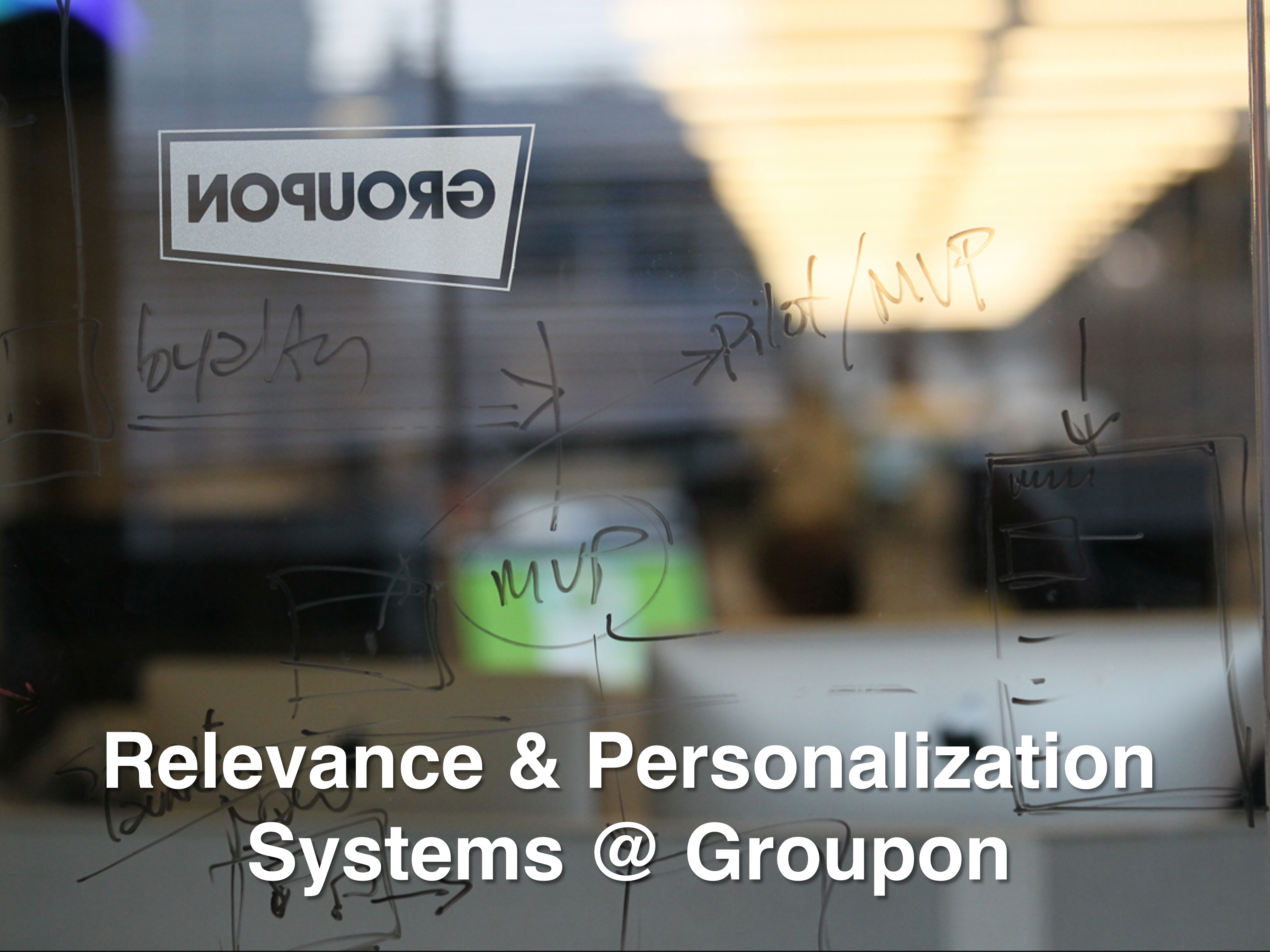
# Deal Personalization Systems @ Groupon

GROUPON®

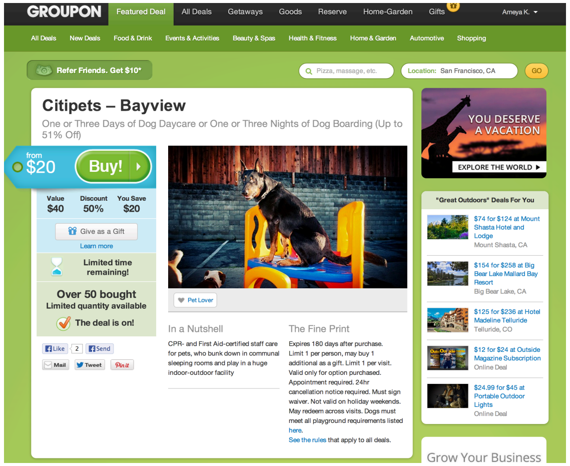

Ameya Kanitkar
ameya@groupon.com


Relevance & Personalization
Systems @ Groupon
Pilot / MVP
MVP

# What are Groupon Deals?

# Our Relevance Scenario

Users



# Our Relevance Scenario

Users

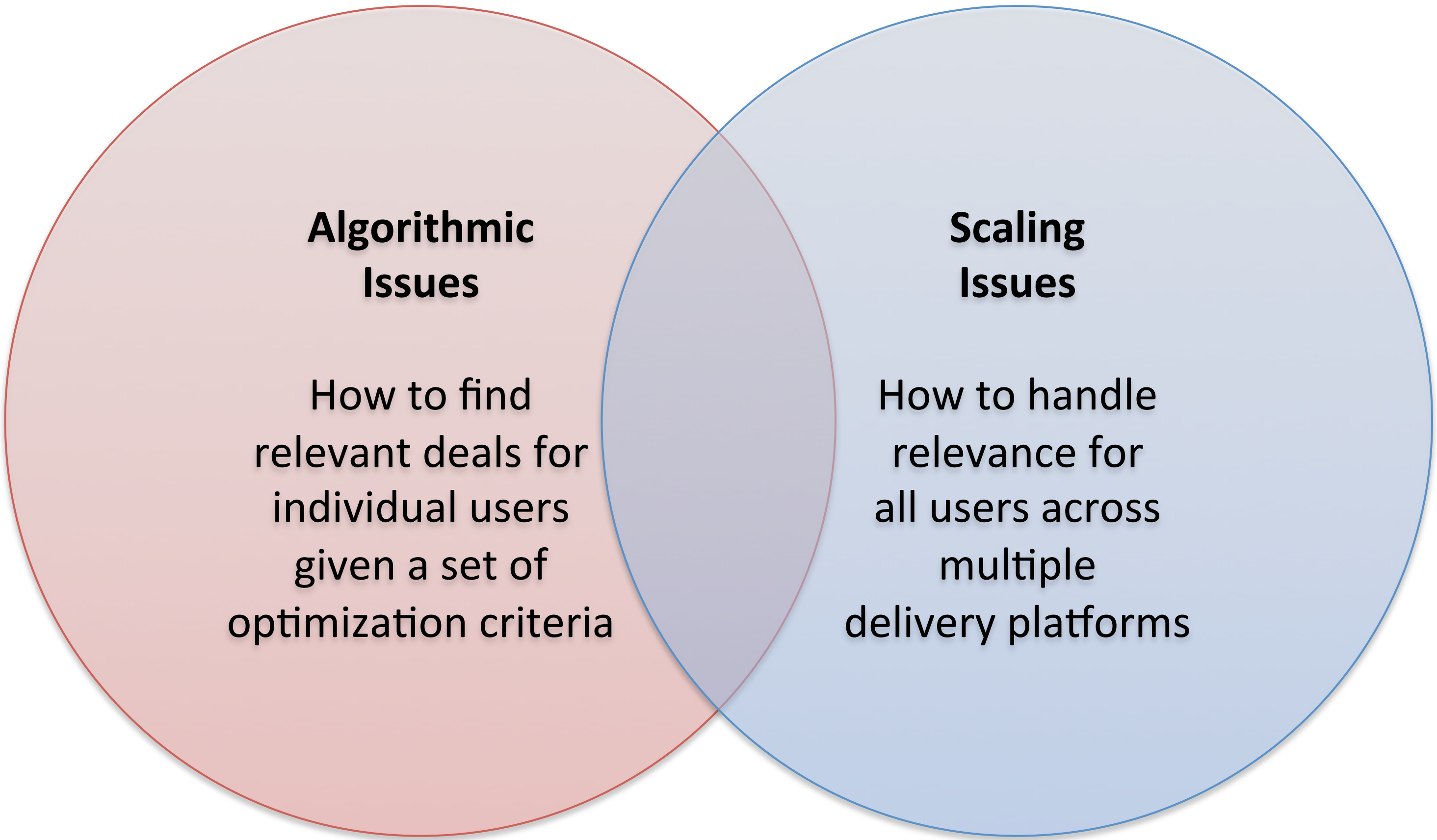
How do we surface relevant deals ?

- Deals are perishable (Deals expire or are sold out)
- No direct user intent (As in traditional search advertising)
- Relatively Limited User Information
- Deals are highly local




GROUPON

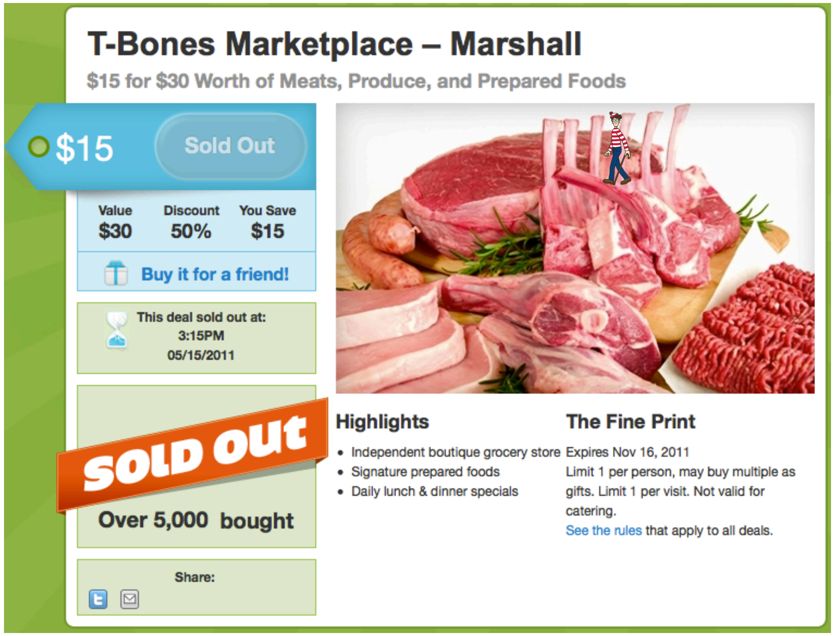
# Two Sides to the Relevance Problem



GROUPON

# Developing Deal Ranking Algorithms

- **Exploring Data**
  - ➡ Understanding signals, finding patterns
- **Building Models/Heuristics**
  - ➡ Employ both classical machine learning techniques and heuristic adjustments to estimate user purchasing behavior
- **Conduct Experiments**
  - ➡ Try out ideas on real users and evaluate their effect



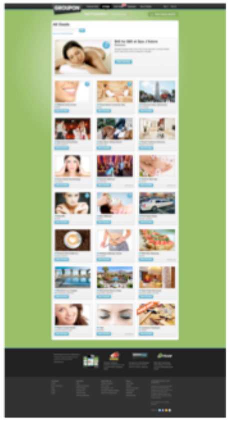
# Data Infrastructure

## Growing Deals

2011

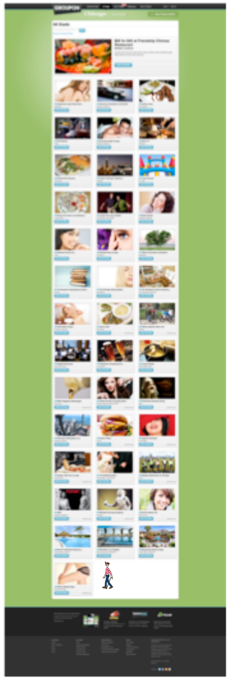
20+

2012



400+

2013



2000+

## Growing Users

- 100 Million+ subscribers
- We need  to store data like, user click history, email records, service logs etc. This tunes to billions of data points and TB's of data

# Deal Personalization Infrastructure Use Cases

## Deliver Personalized Emails


Personalize billions of emails for hundreds of millions of users

Offline System

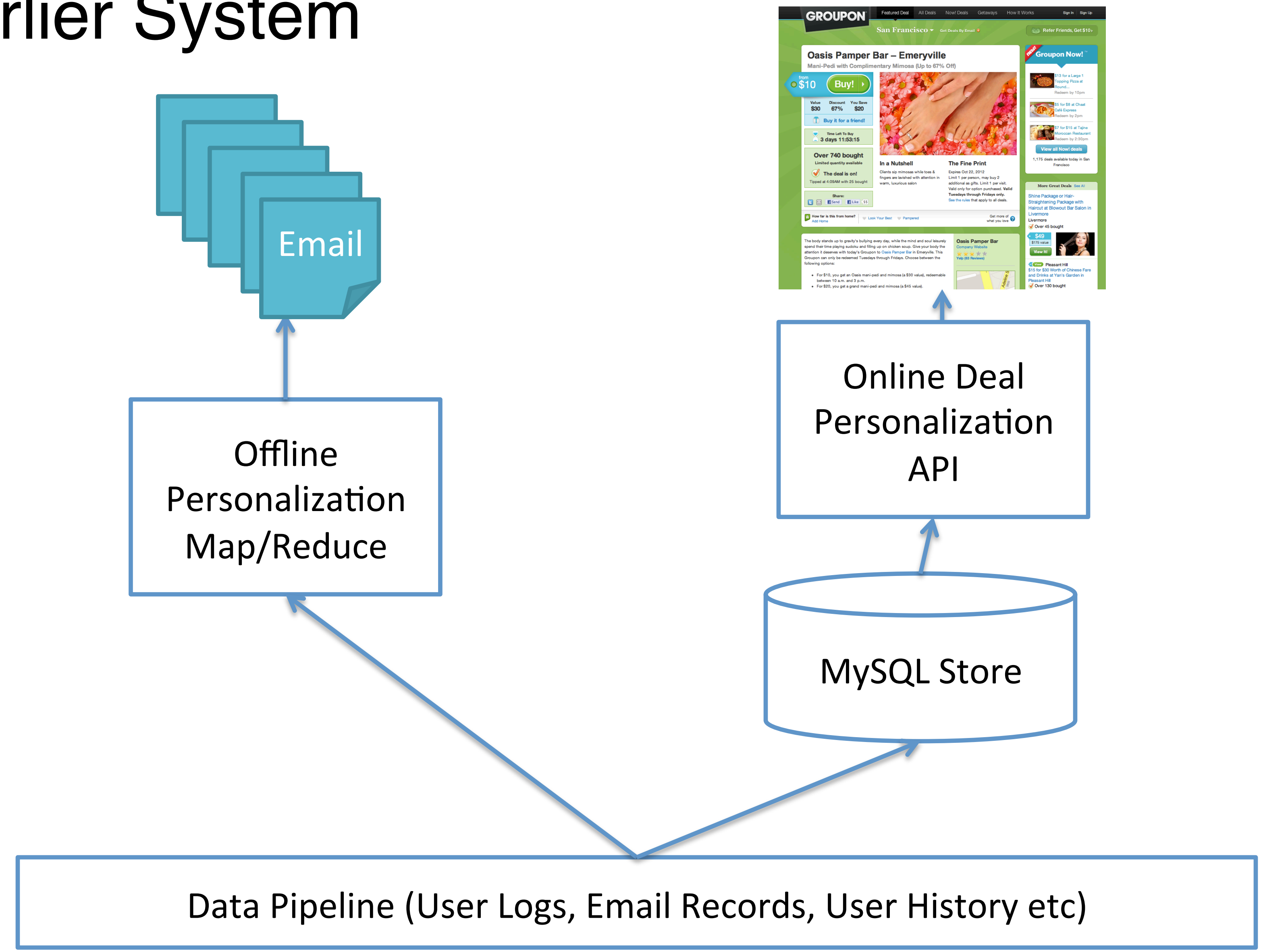
## Deliver Personalized Website & Mobile Experience



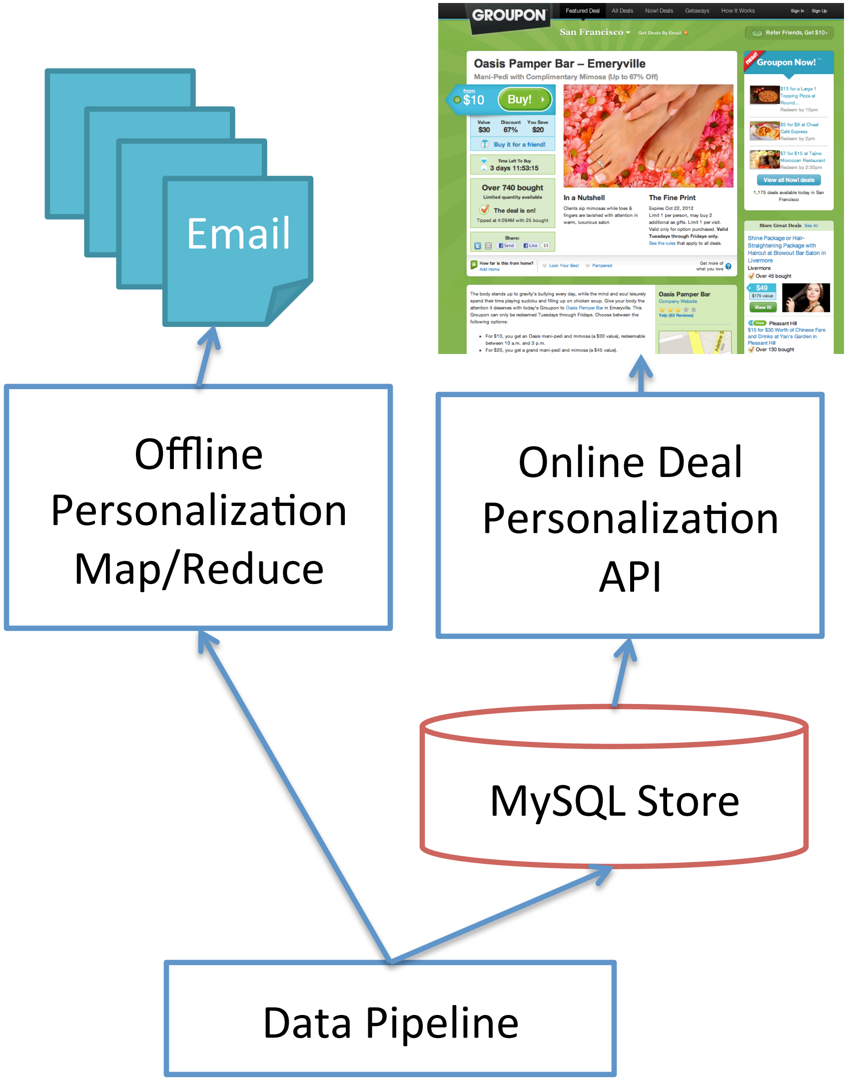
Personalize one of the most popular
e-commerce mobile & web app
for hundreds of millions of users & page views

Online System

# Earlier System

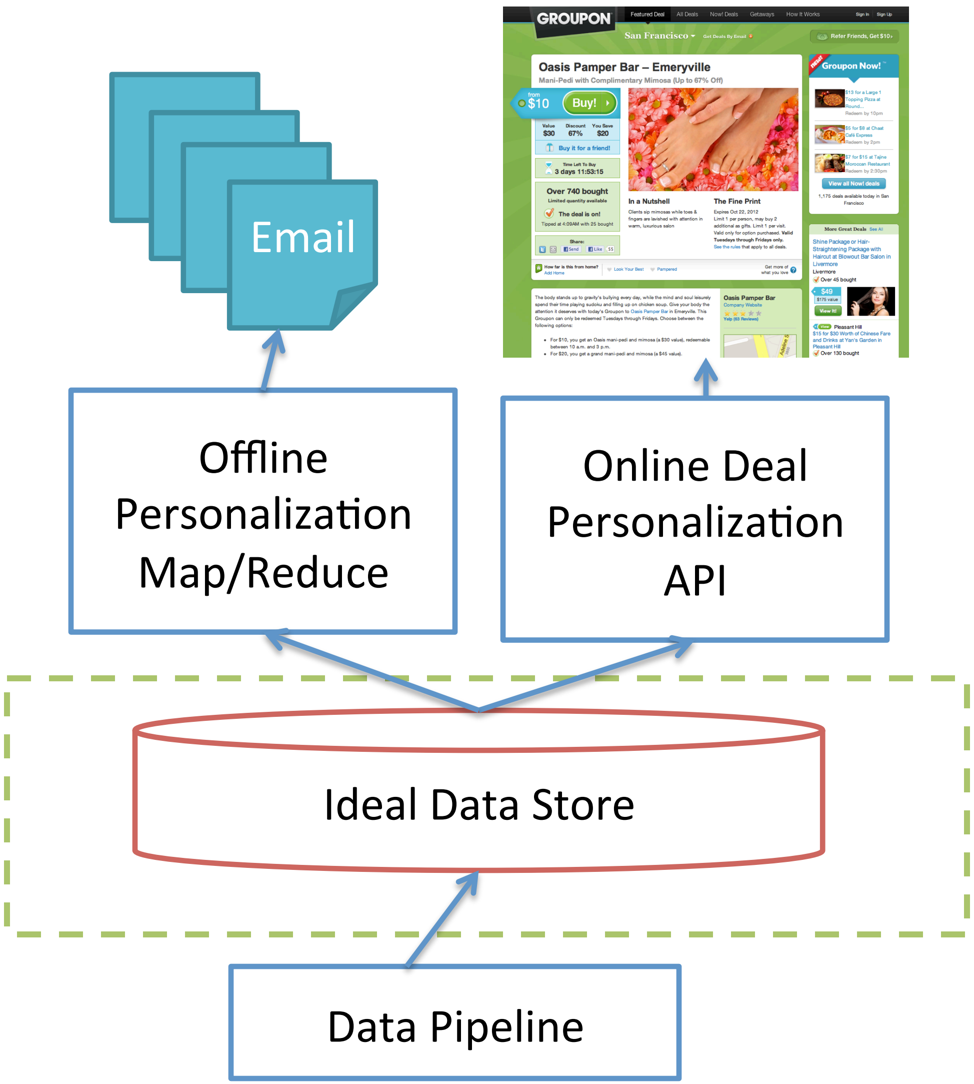


# Earlier System




- Scaling MySQL for data such as user click history, email records was painful unless we shard data
- Need to maintain two separate data pipelines for essentially the same data.

# Ideal System




- Common data store that serves data to both online and offline systems
- Data store that scales to hundreds of millions of records
- Data store that plays well with our existing hadoop based systems
- Data store that supports get() put() access patterns based on a key (User ID).

GROUPON

# Why HBase?

- Open Source distributed map data store modeled after Google’s Big Table
- Distributed Data Store: Store data on 1-700 node cluster. Linear scaling. Add capacity by adding more machines.
- Very light schema. Each row may have any number of columns. Columns need not be defined upfront. (Something like: Row1-> Map<byte[], byte[]>)

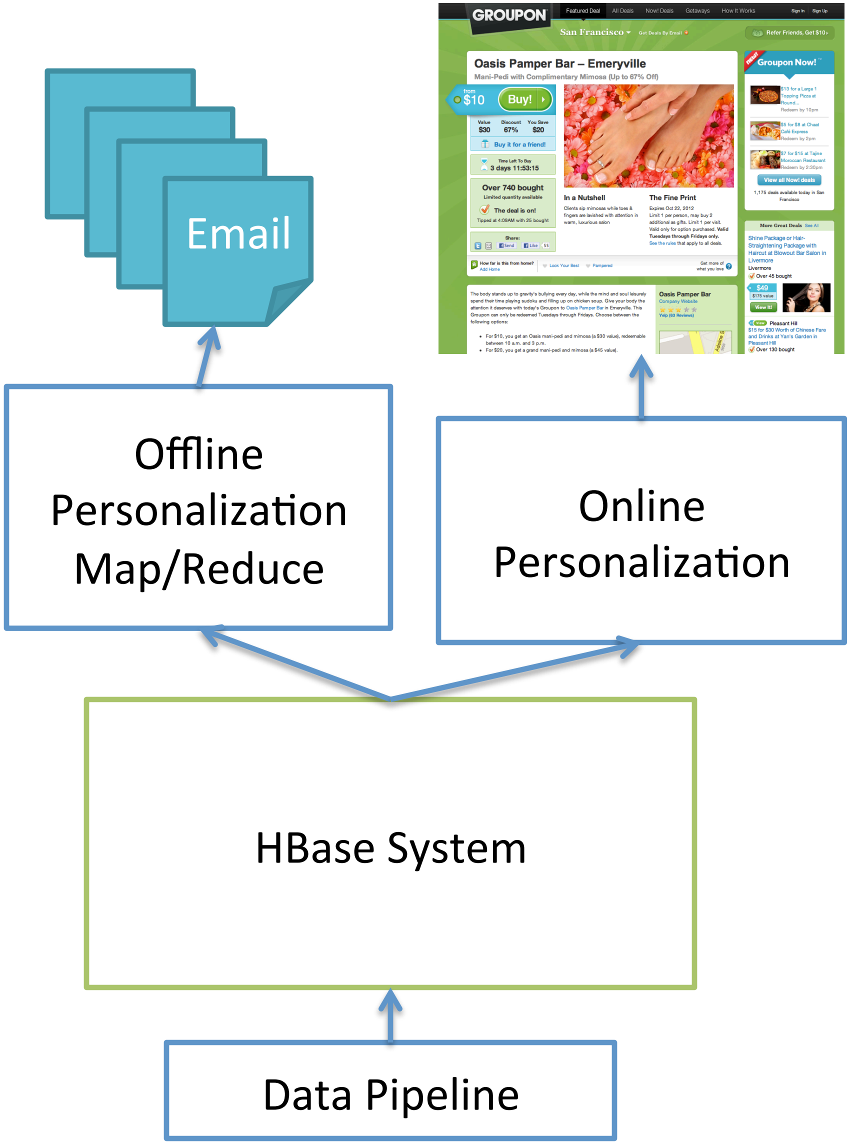
# Why HBase?

- Consistent Database. Highly available. Automatically shards/ scales. Can scale to billions of rows and multi terabyte data sizes
- Writes : 1-10 ms, Reads 20-50 ms
- Tight out of the box integration with Hadoop and Map Reduce

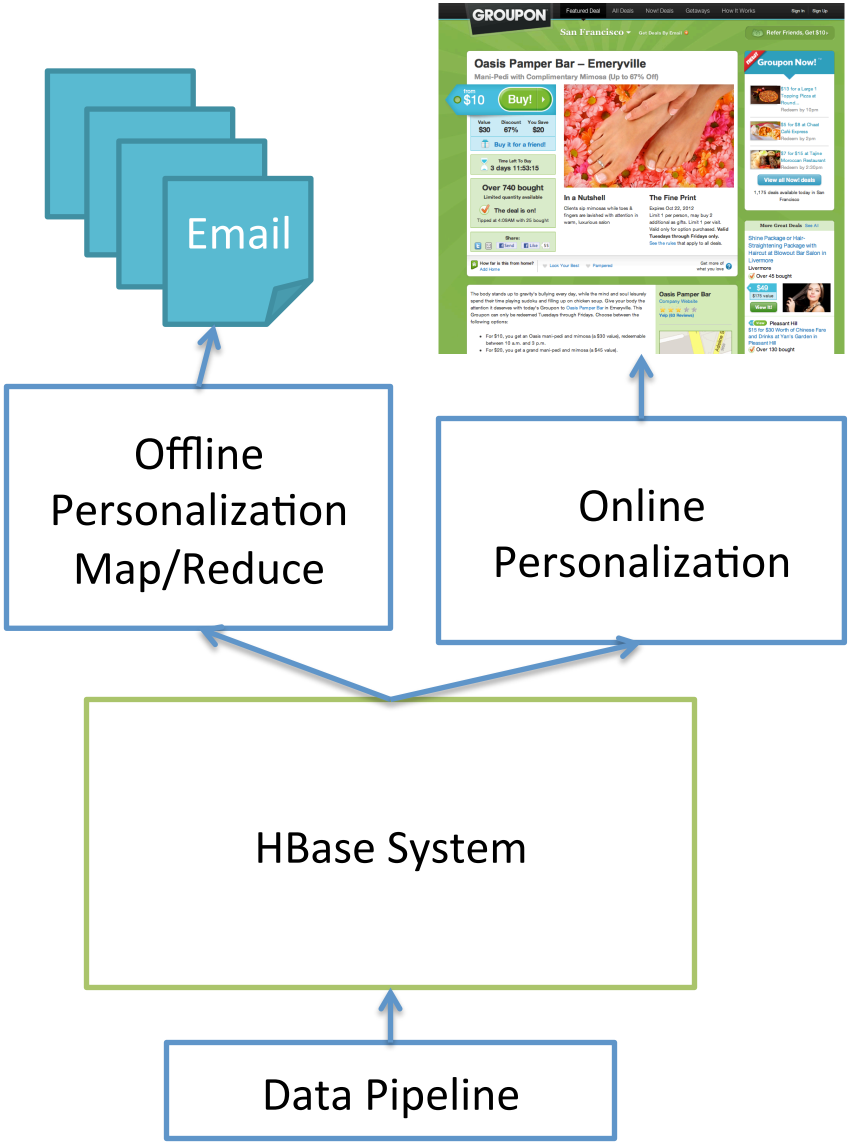
# HBase Table

| Row | Cf:<qual> | Cf:<qual> | .... | Cf:<qual> |
|---|---|---|---|---|
| row1 | Cf1:qual1 | Cf1:qual2 | | |
| row11 | Cf1:qual2 | Cf1:qual22 | Cf1:qual3 | |
| row2 | Cf2:qual1 | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| rowN | | | | |

# Architecture Options



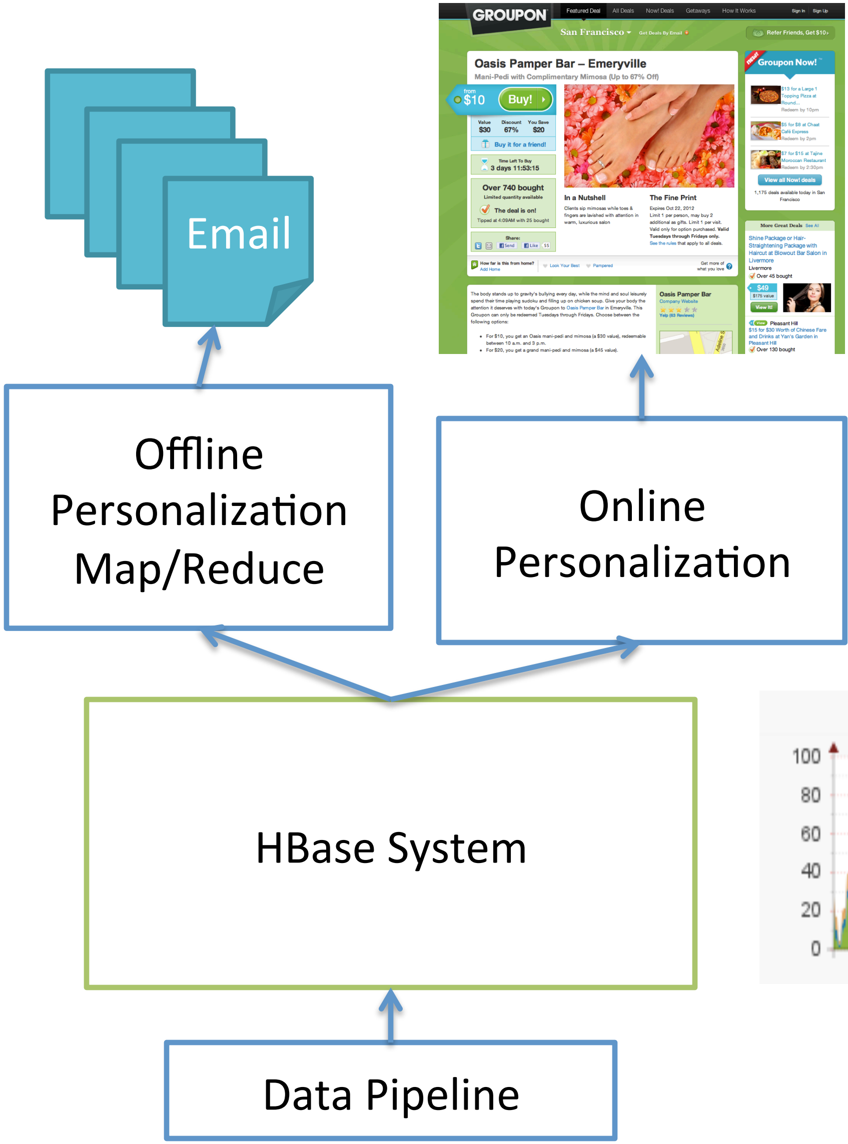
# Architecture Options




Pros

- Simple design
- Consolidated system that serves both online and offline personalization

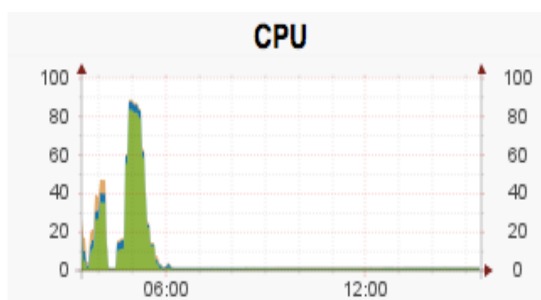
# Architecture Options

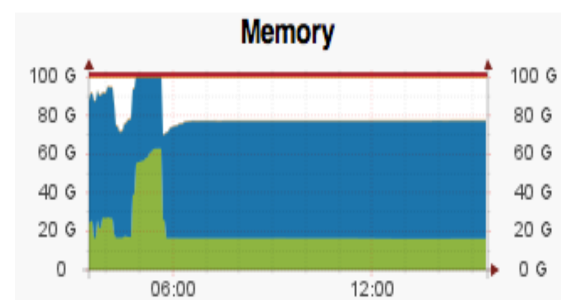



Cons

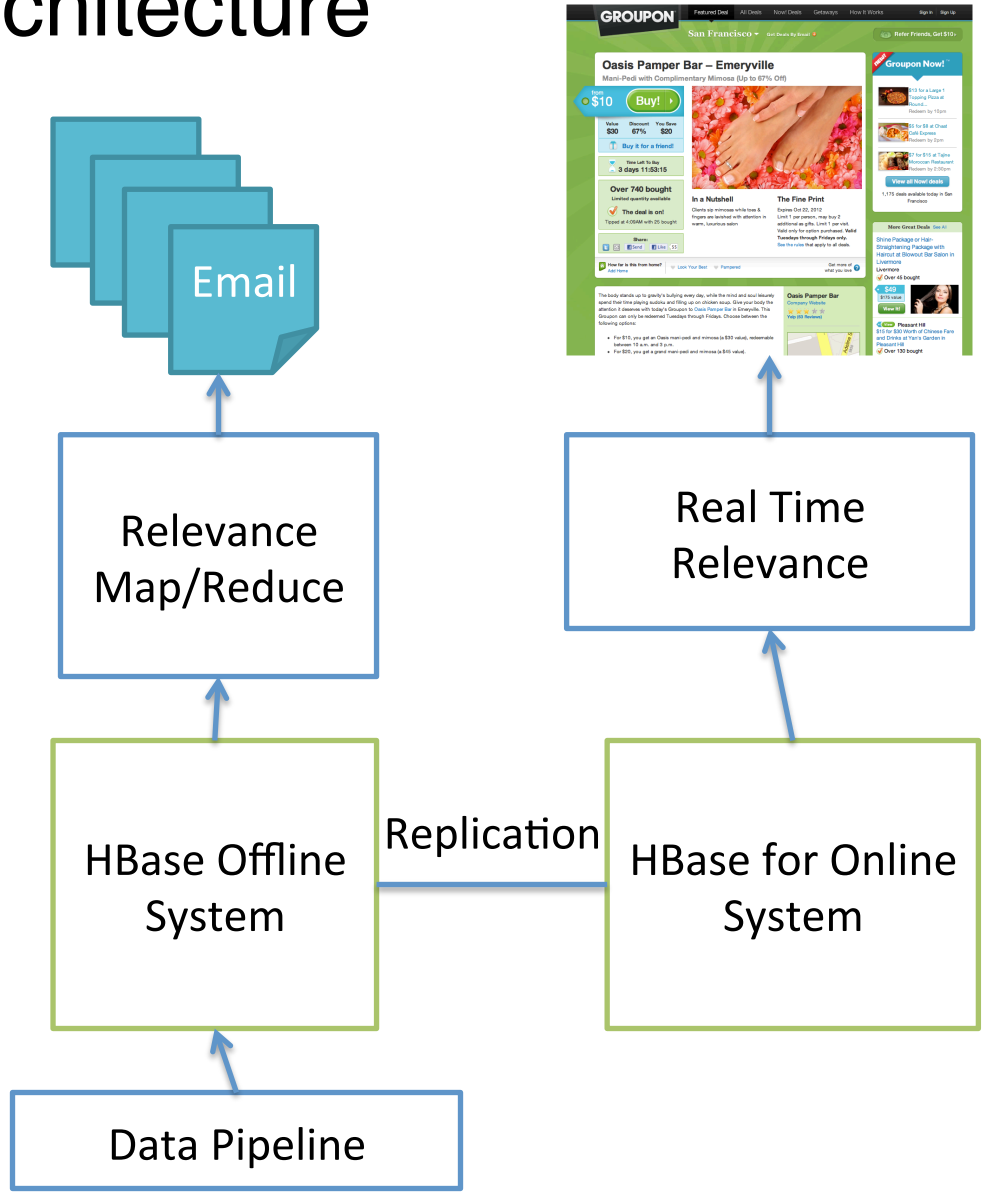
- We now have same uptime SLA on both offline and online system
- Maintaining online latency SLA for bulk writes and bulk reads is hard.

And here is why…



# Architecture




- We can now maintain different SLA on online and offline systems
- We can tune HBase cluster differently for online and offline systems

# HBase Schema Design

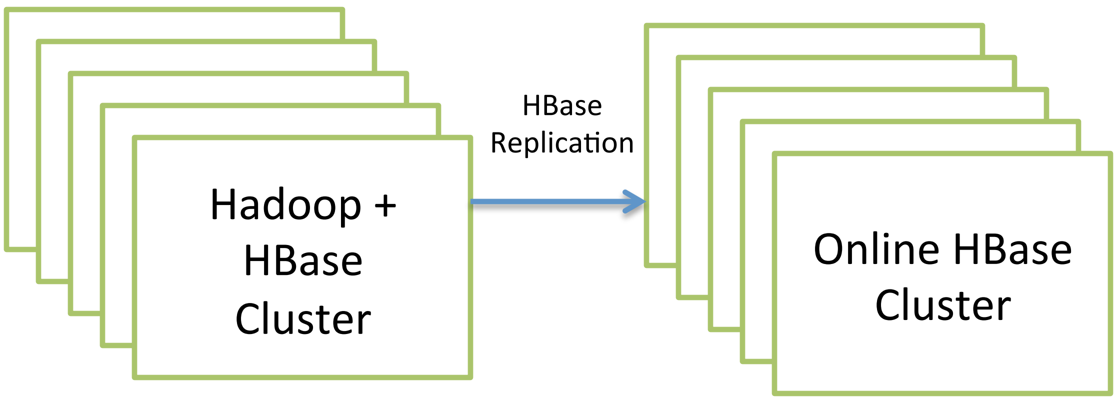
| User ID | Column Family 1 | Column Family 2 |
|---|---|---|
| Unique Identifier for Users | User History and Profile Information | Email History For Users |

Overwrite user history and profile info

Append email history for each day as a separate columns. (On avg each row has over 200 columns)

- Most of our data access patterns are via “User Key”
- This makes it easy to design HBase schema
- The actual data is kept in JSON

# Cluster Sizing




100+ machine Hadoop cluster, this runs heavy map reduce jobs
The same cluster also hosts 15 node HBase cluster

10 Machine dedicated HBase cluster to serve real time SLA

- **Machine Profile**
- **96 GB RAM (HBase 25 GB)**
- **24 Virtual Cores CPU**
- **8 2TB Disks**

- **Data Profile**
- **100 Million+ Records**
- **2TB+ Data**
- **Over 4.2 Billion Data Points**

# Other Takeaways

- Choose data storage format carefully. (We are using JSON, but one can consider Avro, Protobufs etc)
- Always store compressed data. We use LZO, its easy to map reduce

- Always store processed data in HBase.
- HBase needs some tuning before it scales. Tuning garbage collection is important. So is various timeouts and caching parameters, cluster can be unstable without these tuning parameters.

# Questions?


# Questions?

# Thanks!

ameya@groupon.com

www.groupon.com/techjobs