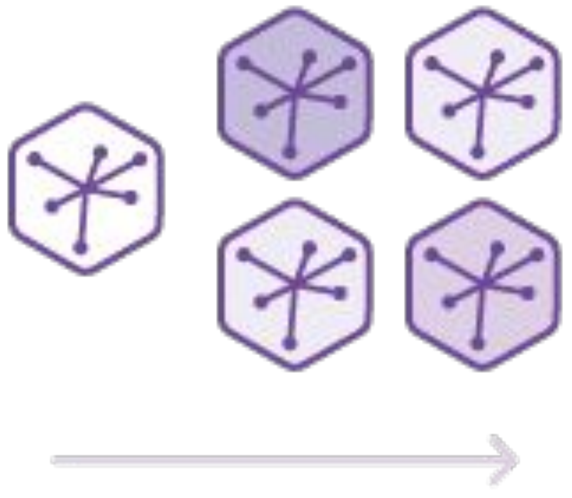
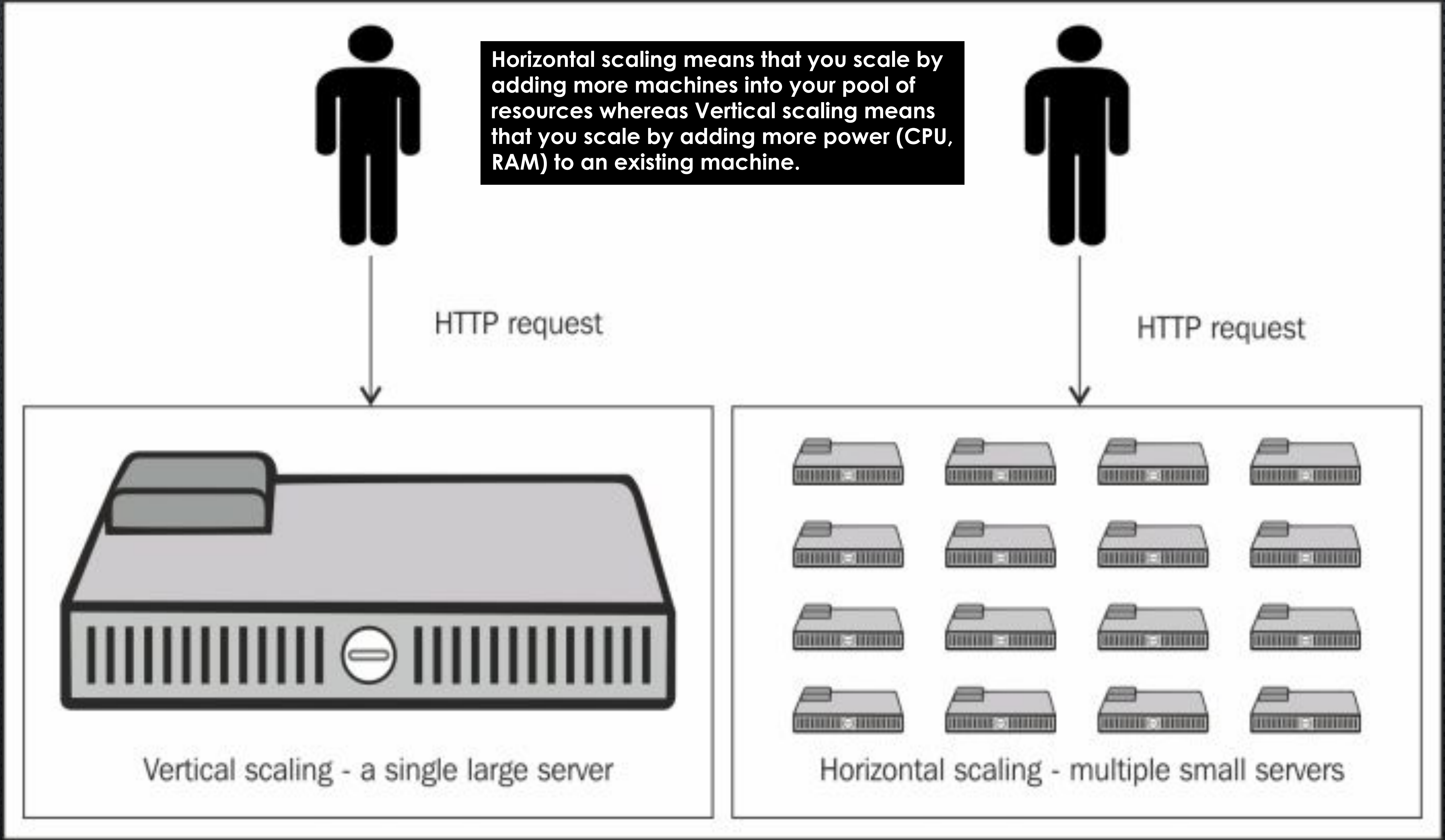


SCALING WITH MULTIPLE DYNOS

Shu Lin Chan | Priyanka Grover | James Wang

OUTLINE

- Horizontal scaling - multiple (free) dynos
- Load balancing
- Handling separate databases


HTTP request
HTTP request
Vertical scaling - a single large server
Horizontal scaling - multiple small servers

Horizontal scaling means that you scale by adding more machines into your pool of resources whereas Vertical scaling means that you scale by adding more power (CPU, RAM) to an existing machine.

HORIZONTAL SCALING – INTRODUCTION

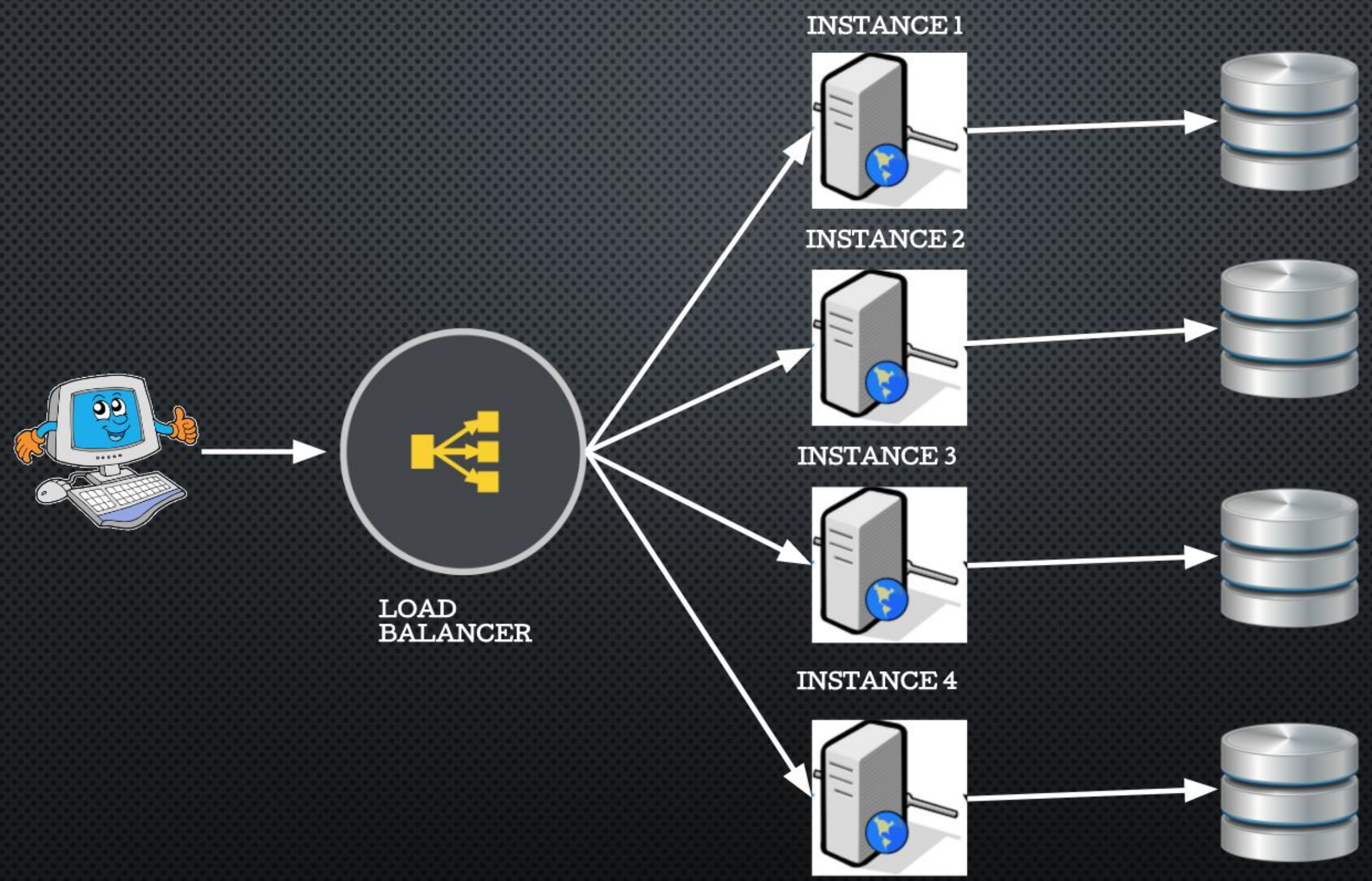
Horizontal scaling means that you scale by adding more machines into your pool of resources.

Heroku allows for auto-scaling dyno count for paid applications.

PROBLEMS

Two problems that do not exist in single-dyno applications:

- How to distribute the HTTP requests across the multiple dynos.
- How to manage the contents of each applications databases.



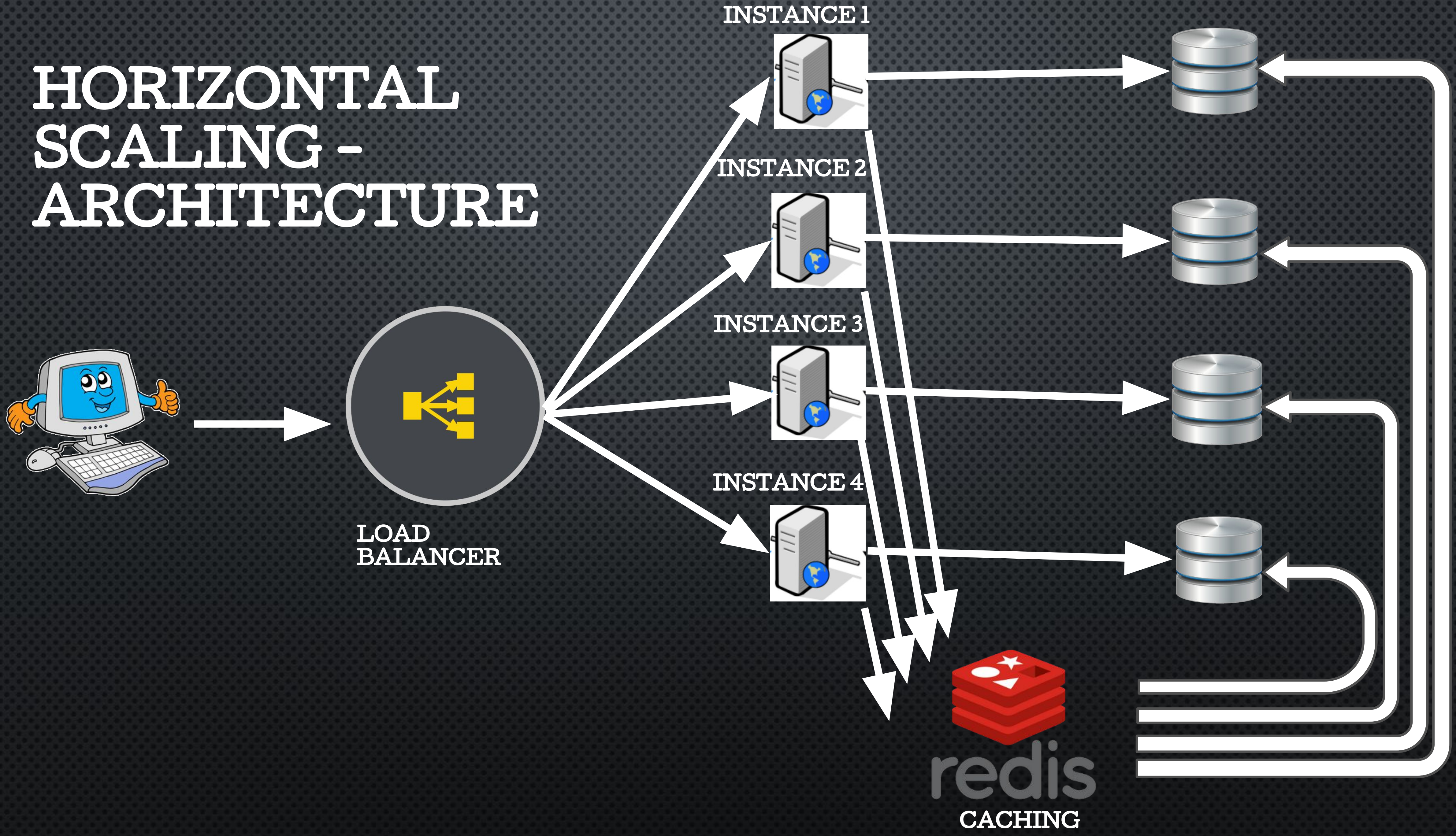
HORIZONTAL SCALING - IMPLEMENTATION

MAIN IDEA: IMITATE DYNO SCALING USING MULTIPLE FREE APPLICATIONS INSTEAD OF MULTIPLE DYNOS IN A SINGLE APPLICATION.

OUR SOLUTIONS:

- PRIMITIVE LOAD BALANCER TO DISTRIBUTE REQUESTS.
- PERIODIC DATABASE UPDATES, CACHING, AND LOCAL READ/WRITES.

HORIZONTAL SCALING - ARCHITECTURE



LOAD BALANCING - INTRODUCTION

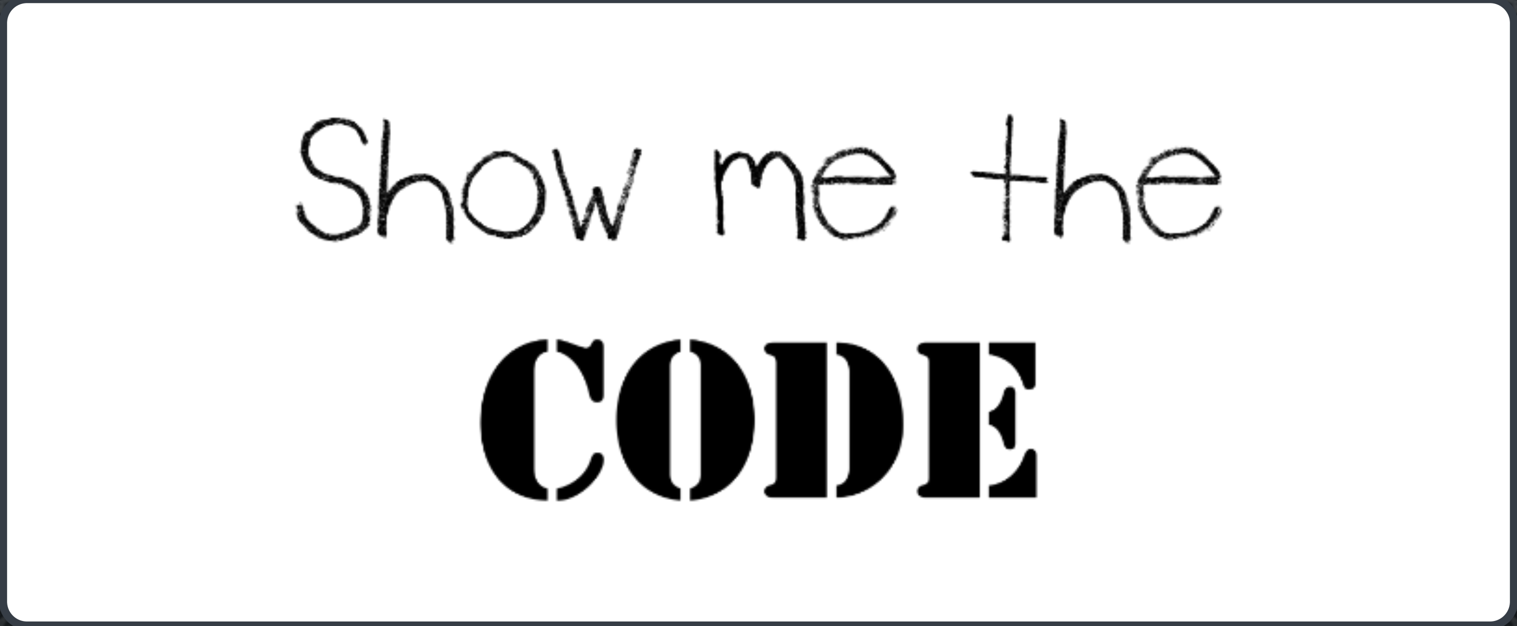
- ❖ Load Balancing is how to distribute load across multiple instances.
- ❖ Essential to make sure that all dynos are being utilized effectively.
- ❖ Needs some form of scheduler to handle load
  - ➢ Round Robin: Even distribution.
  - ➢ Least Connection: A master figures out which server has the lowest current load and redirects new requests to that server.
  - ➢ Agent Based Adaptive: Each server keeps track of it’s load and tells the master, the master then picks the lowest and sends requests to that server.
  - ➢ Chained Failover: Chain of servers, requests are all sent to the first in the chain and do not go to the next until the first is overloaded.

LOAD BALANCING - IMPLEMENTATION

Chose Round Robin because it was the easiest and the most relevant load balancer.

Round Robin implementation:

- Increments a value in the cache for every visit
- Takes the mod of the value and sends it to the corresponding instance
- Only triggers on a visit to the master and not on visits to the application instances



LOAD TEST WITH 1 INSTANCE VS 4 INSTANCES

SEPARATED DATABASES - INTRODUCTION

Multiple instances of our application causes different database instances. Hence, we have to combine the separate database instances.

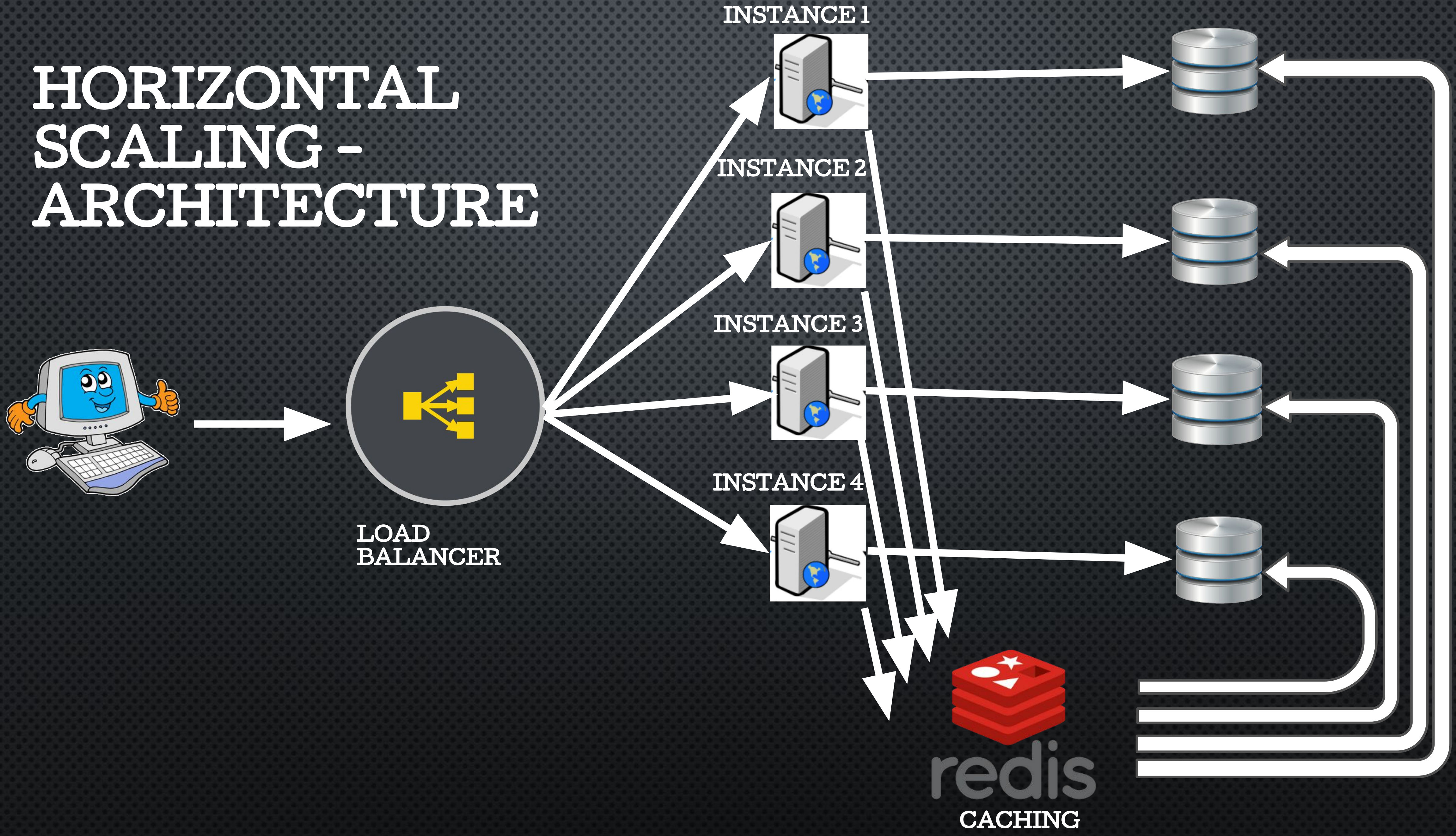
Many different solutions to solve this problem:

- **Use a single master database that all instances access**
- **Use a read/write database and many read-only databases that are periodically updated.**
- **Use a cache to store new updates and write to all databases periodically.**
- **Use unsynced local databases and then sync them periodically to a master with all updates.**

SEPARATED DATABASES - IMPLEMENTATION

- ❖ Chose to use unsynced local databases due to the nature of our load balancer.
- ❖ Unsynced local database implementation:
    - ➢ Each instance writes and reads from it’s own database.
    - ➢ Sends data asynchronously to the master database that keeps track of updates.
    - ➢ Master database propagates updates to instances every 15 minutes.
    - ➢ Since a user will remain on the same instance all their activities will be instantly updated.

HORIZONTAL SCALING - ARCHITECTURE



ADVANTAGES

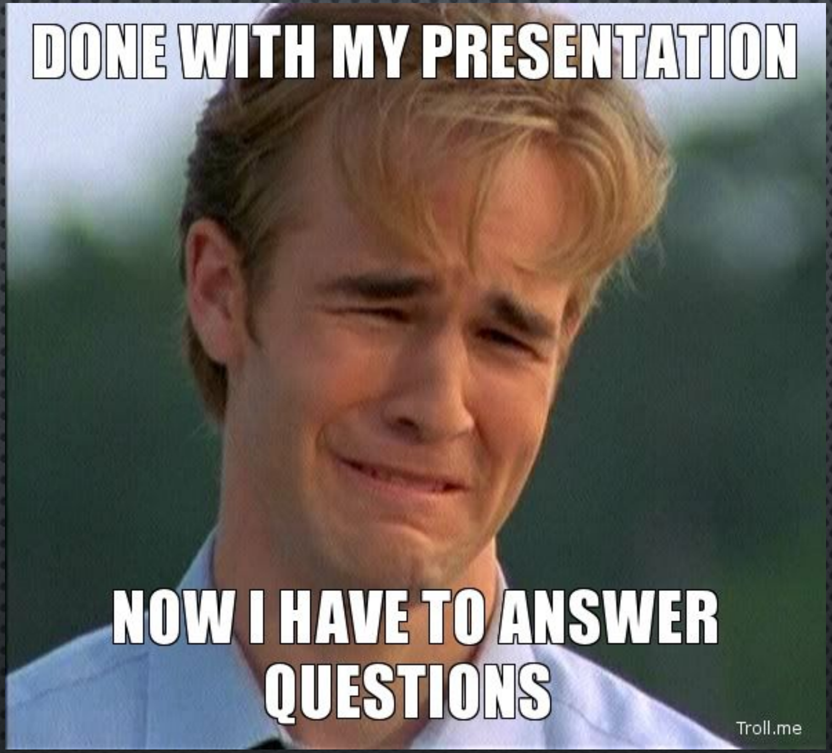
Better scaling than on one dyno

- Cuts the load of each dyno linearly (Ex. with 10 instances each instance is taking 1/10 of the clients of a single dyno).

DISADVANTAGES

- Adds an overhead cost of an additional redirect and the load balancer
- Soft consistency between the databases of different instances due to delayed updating.

Thank you!



Of course you have QUESTIONS!!