# Cloud Computing Project

## Lo-okup

Hemanth Aditya
Omid Askarisichani

# Motivation

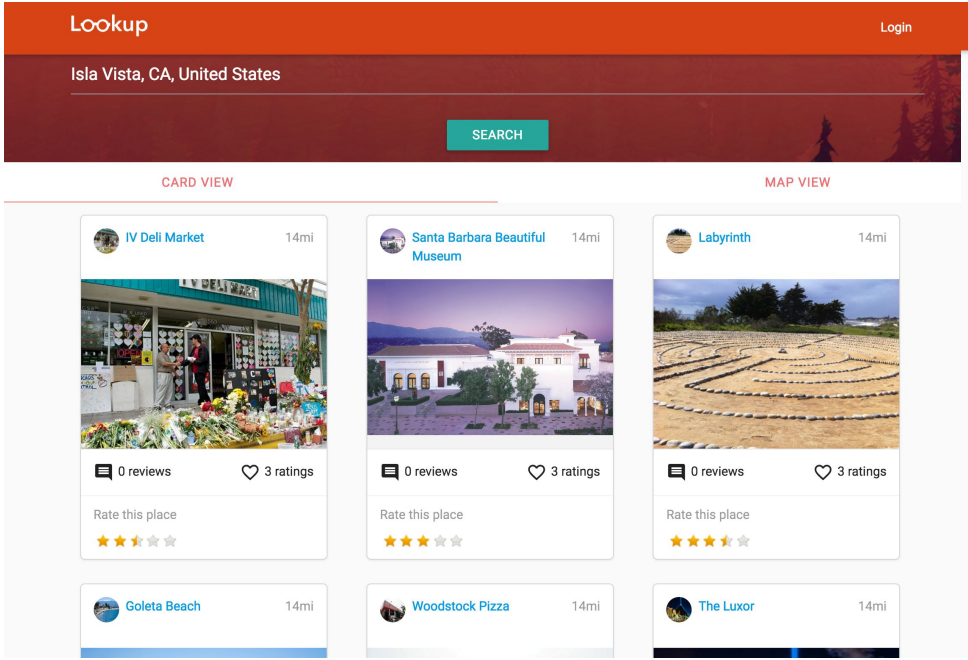
- An open source web application to add any picture to any place on Google Maps
- A social networks of users with the ability of like, rate, endorse and comment for each picture
- Be able to run on any device, and for the most part, compatible with all phone Operating Systems
- Highly available, consistent and fast

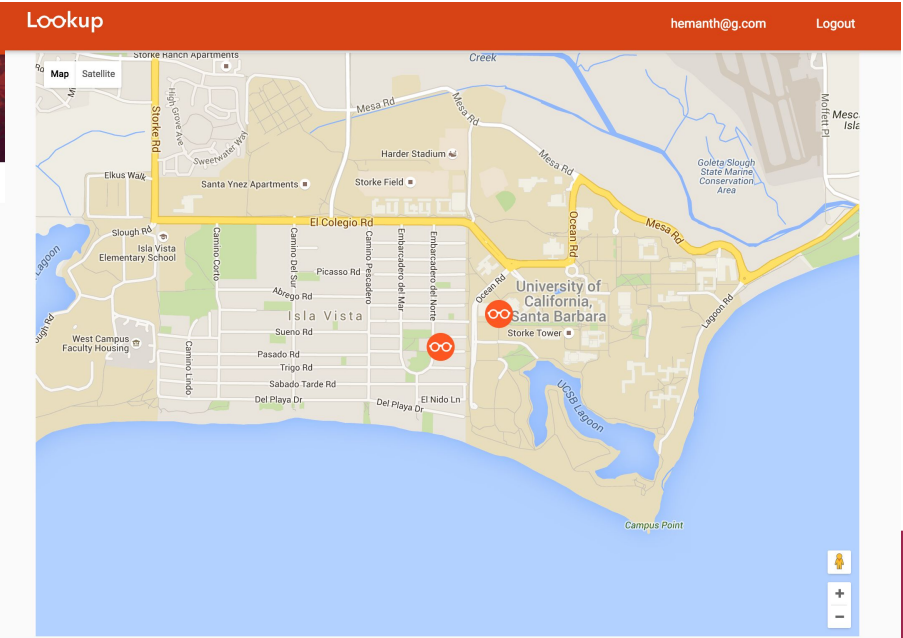
# Application


Lookup allows users to pin places around them on a map for the rest of the community to explore.

Users can browse through places around them within a 50 miles radius. They can rate, review and share places.

# Lookup


Lookup
Login
Isla Vista, CA, United States
SEARCH
CARD VIEW
MAP VIEW
IV Deli Market
14mi
0 reviews
3 ratings
Rate this place
Santa Barbara Beautiful Museum
14mi
0 reviews
3 ratings
Rate this place
Labyrinth
14mi
0 reviews
3 ratings
Rate this place
Goleta Beach
14mi
Woodstock Pizza
14mi
The Luxor
14mi


Lookup
hemanth@g.com
Logout
Map
Satellite
Storke Rd
Mesa Rd
El Colegio Rd
Harder Stadium
Santa Ynez Apartments
Storke Field
Elkus Walk
Slough Rd
Isla Vista Elementary School
Picasso Rd
Abrego Rd
Isla Vista
Sueno Rd
Pasado Rd
Trigo Rd
Sabado Tarde Rd
Del Playa Dr
El Nido Ln
West Campus Faculty Housing
University of California, Santa Barbara
Storke Tower
Ocean Rd
Lagoon Rd
UCSB Lagoon
Campus Point
Goleta Slough State Marine Conservation Area

# Demo

# Application: Framework

Ruby on Rails serves as the MVC framework of the application.

Notable gems:

- Geokit / Geokit-rails
- Devise
- Ratyrate
- Paperclip
- Unicorn
- nginx

# Cloud Infrastructure: Storage

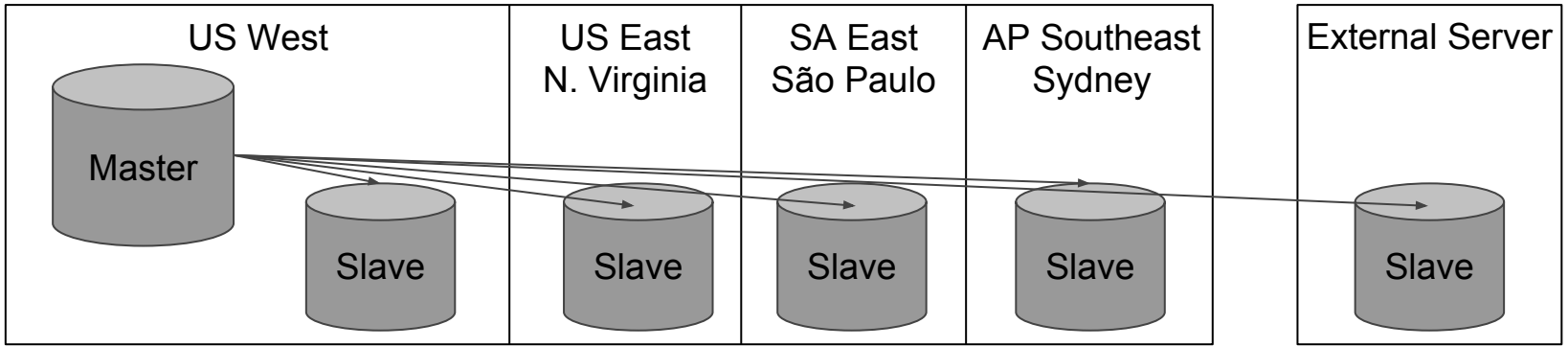
- Data (coordinations, names, ratings and other information) are stored in the database backend
- Pictures (large entities) are stored in AWS S3 (and Google bucket storage)

- We use MySQL database because our application requires relational structure that SQL-based databases make more sense

# Cloud Infrastructure: Database

In order to make the data available we use replication.

Master-Slave replication architecture on multiple regions:




Writes go to the master and reads go to the slaves.

# Cloud Infrastructure: Deploy the Application

Application is written in Ruby on Rails framework and is accessible from Github.

https://github.com/adityahemanth/lookup

We deploy the app with Unicorn and Nginx on Ubuntu 14.04 instances:

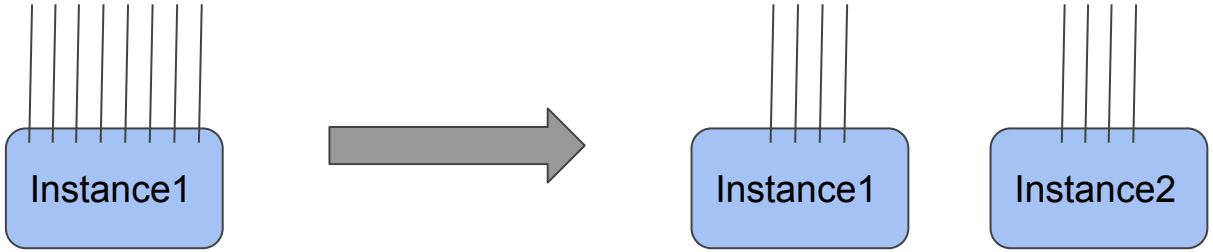
- Unicorn is an application server that enables Rails application to process requests concurrently
- Since Unicorn is not designed to be accessed by users directly, we are using Nginx as a reverse proxy that will buffer requests and responses between users and application

# Cloud Infrastructure: Deployment Details

- We created an instance, installed all prerequisites and built lookup app on it
- Then we created a bootable unicorn service as /etc/init.d/unicorn_lookup
- This service starts every time an instance is booted, serves lookup app in production environment and listens on shared/sockets/unicorn.sock
- Therefore, we run a Nginx service to redirect all requests on port 80 to the aforementioned unicron socket
- Last but not least, we have to open firewall for any IP on port 80

# Cloud Infrastructure: Scale in/out

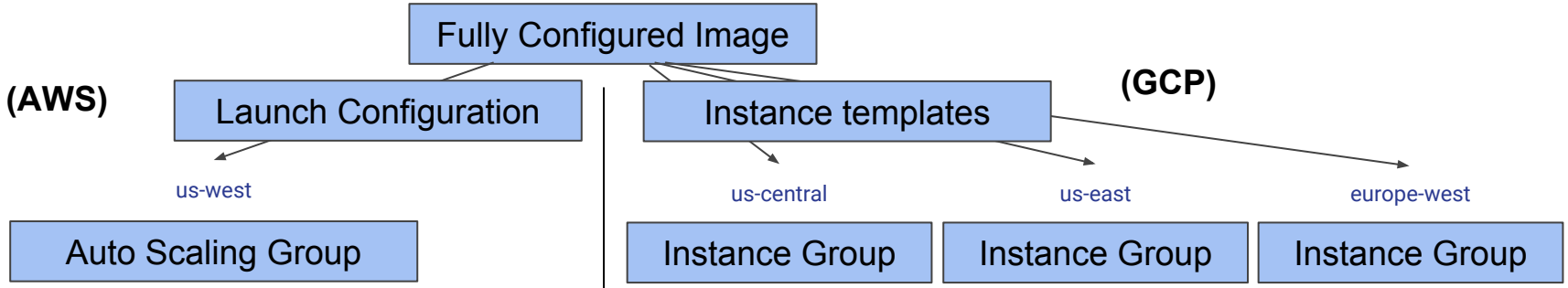
- Application is stateless and each instance can handle multiple users




- Thus, we need load balancers and auto scalars to do the rest for us

We are using Google Cloud Platform (GCP) and Amazon Web Services (AWS)

# Cloud Infrastructure: Scale in/out




Min: 5
Max: 40

When 60 <= CPU Utilization < 70: Add 1
When 70 <= CPU Utilization < 80: Add 2
When 80 <= CPU Utilization: Add 5

When 40 >= CPU Utilization > 30: Rem 1
When 30 >= CPU Utilization > 20: Rem 2
When 20 >= CPU Utilization> 10: Rem 3
When 10 >= CPU Utilization: Rem 5

Min: 2
Max: 8

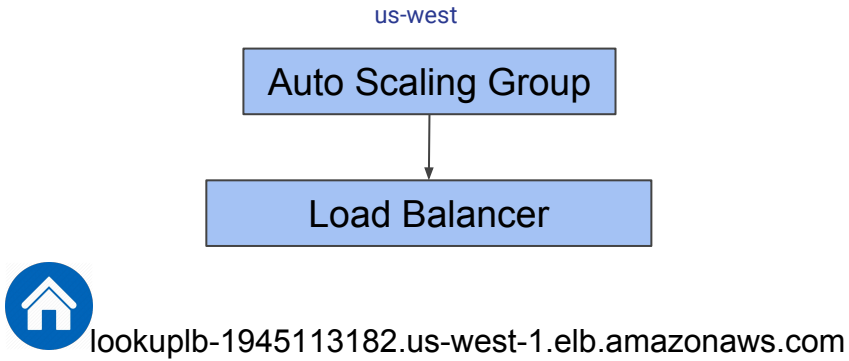
Min: 2
Max: 8

Min: 2
Max: 8

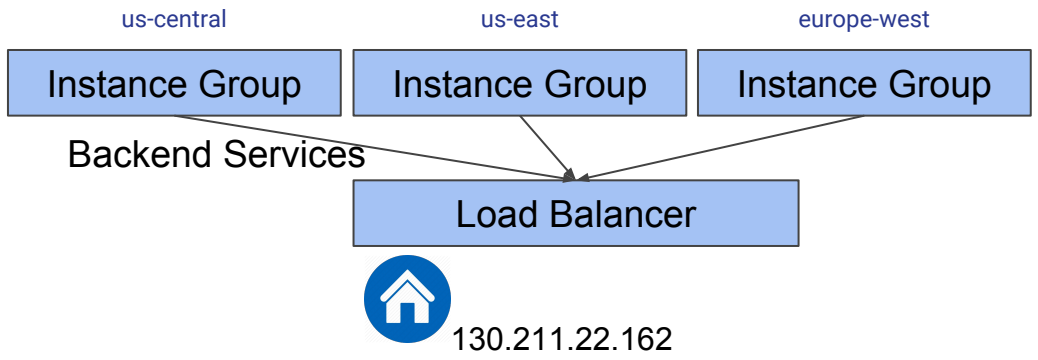
Auto scale such that target
CPU Utilization will be 60%

# Cloud Infrastructure: Load Balancer

**(AWS)**

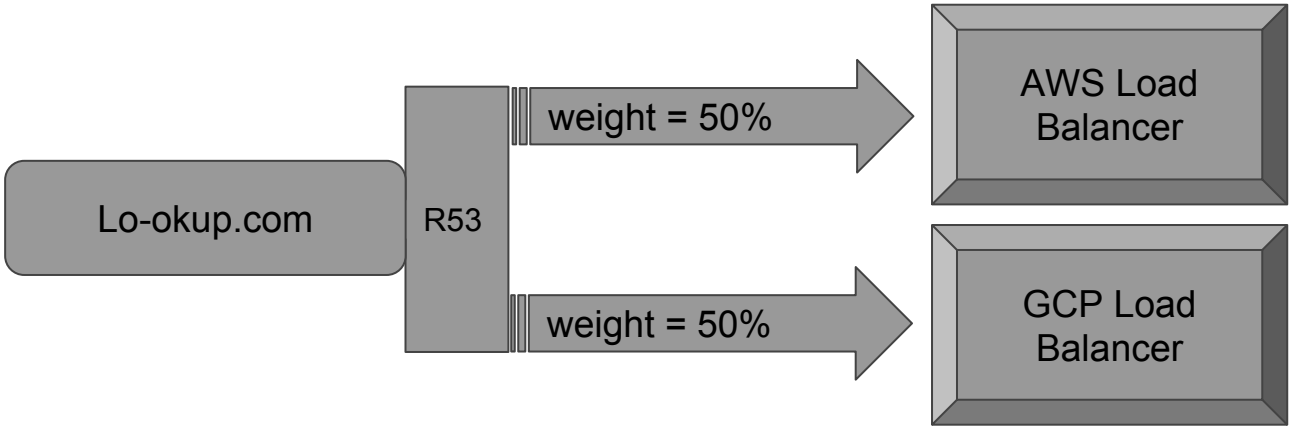



**(GCP)**

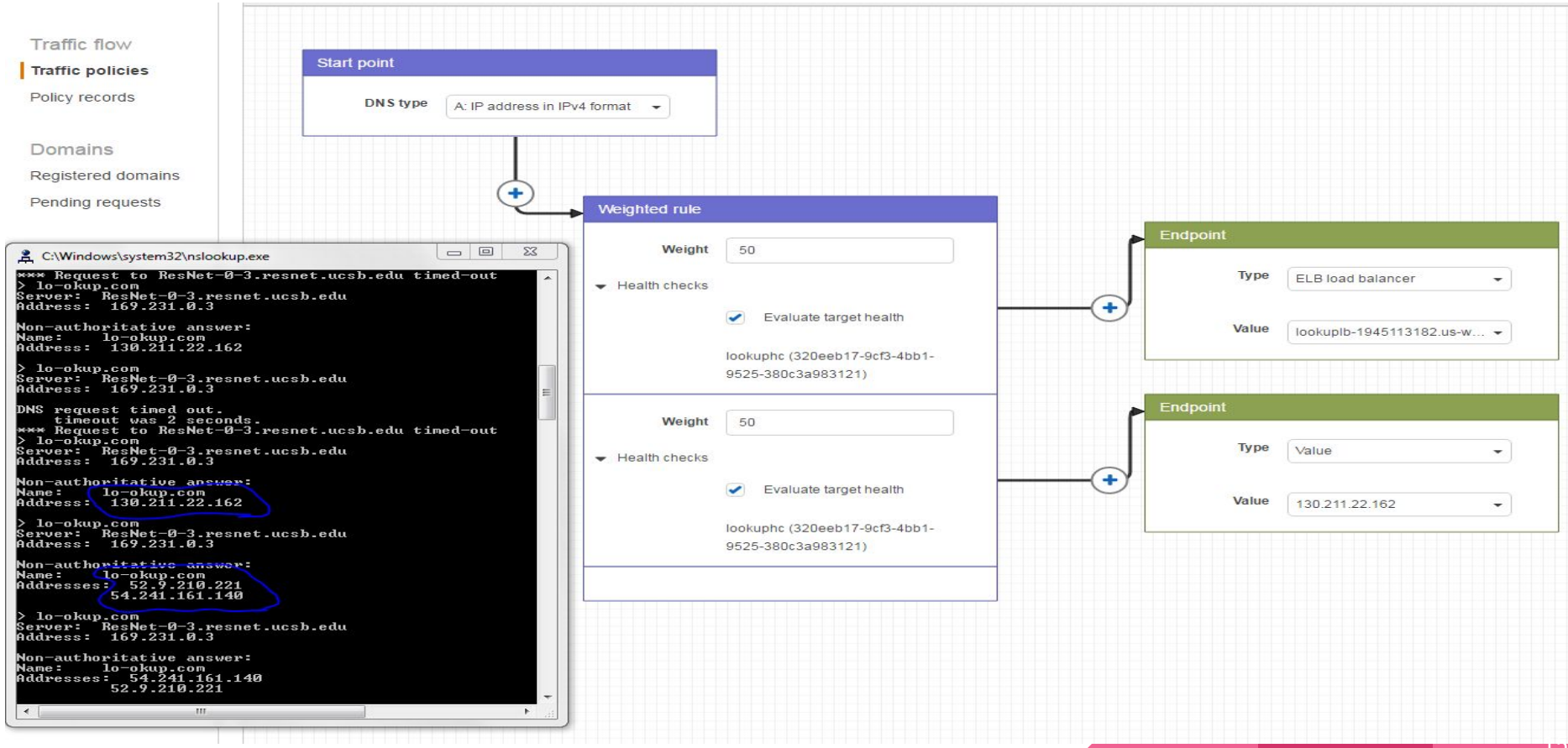


# Cloud Infrastructure: DNS

- Route 53 for Round Robin DNS
- A weighted policy for two backend servers
- We use load balancers' addresses for DNS



# Cloud Infrastructure: DNS

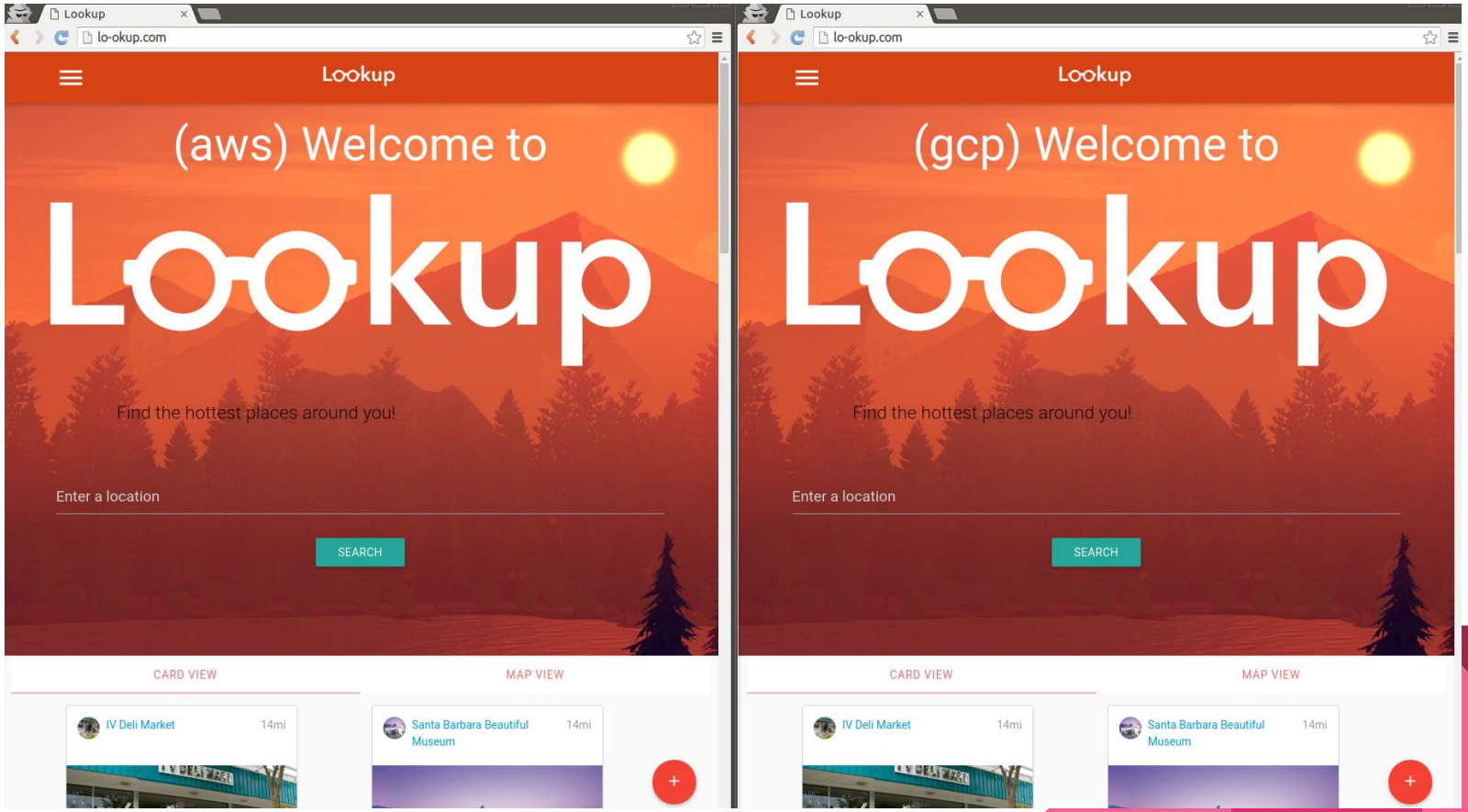


# All name systems

In order to have test the system properly, we created separate name systems (sub domains) as follows:

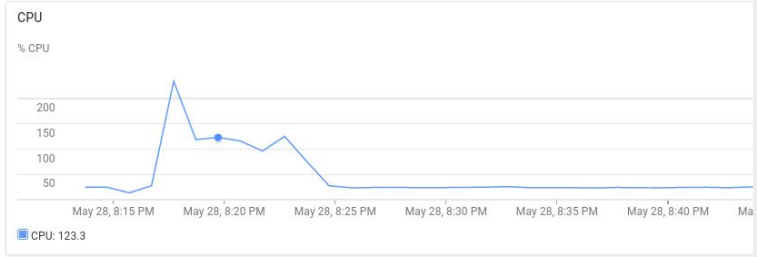
- s1.lo-okup.com
- aws.lo-okup.com
- gcp.lo-okup.com
- lo-okup.com

# lo-okup.com


Lookup
lo-okup.com
Lookup
(aws) Welcome to
Lookup
Find the hottest places around you!
Enter a location
SEARCH
CARD VIEW
MAP VIEW
IV Deli Market
14mi
Santa Barbara Beautiful Museum
14mi
Lookup
lo-okup.com
Lookup
(gcp) Welcome to
Lookup
Find the hottest places around you!
Enter a location
SEARCH
CARD VIEW
MAP VIEW
IV Deli Market
14mi
Santa Barbara Beautiful Museum
14mi

# Test

- Using simple python script:




| Name | Zone | Network | In use by | Internal IP | External IP | Connect |
|---|---|---|---|---|---|---|
| lookup-instance-group-2fun | europe-west1-b | default | | 10.132.0.2 | 146.148.28.67 | SSH |
| lookup-instance-group-3xky | europe-west1-b | default | | 10.132.0.4 | 130.211.64.106 | SSH |
| lookup-instance-group-bmxj | europe-west1-b | default | lookup-instance-group | 10.132.0.3 | 104.155.23.12 | SSH |
| lookup-instance-group-e6qy | europe-west1-b | default | lookup-instance-group | 10.132.0.5 | 104.155.21.85 | SSH |
| lookup-instance-group-s1fn | europe-west1-b | default | lookup-instance-group | 10.132.0.7 | 146.148.9.31 | SSH |

In [ ]:
```
# Omid55
# Scalability check for servers
import urllib.request as req
import threading
```

In [ ]:
```
def worker():
    lookupurl = 'http://gcp.lo-okup.com/'
    opens = 10000

    for i in range ( 1 , opens + 1 ):
        print ( i , end = " " )
        req.urlopen ( lookupurl )

    print( "DONE" )
```

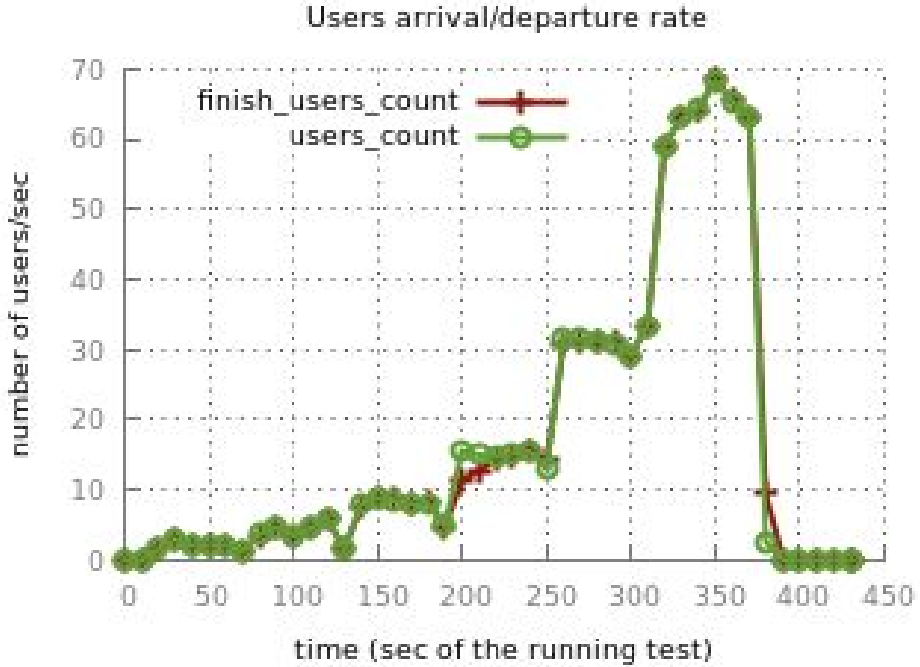
In [ ]:
```
threadsCnt = 5
threads = []
for i in range( threadsCnt + 1 ):
    t = threading.Thread( target = worker )
    threads.append( t )
    t.start()
```

In [ ]:

# Test using Tsung

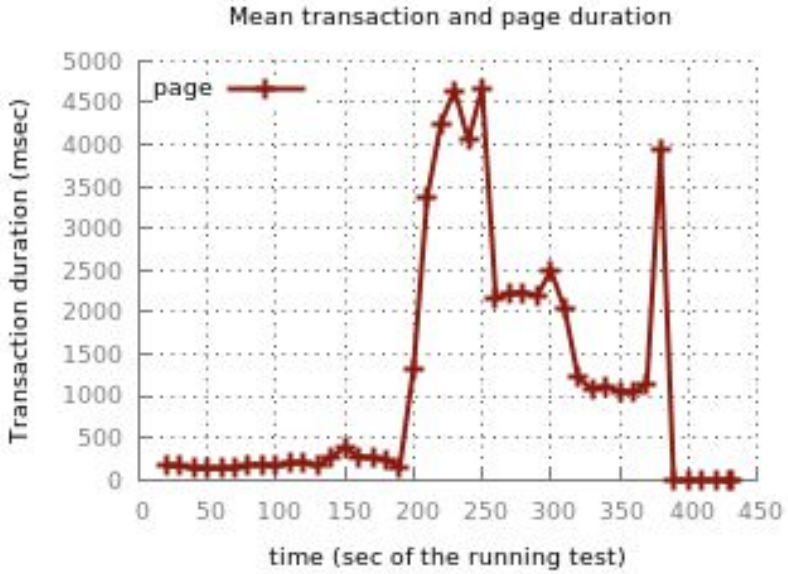
- We did not stop there for checking the availability and scaling properties!
- We used Tsung for HTTP load stress test

We start with 4 and ends with 64 users in incrementing order of time:



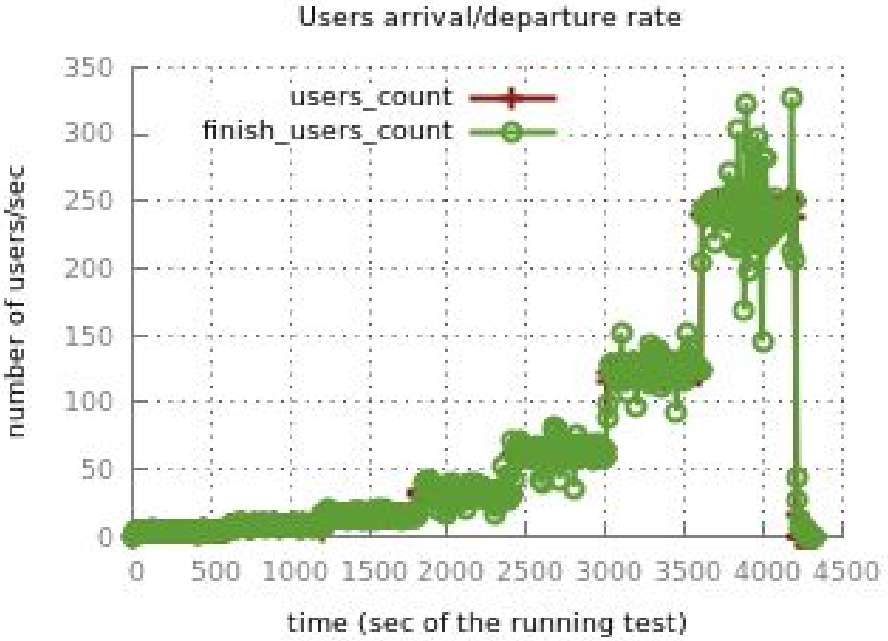
# Test

- Test on s1.lo-okup.com shows one micro-cpu can hanlde 16 users/second:



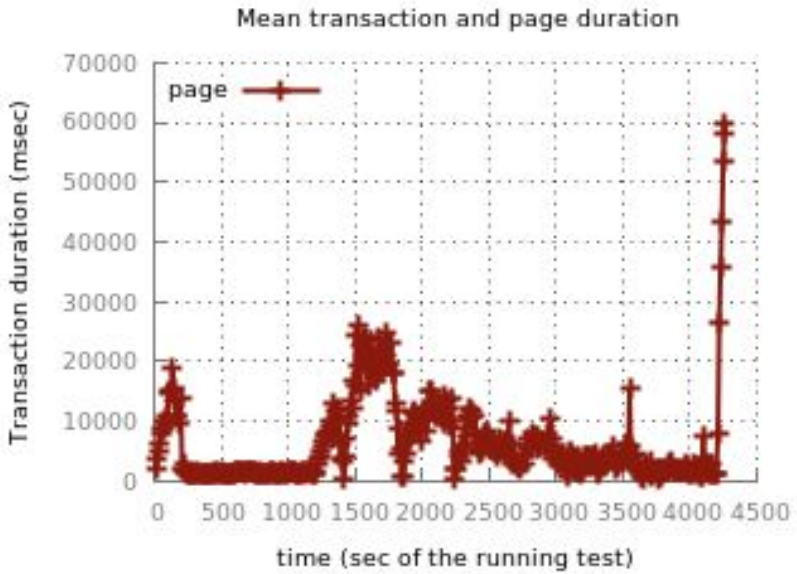

# Test

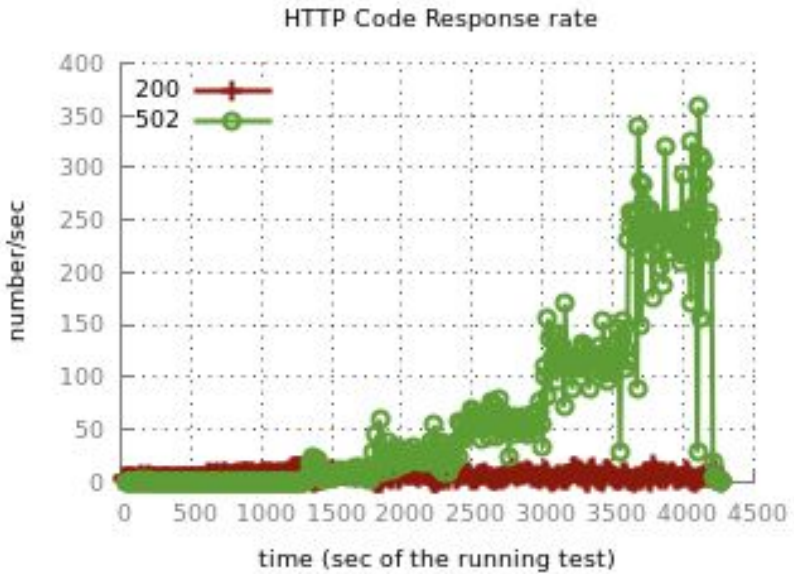
- We double number of users per second every 10 minutes
- It starts with 4 users/second and ends with 256 users/second



# Test

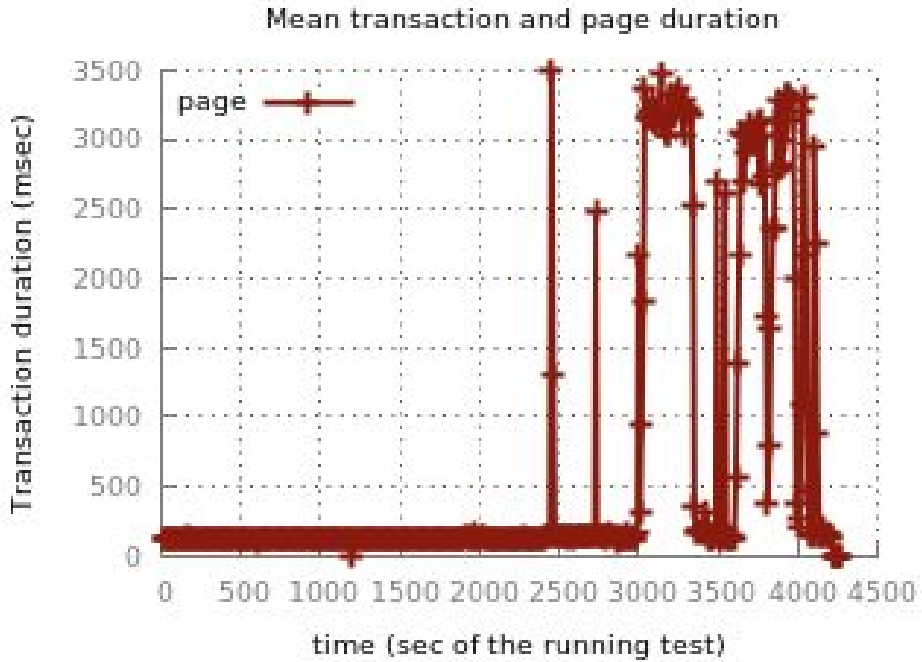
- Test on one of the load balancers (gcp.lo-okup.com):

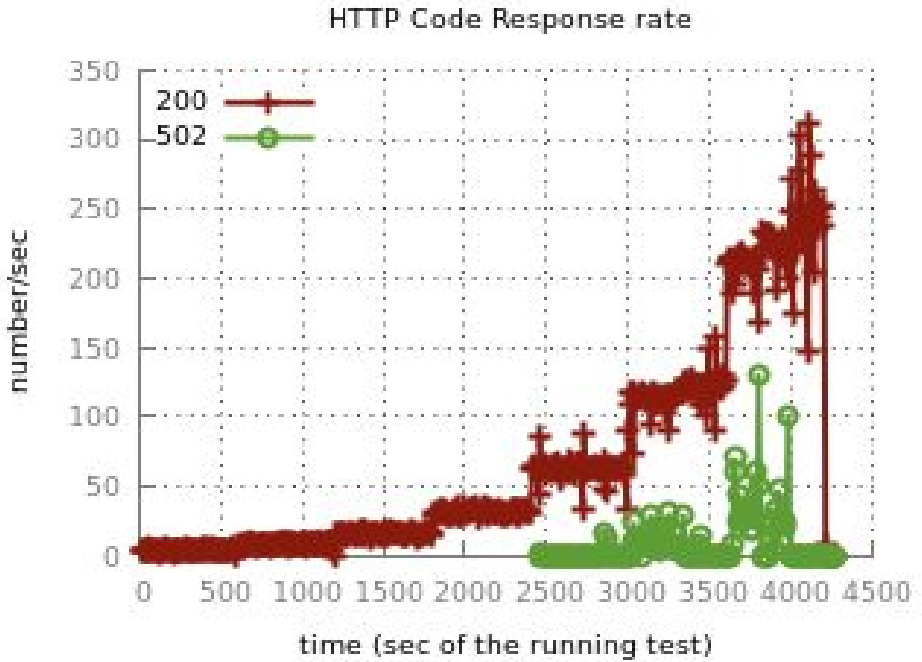


# Test

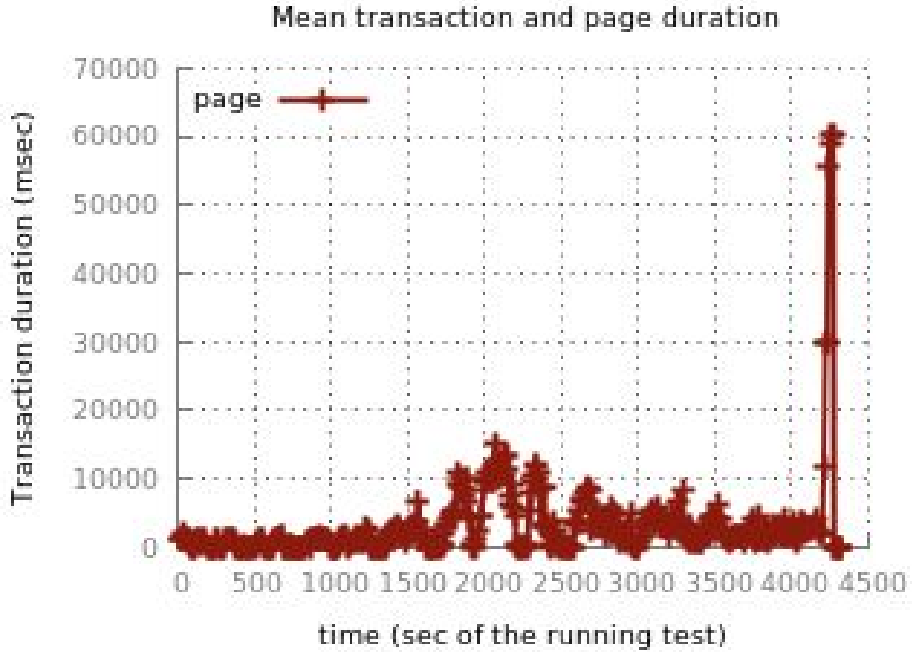
- Test on aws system (aws.lo-okup.com):

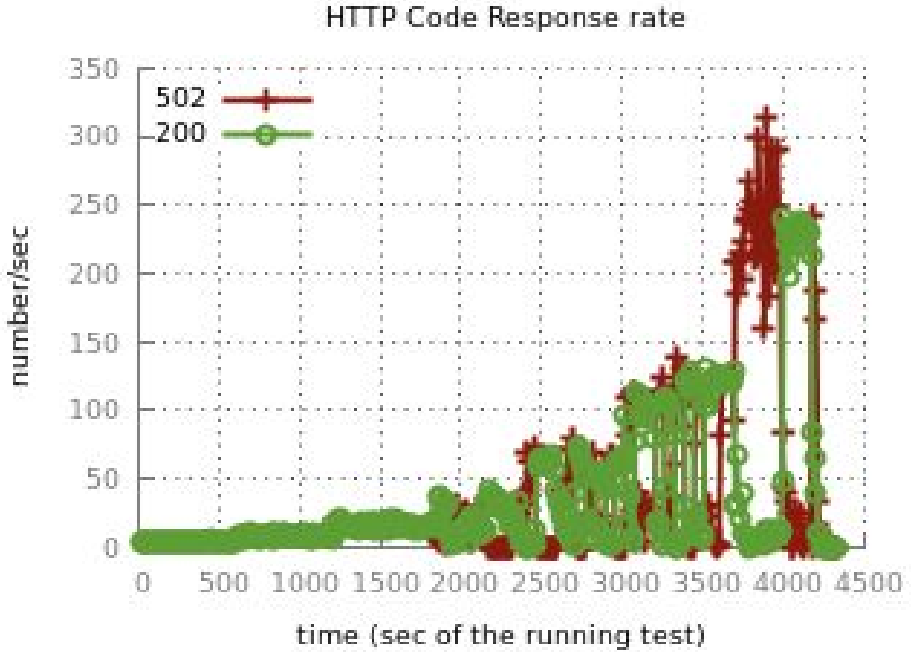


# Test

- Test on whole system (lo-okup.com):



# Future works

Route 53 dynamic weight changes.

Cost optimizations / load time optimizations

# Thank you