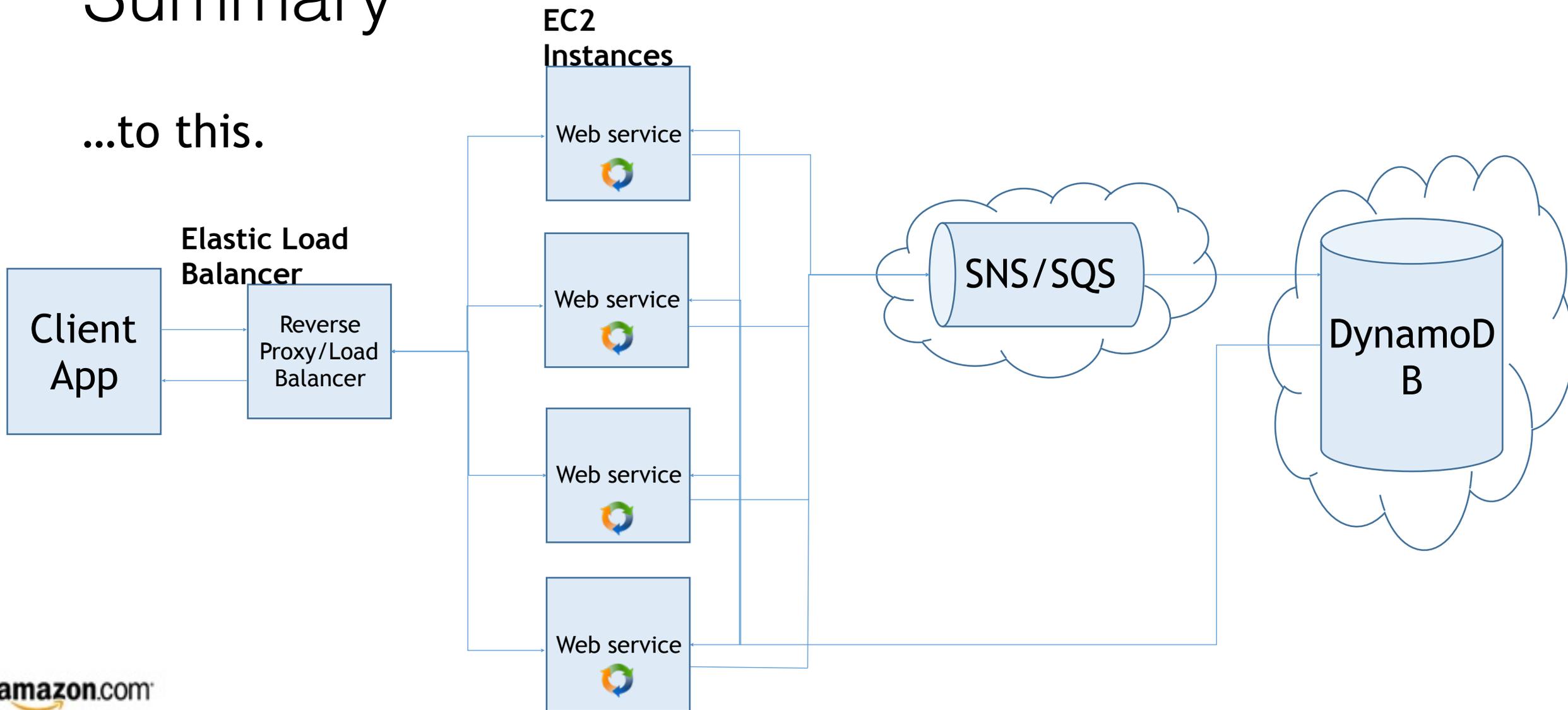
Summary

- Big wins.
  - Linearly scalable.
  - More money → handle more traffic.
  - Availability, performance, reliability.
  - Managed services ≈≈ **significant** savings. (http://tinyurl.com/q29n6h3)

Implications

- Real world services are significantly more complex, **but they use the same techniques described here.**
  - Asynchronous IO
  - Messages queues
  - Managed cloud databases (both SQL and NoSQL) with service guarantees
  - Load balancing
  - Caching
- You can make your entire infrastructure completely **horizontally scalable.**

Questions?