W4156

Scalability

Agenda

❏ Success! Our app explodes in popularity - will it still work?!?!?

❏ Defining Scalability

❏ Scaling Web-Architectures

❏ Scale Cube

❏ Summary

Defining Scalability

Success!!!

Your project was featured on Reddit/ Product Hunt and tomorrow you have x100 users and you are growing 2% per day

Hmmm

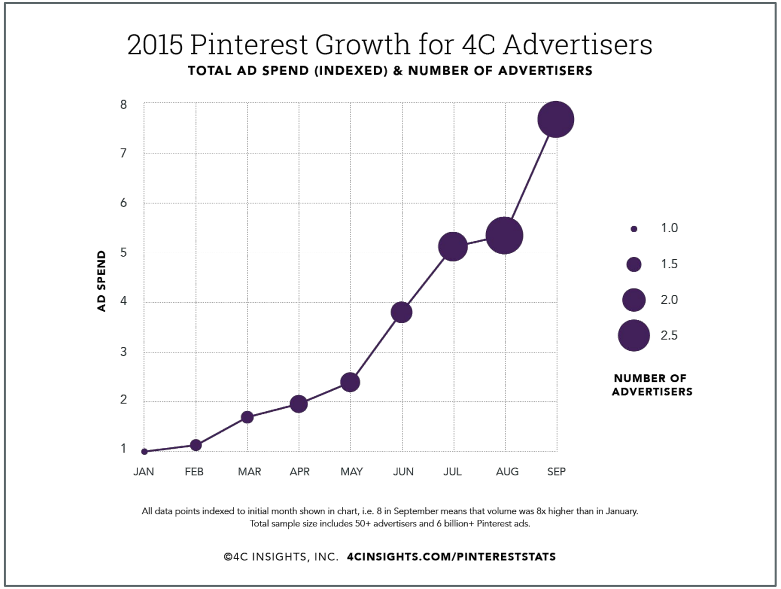
Q1: Will it handle this load?

Q2: Could it handle x10,000?

Q3: At what point will it break?

Q4: What changes are required?

Rapid *Growth*

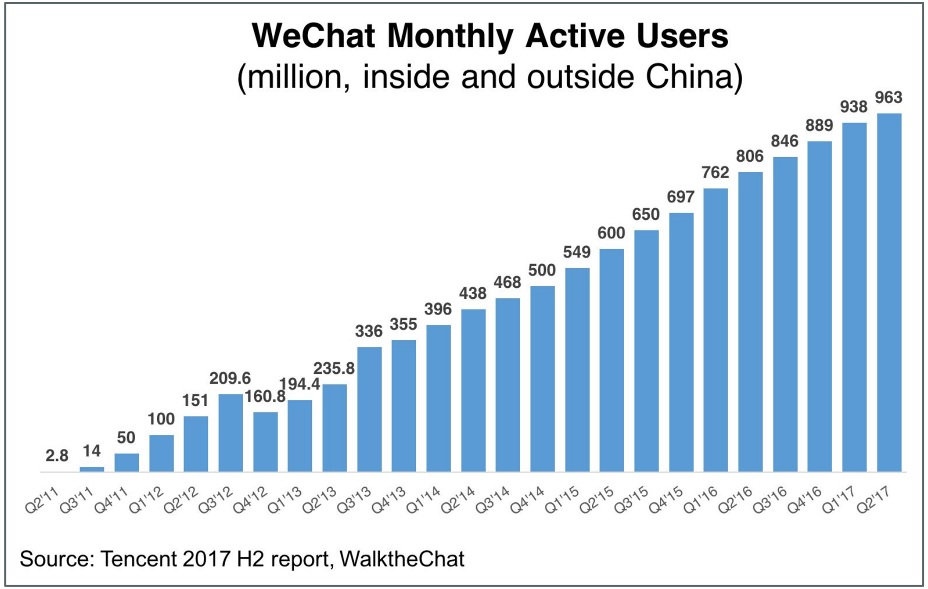


2015 Pinterest Growth for 4C Advertisers

TOTAL AD SPEND (INDEXED) & NUMBER OF ADVERTISERS

All data points indexed to initial month shown in chart, i.e. 8 in September means that volume was 8x higher than in January.
Total sample size includes 50+ advertisers and 6 billion+ Pinterest ads.

©4C INSIGHTS, INC. **4CINSIGHTS.COM/PINTERESTSTATS**

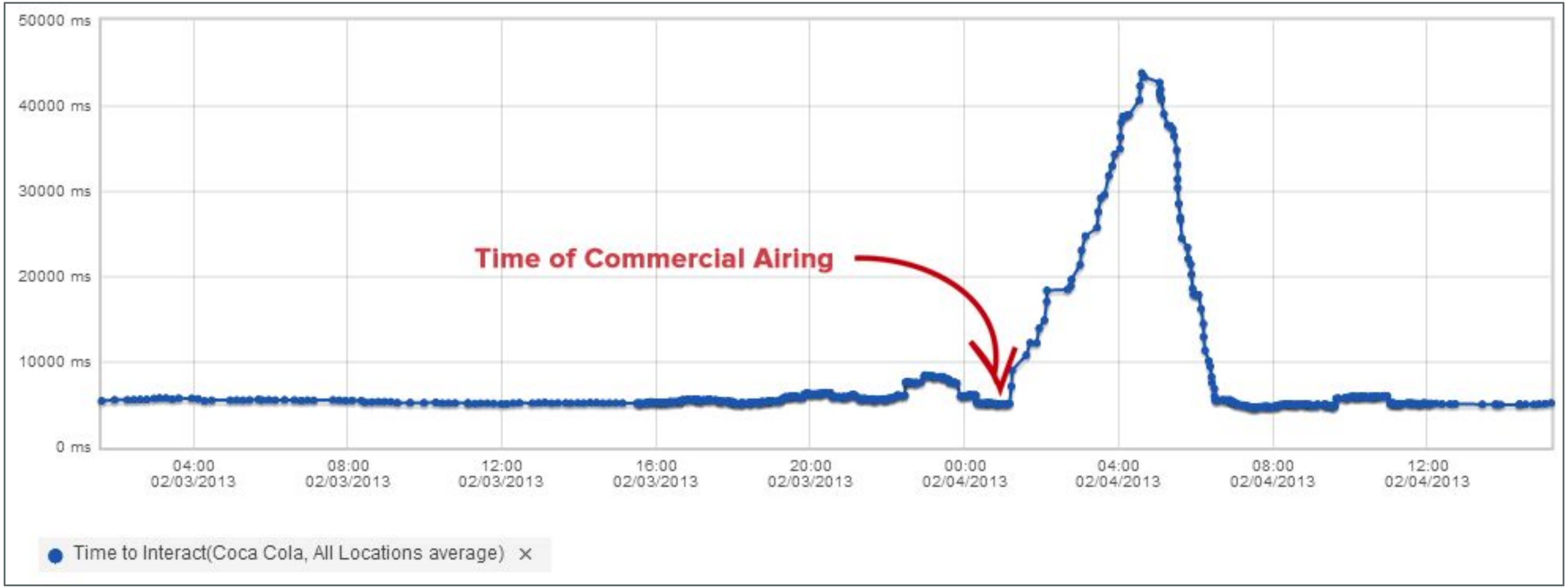


WeChat Monthly Active Users
(million, inside and outside China)

Source: Tencent 2017 H2 report, WalktheChat

Sudden *Surges*

Spend Millions on Advertising for your website to Crash ....



Superbowl Advertiser Crash

Systems seem to fail at the point when they are most needed / greatest opportunity?

Scalability

A *system* is *scalable* if it can

1. quickly accommodate an *increase* or *decrease* in usage
   (usage = #users / #transactions / data volume)

whilst maintaining acceptable

2. QoS
3. financial cost
4. administrative cost
5. developer productivity

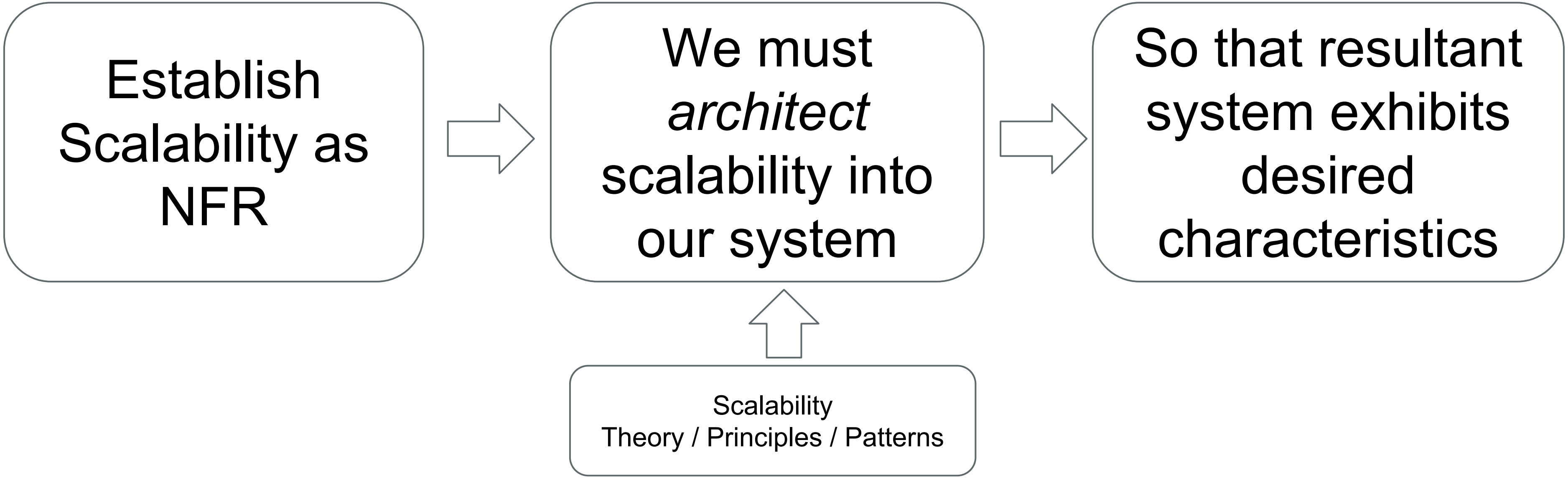
Scalability != Performance

If your system is deathly slow with 1 user you have a *performance problem*

If your system cannot add N users or cannot meet QoS @ increased load you have a *scalability problem*

Are we ready to architect scalability?

Architecting in Scalability



Establish Scalability as NFR

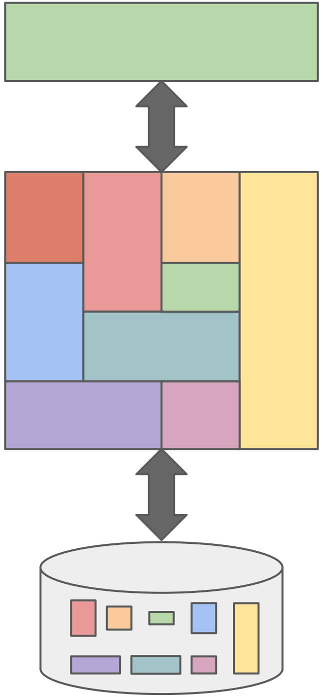
We must *architect* scalability into our system

So that resultant system exhibits desired characteristics

Scalability
Theory / Principles / Patterns

We must consider key NFRs but similar approach. Availability is similar (requirement, architecture decisions, etc). However, we have finite time so will focus on scalability as one example of architecting NFRs

Let's use a 3-tier web app exemplar



Step 0: Stop and Think

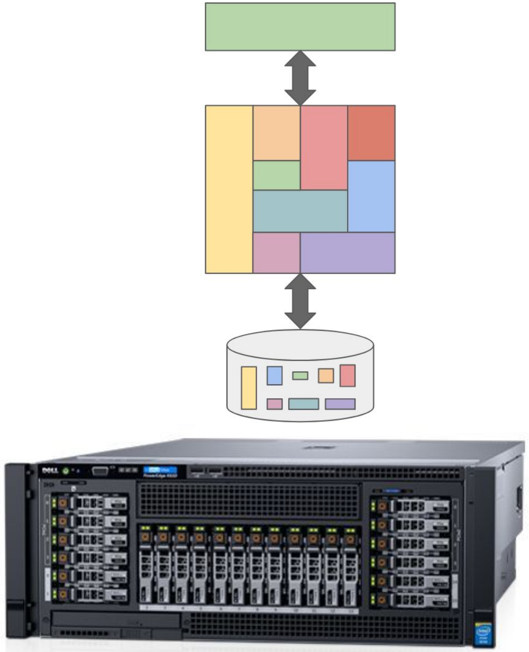
1. What is the scalability requirement?

2. What is the current limiting factor based on problem domain and application usage profile?
    i. Which use-case is scale limiting?
    ii. What is the read vs write profile of the app?

Disclaimer: I will run through a set of tactics/evolution. Ordering is likely to be application specific (what is app bottle neck)

Tactic 1a: Buy a bigger/faster server

$729



$18,099

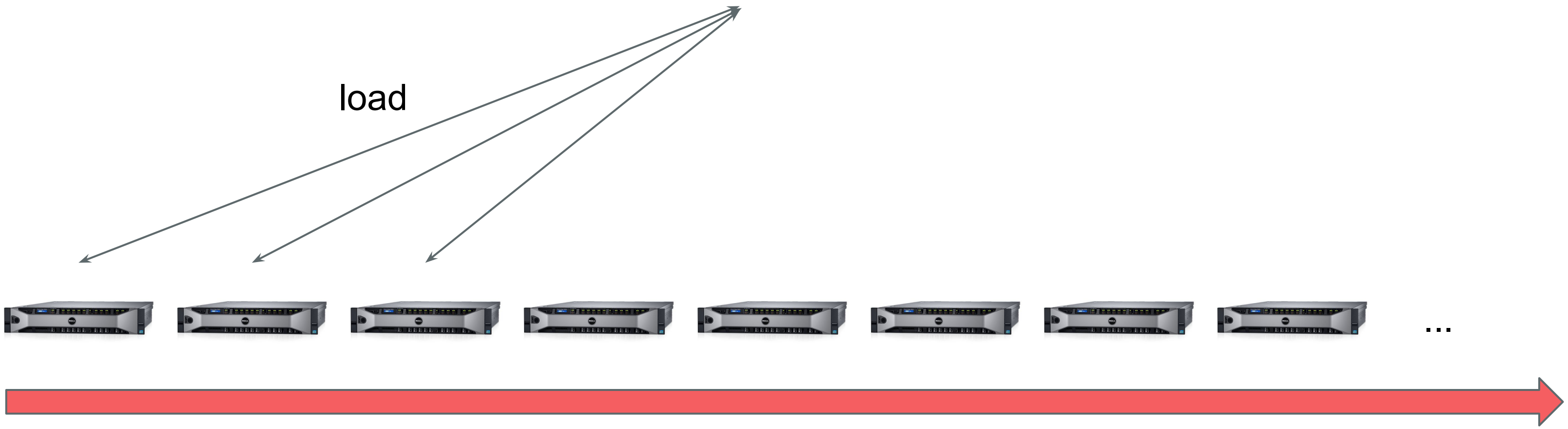
Scale by buying bigger h/w

Or AWS Instance Types

Tactic 1a: Vertical Scaling

Pros	Cons
Quick (Buy and Switch)	Stops working / limits! At some point there are no bigger servers!
Easy to manage (1 server)	$/Flop not efficient
	(Likely need to run >1 server for availability anyway so can't avoid)

Tactic 1b: Scale Horizontally

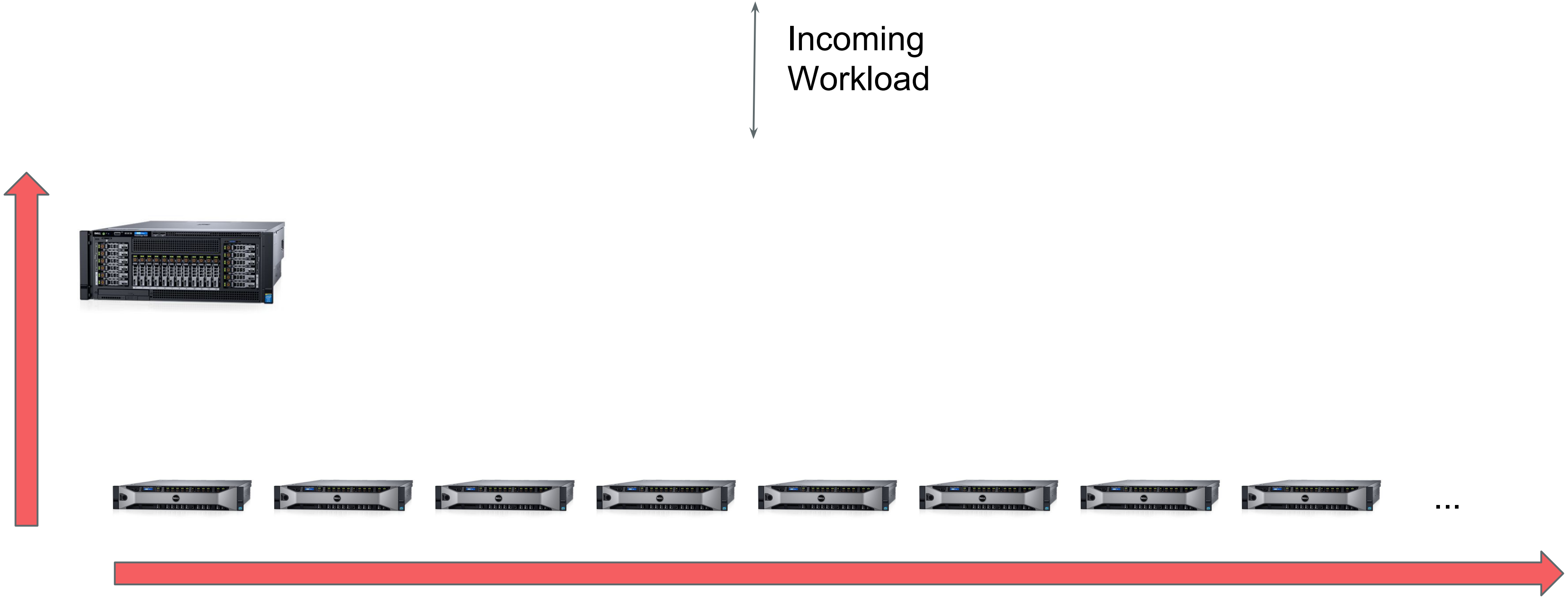


Scale by adding more hardware

Tactic 1b: Scale Horizontally

Pros	Cons
Increased headroom (vs vertical)	Distributed system (oh oh)
Better $/Flop (commodity machines)	More complex deployment and maintenance
H/W redundancy	More complex impact on design

Vertical and Horizontal

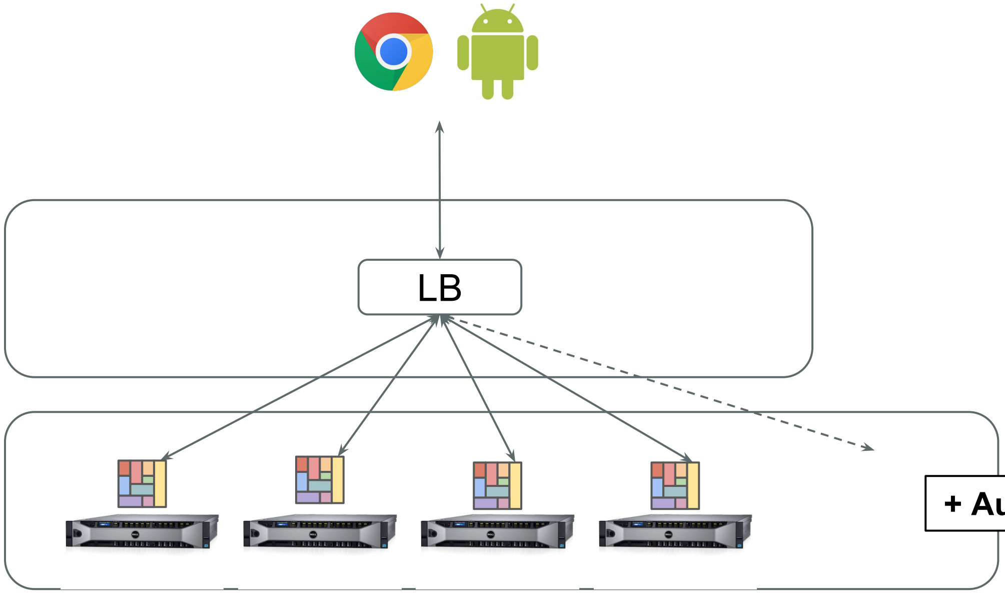


Incoming Workload

...

Tactic 3: Load Balancers

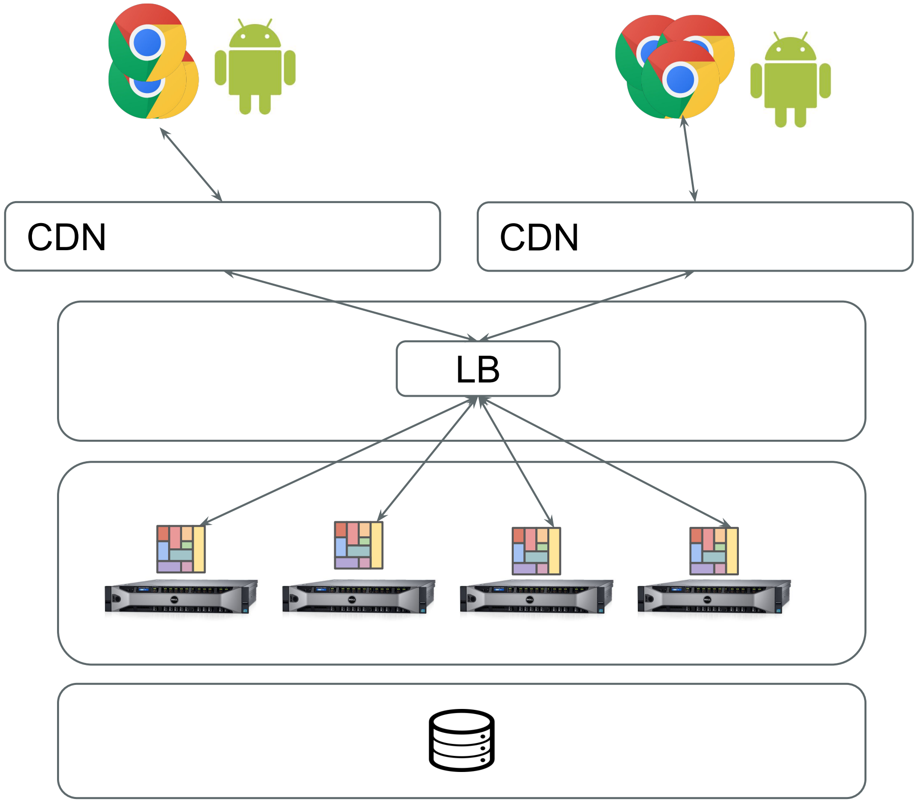
As soon as we ***scale horizontally*** we need to route requests and load balance



+ Autoscaling group

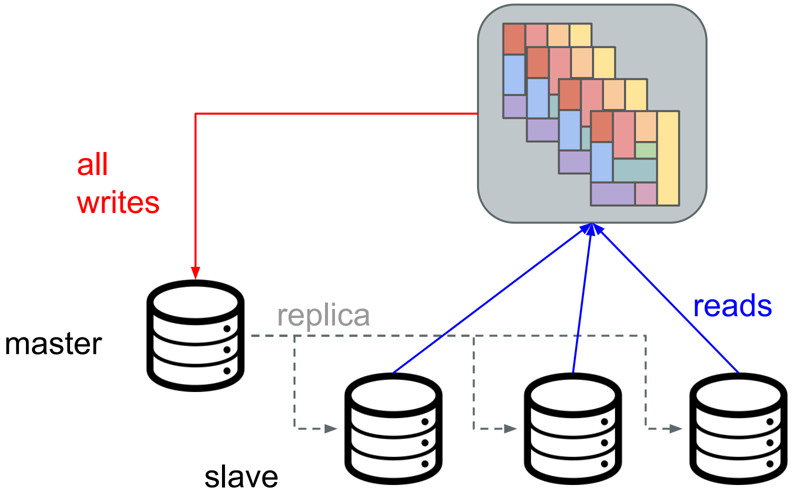
Tactic 4: CDN

(Cloudfront)



(ordering of tactics app specific - CDN is generally configuration and high% of content may be cacheable)

Tactic 5: Scaling Database Tier

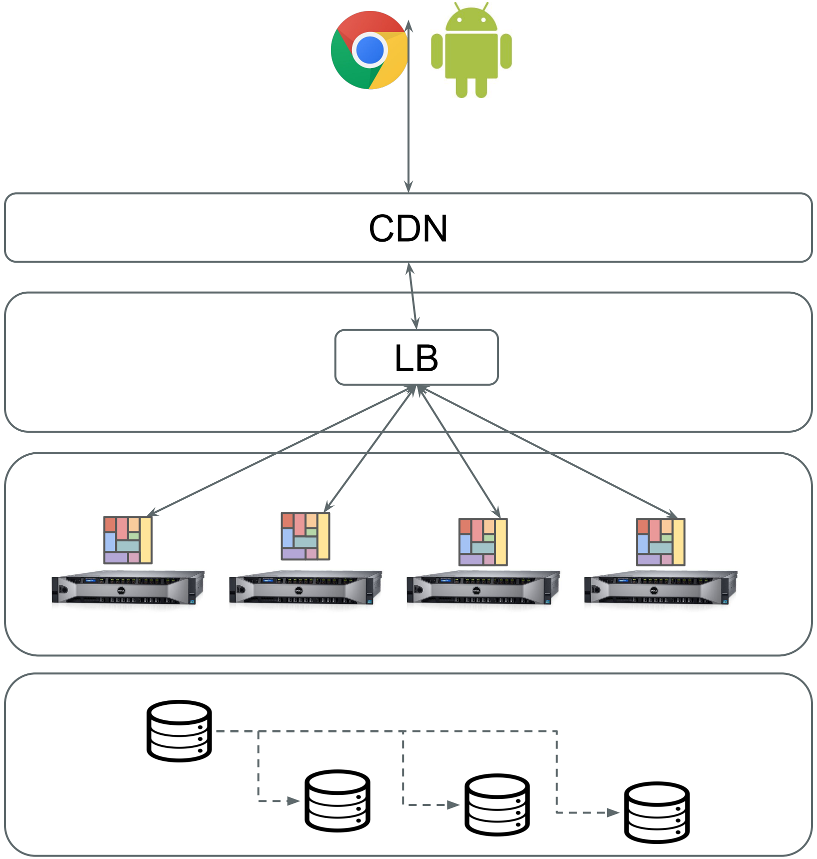


Scaling ***reads*** is easier than scaling ***writes*** (consistency, transactionality, distributed system)

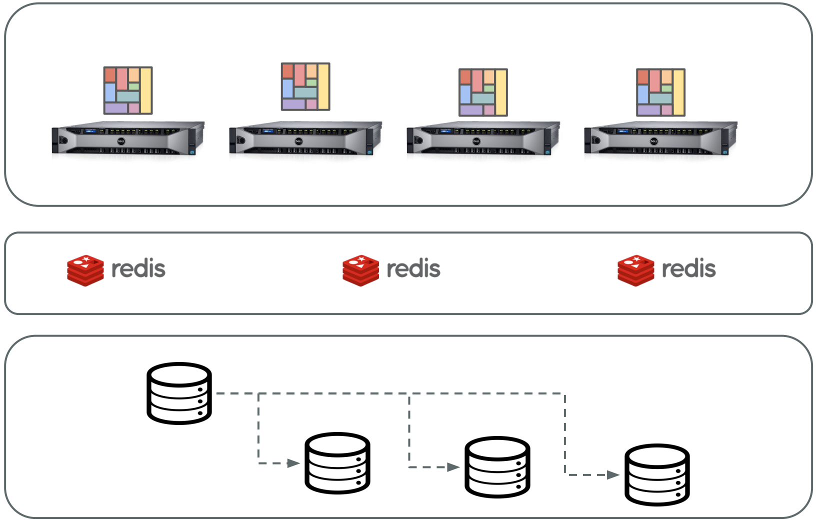
(We get hit by CAP theorem)

(one of many database scaling patterns: master-master, tree)

Interim State



Tactic 6: Cache

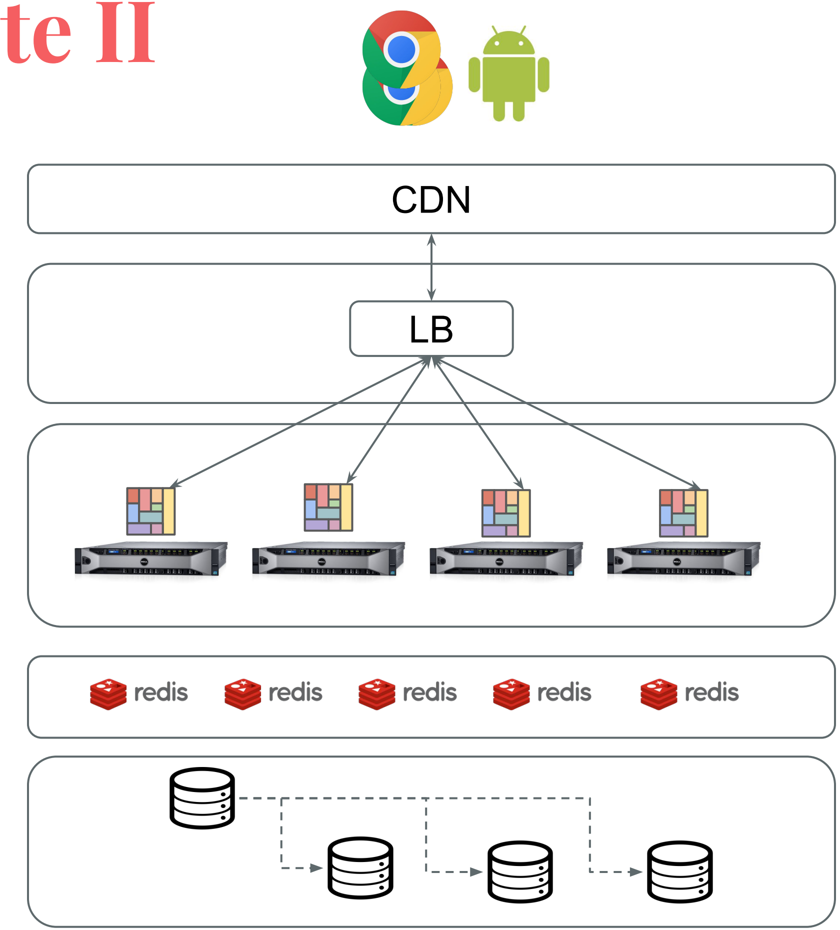


- Caching Strategies
- Caching Patterns and Two
  - Pattern: Write Through, Behind, Aside
  - Eviction: LRU, LFU, FIFO, etc
- Application level data structures can also be cached
- CDN is essentially a cache of a subset of assets

Complexity:

- Data : Cache mapping
- Maintaining logical integrity
- Need to consider rollouts ….

Interim State II



Success!!!

Khloe Kardashians mentioned your app on Oprah, tweeted it and then hashtagged it on instagram

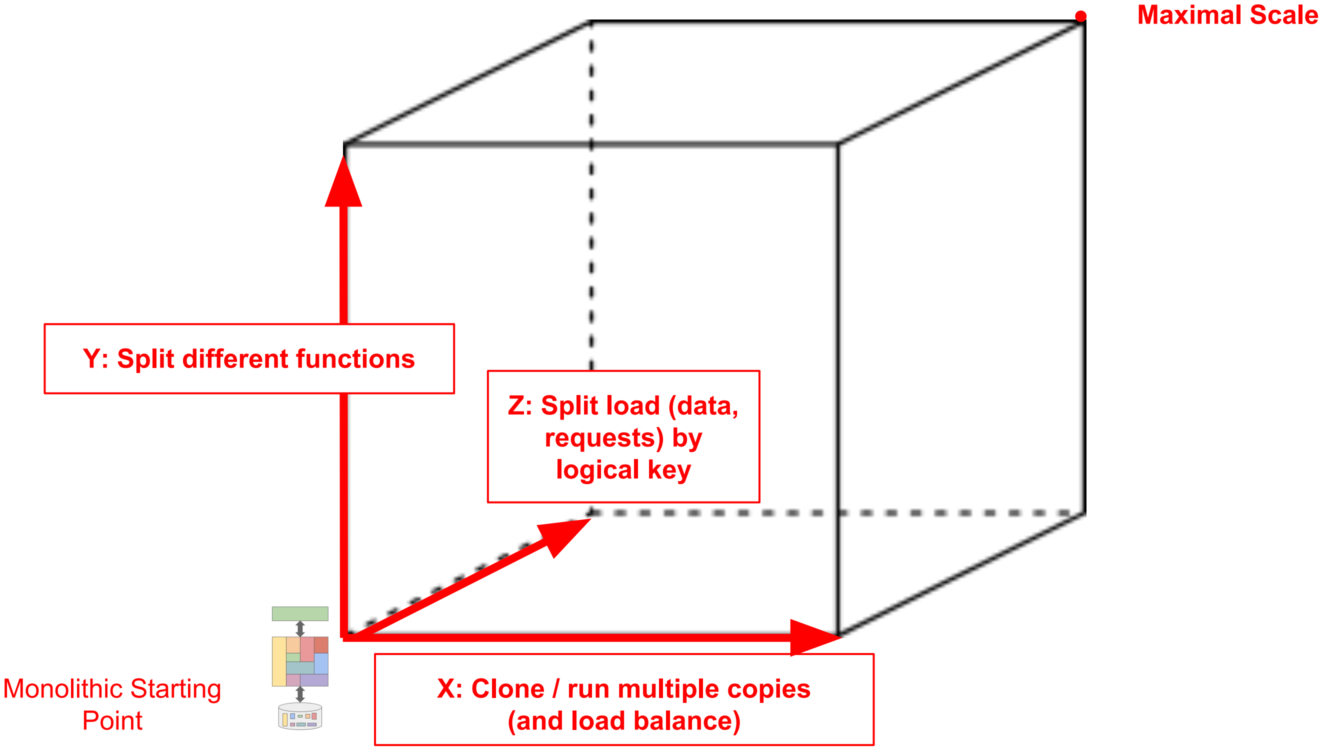
You have x1000 users. What now?

Scale Cube

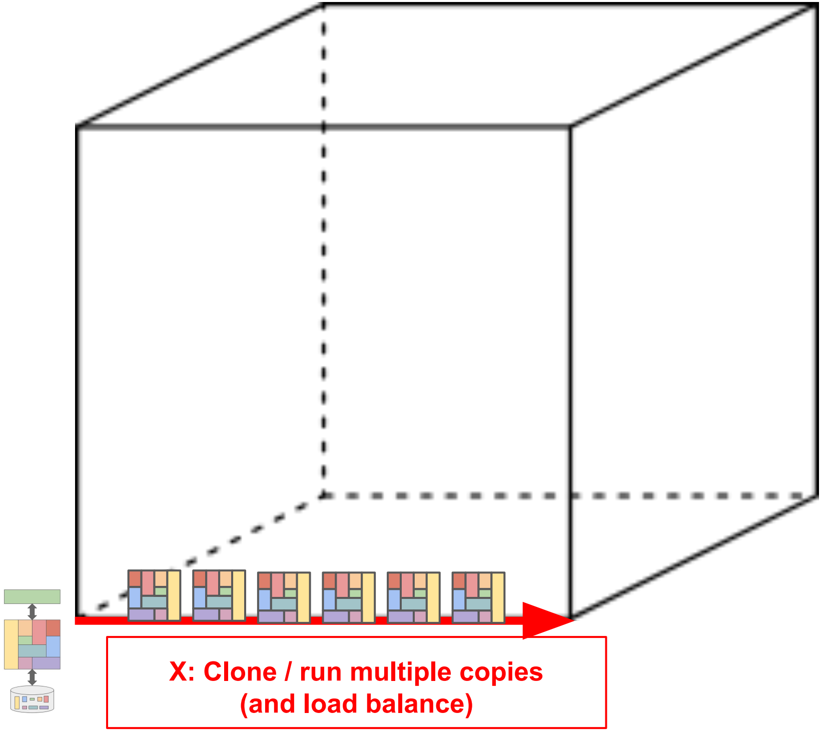
(we have already been doing one *dimension* of the scale cube but let’s introduce it now to give us more options)

Scale Cube Model

"*Starting* with a *monolithic* what are my options to scale?"


Maximal Scale
Y: Split different functions
Z: Split load (data, requests) by logical key
Monolithic Starting Point
X: Clone / run multiple copies (and load balance)

X-Scale

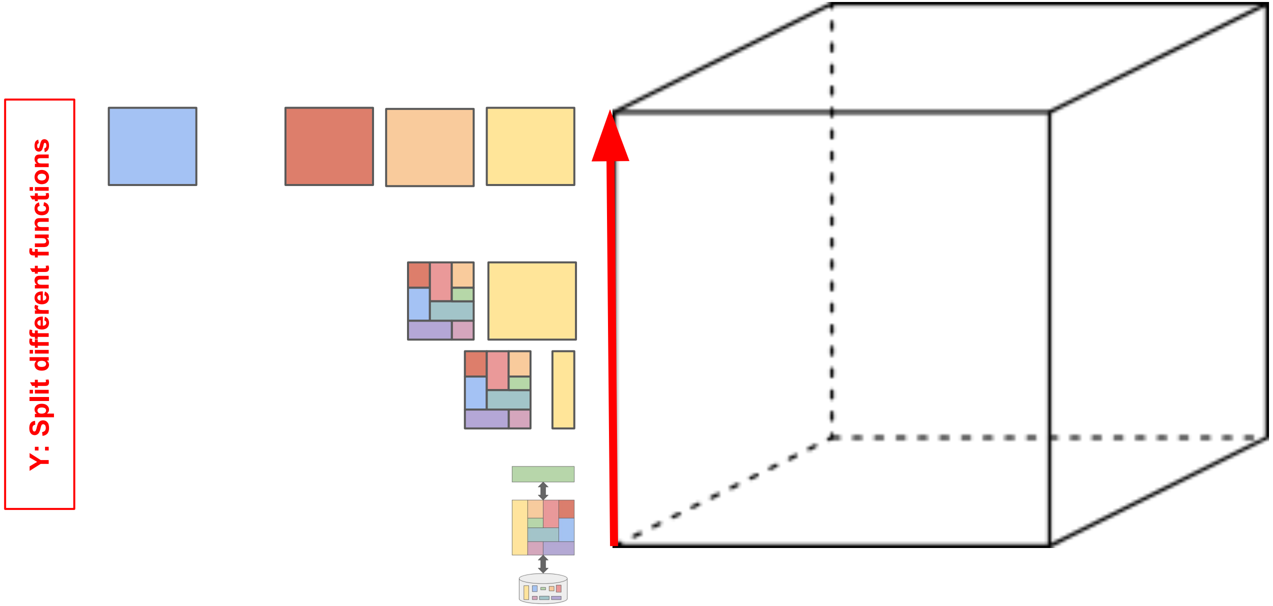


Monolithic Starting Point

X: Clone / run multiple copies (and load balance)

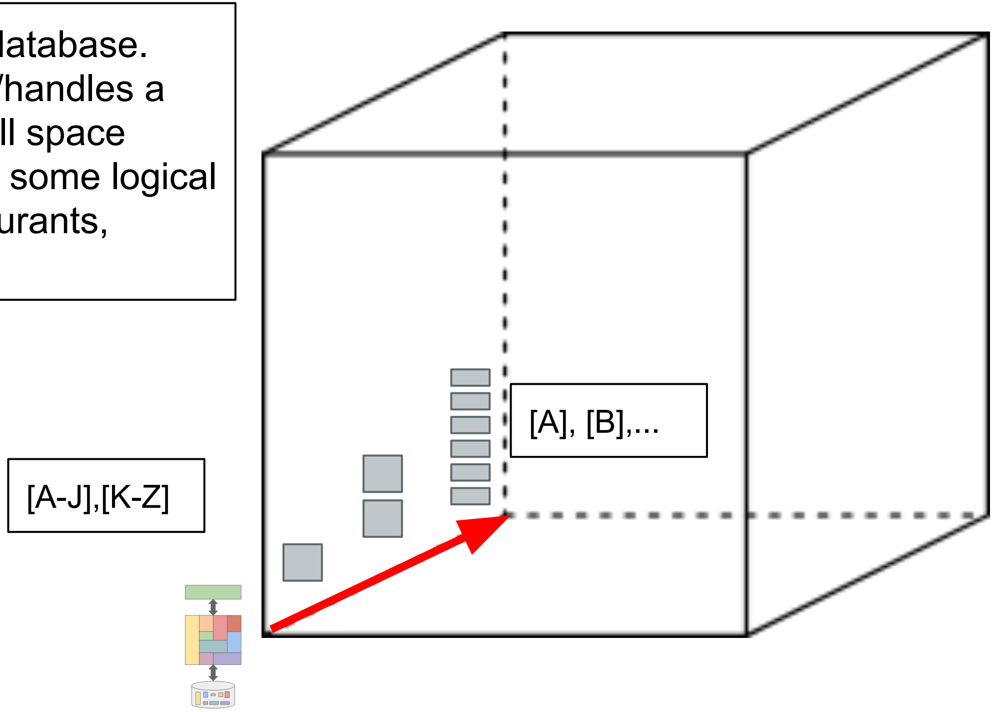
Y-Scale

Typically applies to database. Each instance holds/handles a partition of the overall space (partition by users or some logical key of domain, restaurants, geography, etc)



Z-Scale

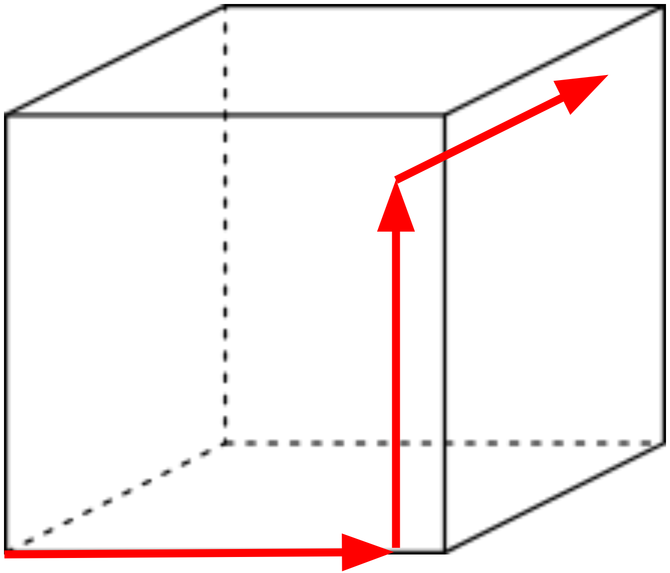
Typically applies to database. Each instance holds/handles a partition of the overall space (partition by users or some logical key of domain, restaurants, geography, etc)

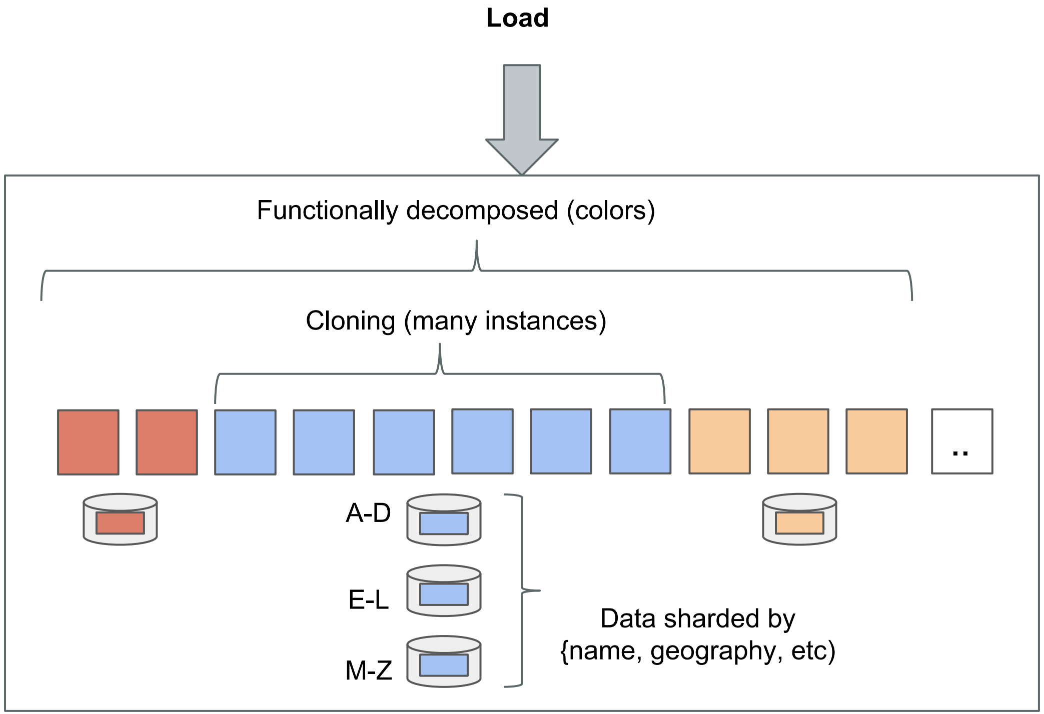


[A-J],[K-Z]

[A], [B],...

All Three





Load

Functionally decomposed (colors)

Cloning (many instances)

..

A-D

E-L

M-Z

Data sharded by {name, geography, etc)

(One hypothetical set of scaling options)

Summary

- Scalability is a Quality Attribute of a system
- Understand Q.A. and design for required scalability
- Vertical and Horizontal Scalability Options
- Well established scalability patterns for web-apps
- Scale Cube is a further model with three dimensions
  - Clone
  - Functionally Decompose
  - Partition/Shard
- All designs have pros/cons / considerations

Pop Quiz

Question	Answer
Scalability is the latency of response to a user {T/F}?	
What is the relationship between writing better performing code and scalability?	
It is possible to always scale by buying bigger, more performant servers?	
The difference between horizontal scaling and vertical is ….?	
The three dimensions of the scale cube are {}?	
Where is vertical scaling on the scale cube?	

Reading

Reading	Optionality
Availability Patterns	Recommended
Scalability Rules: Principles for Scaling Web Sites, Second Edition	Chap 2,3,6
Microservices (recap of arch style + now to connect w/ scalability)	Required
Hug of Death	Optional
CAP	Optional*
Building Scalable Websites [Cal Henderson]	Optional/FYI

* Not part of this course. However, important to understand that as soon as you scale/architect across two nodes then you fulfill the criteria of a distributed system -> are hit by CAP -> are in a world of fun trade-offs

Scale Cubing a Burger Joint

A small burger joint has

- 1 chef
- To prepare a burger (chef does it all)
  - Materials mixed (2 min)
  - Formed into patties (1 min)
  - Grill (10 min)
- Restaurant currently has 10 tables

If the restaurant scales from 10,20,50,100 tables how would you scale this with reference to the scale cube?

Traditional Definitions

(I dislike these definitions as they are either ‘static’/slow moving view of scalability, opportunities provided by cloud (up and down) or they do not enumerate the full set of constraints - $, QoS, admin)

“Scalability is the ability of the system to cope with increasing numbers of users without reducing the overall QoS that is delivered to any user.” [Somerville]

“A system is said to be scalable if it can handle the addition of users and resources without suffering a noticeable loss of performance or increase in administrative complexity.” [Neumann]