

← Scaling →

# Pinterest



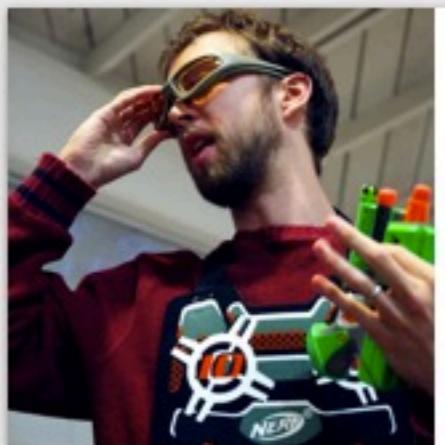
Marty Weiner  
Orodruin, Mordor



Yashh Nelapati  
The Shire

**Pinterest is . . .**

An online pinboard to organize and  
share what inspires you.



# Marty Weiner

Engineer at Pinterest by day. He-Man, Master of the Universe, by night, sworn to defeat the evil forces of Skeletor with Battlecat ([pinterest.com/yashh](http://pinterest.com/yashh)).. Any questions? Email away! ...

   Grayskull, Eternia

## Repins from



Justin Edmund



Candice Weiner



Enid Hwang

44 Boards

581 Pins

51 Likes

Activity

Edit Profile



1739 followers

55 following

## 3D Models

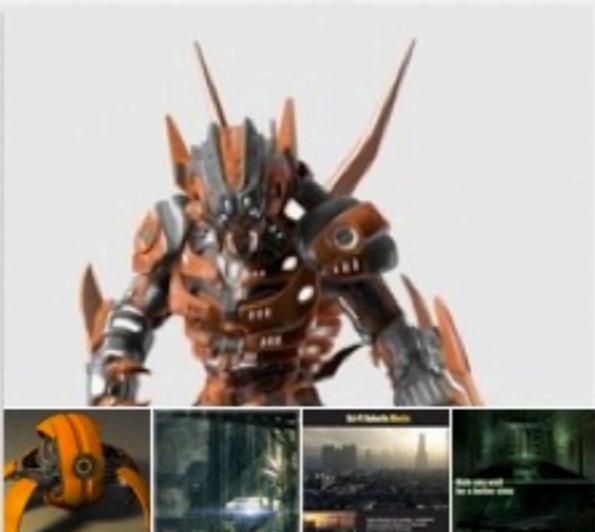
28 pins



Edit

## 3D Models - Future

27 pins



Edit

## 3D Models - Toon

16 pins



Edit

## 3D Models - Fantasy

10 pins



River Boat

Edit

## 3D Models - Buildings

9 pins



## 3D Models - Space

14 pins



## 3D Models - Low Poly

8 pins



## 3D Models - Past

7 pins



Inspiration

47 pins



Edit



Marty Weiner

Uploaded 6 days ago

Repin

Edit

Uploaded by user

Like

Tweet

Embed

Report Pin

Email



## Friends to Follow

**See All**

- |  |               |
|--|---------------|
|  Gene Warren      | <b>Follow</b> |
|  Jessica Spurling | <b>Follow</b> |
|  angela           | <b>Follow</b> |

## Recent Activity

- |   |
|---|
|  Hemanth Pai repinned your pin.<br>1 hour ago                            |
|  Luis Madrigal and 1 other are now following your pins.<br>9 hours ago |
|  myong greenspan and 2 others liked your pin.<br>9 hours ago           |
|  Phyllis Weiner repinned your pin.<br>13 hours ago                     |
|  Phyllis Weiner liked your pin.<br>13 hours ago                        |
|  Phyllis Weiner repinned your pin.<br>14 hours ago                     |
|  Phyllis Weiner liked your pin.<br>14 hours ago                        |
|  Phyllis Weiner repinned your pin.<br>14 hours ago                     |
|  Phyllis Weiner liked your pin.  |



Pizza Chopper

 Matt Jones via Cynthia Maxwell onto General Cooking

Delicious recipe to make Cinnabon

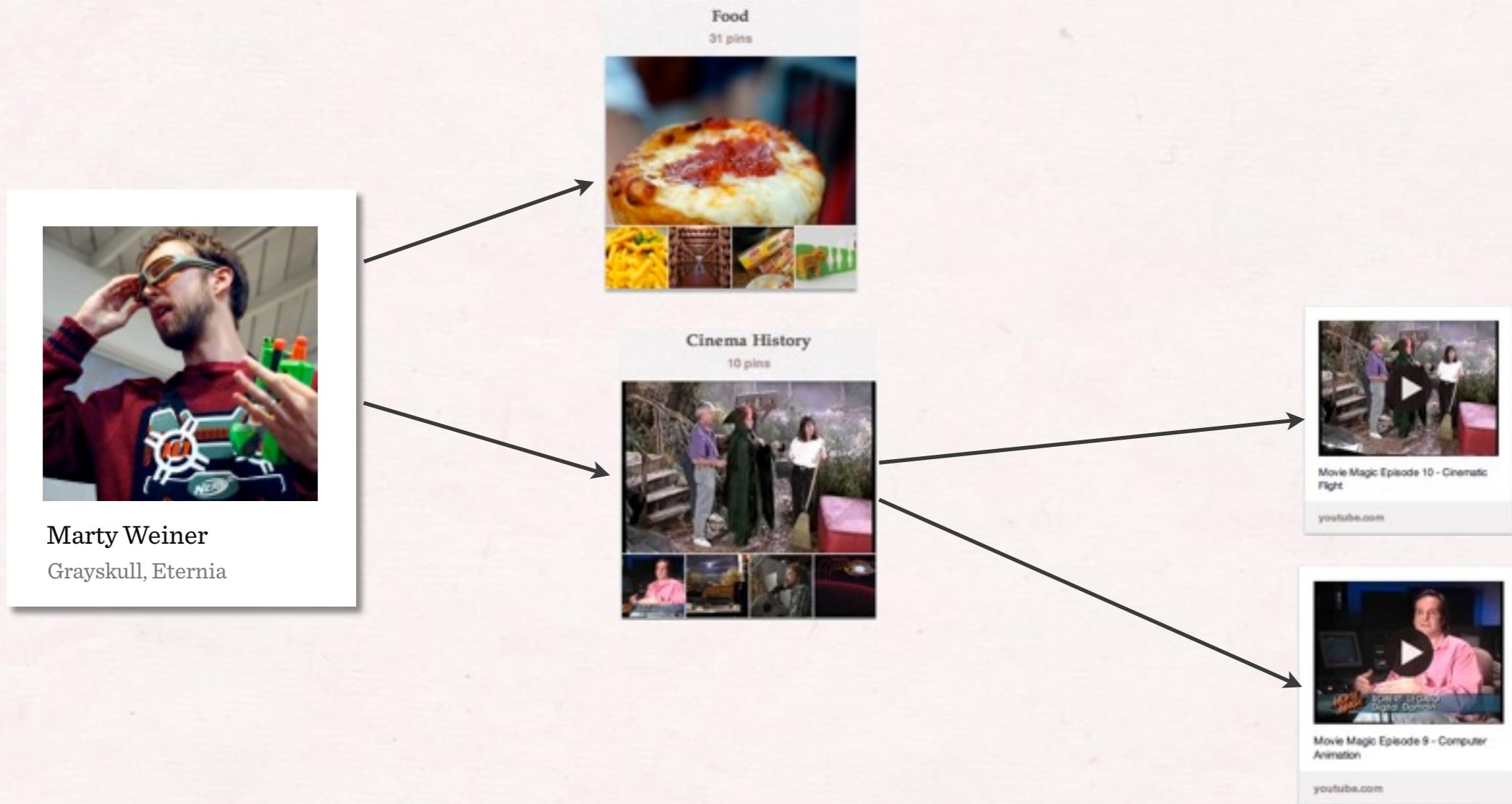
1 comment

 Susan Peck via Lauren onto Baking Susan Peck This looks delicious, but it's taking me to an add. I'm confused right now. Susan Peck via Ashley FitzSimmons onto clothing

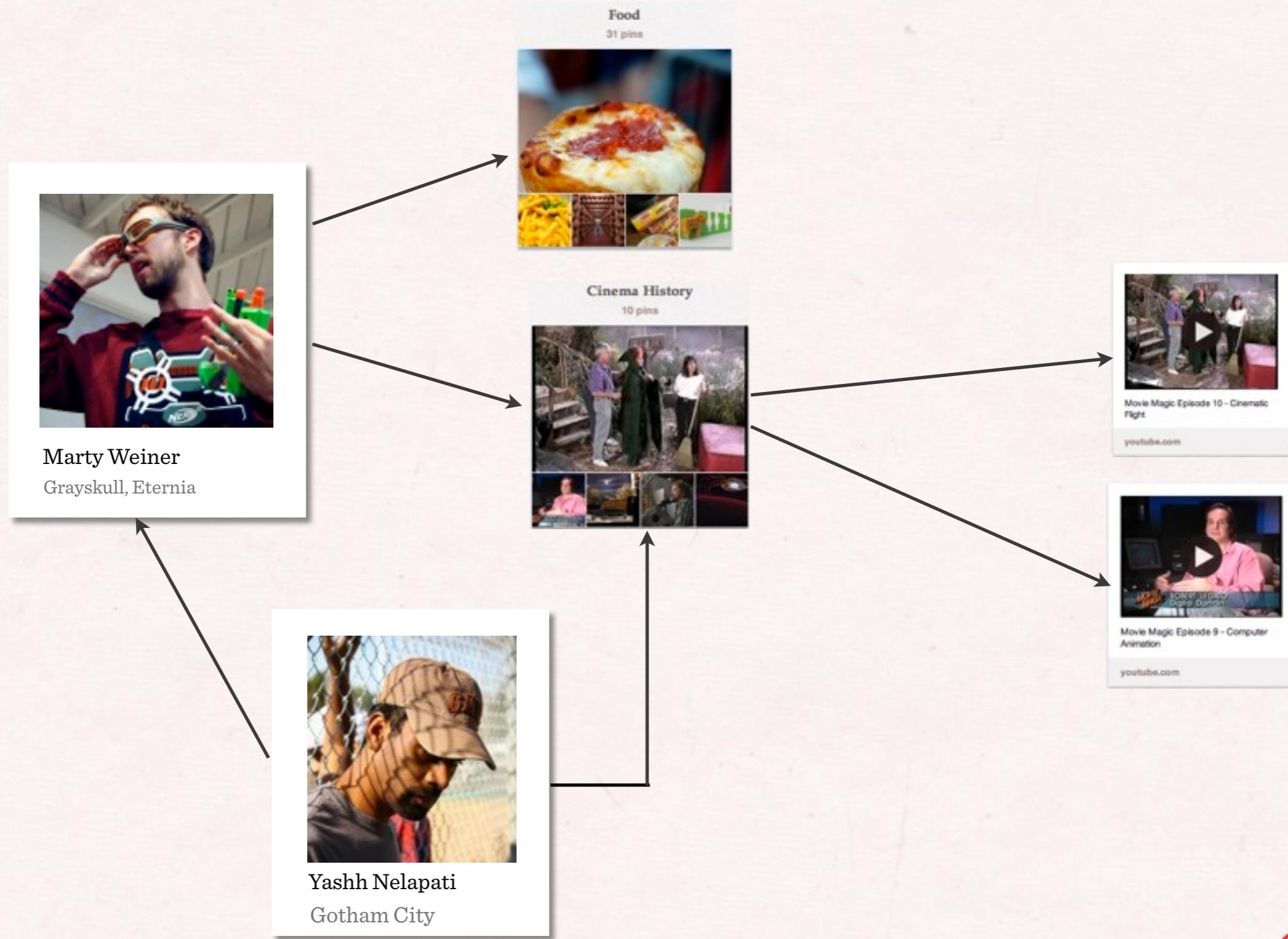
Ah this looks fun, I have to try this!

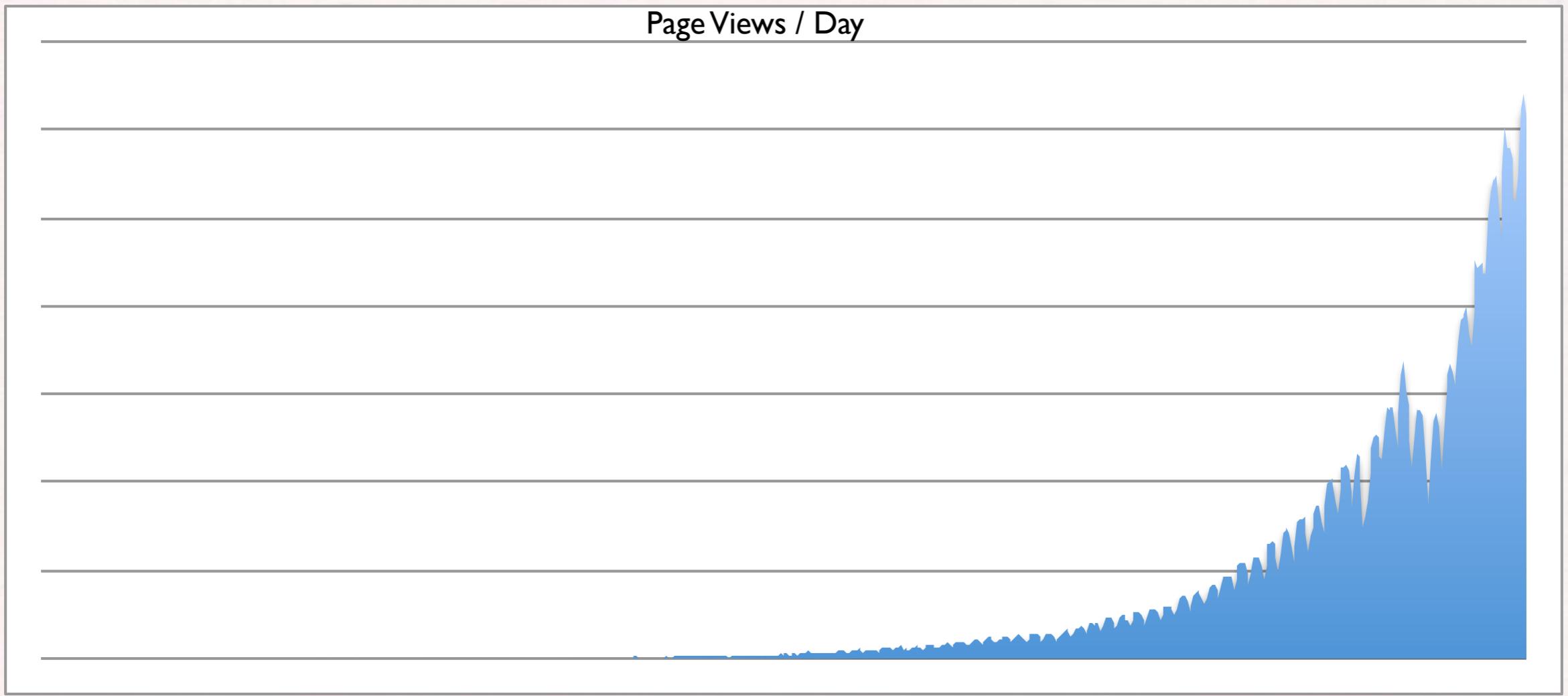
 Susan Peck via Melissa Guffy onto Totally

# Relationships



# Relationships





Mar 2010



Jan 2011

Jan 2012

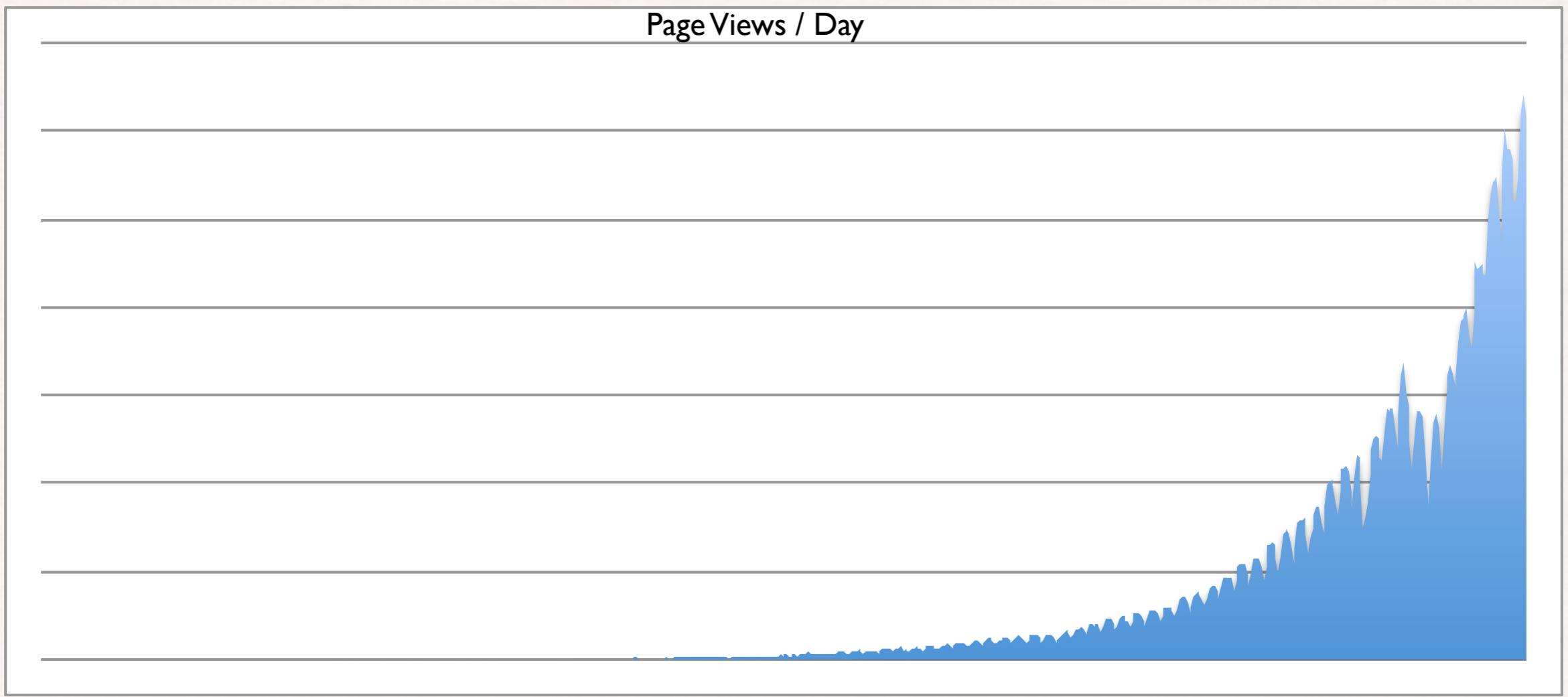


Mar 2010



Jan 2011

Jan 2012

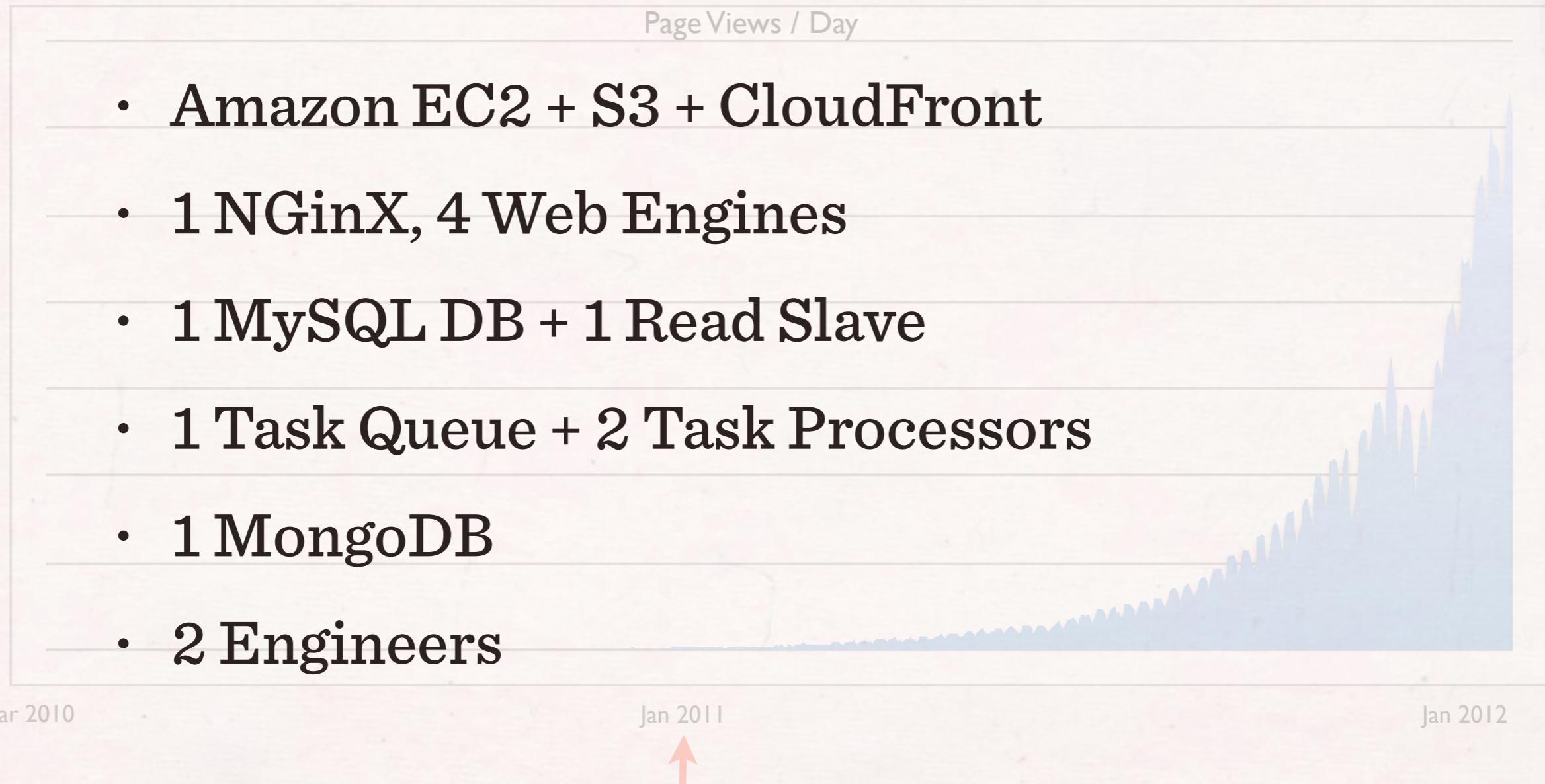


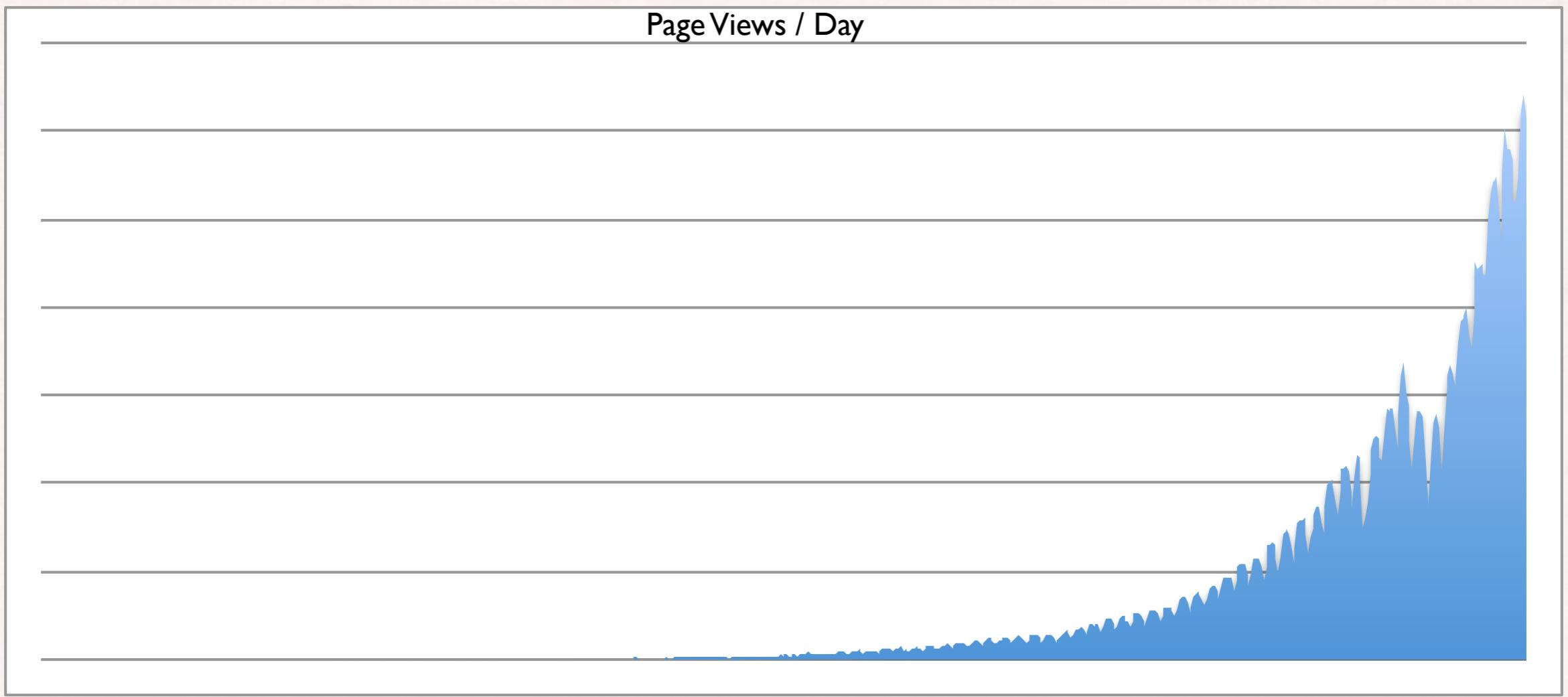
Mar 2010

Jan 2011

Jan 2012







Mar 2010

Jan 2011

Jan 2012



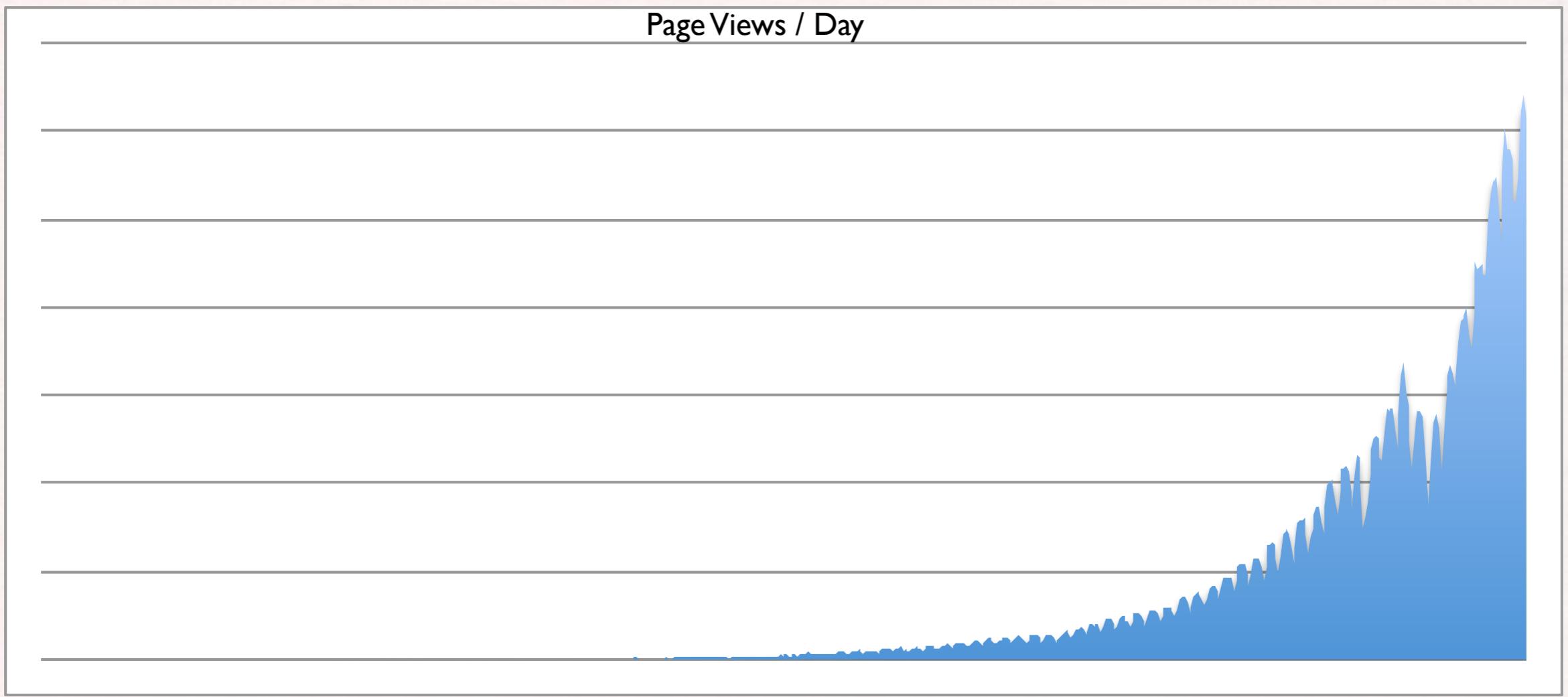
- Amazon EC2 + S3 + CloudFront
- 2 NGinX, 16 Web Engines + 2 API Engines
- 5 Functionally Sharded MySQL DB + 9 read slaves
- 4 Cassandra Nodes
- 15 Membase Nodes (3 separate clusters)
- 8 Memcache Nodes
- 10 Redis Nodes
- 3 Task Routers + 4 Task Processors

Mar 2010      Jan 2011      Jan 2012

- 4 Elastic Search Nodes
- 3 Mongo Clusters
- 3 Engineers

# **Lesson Learned #1**

## **It will fail. Keep it simple.**



Mar 2010

Jan 2011

Jan 2012



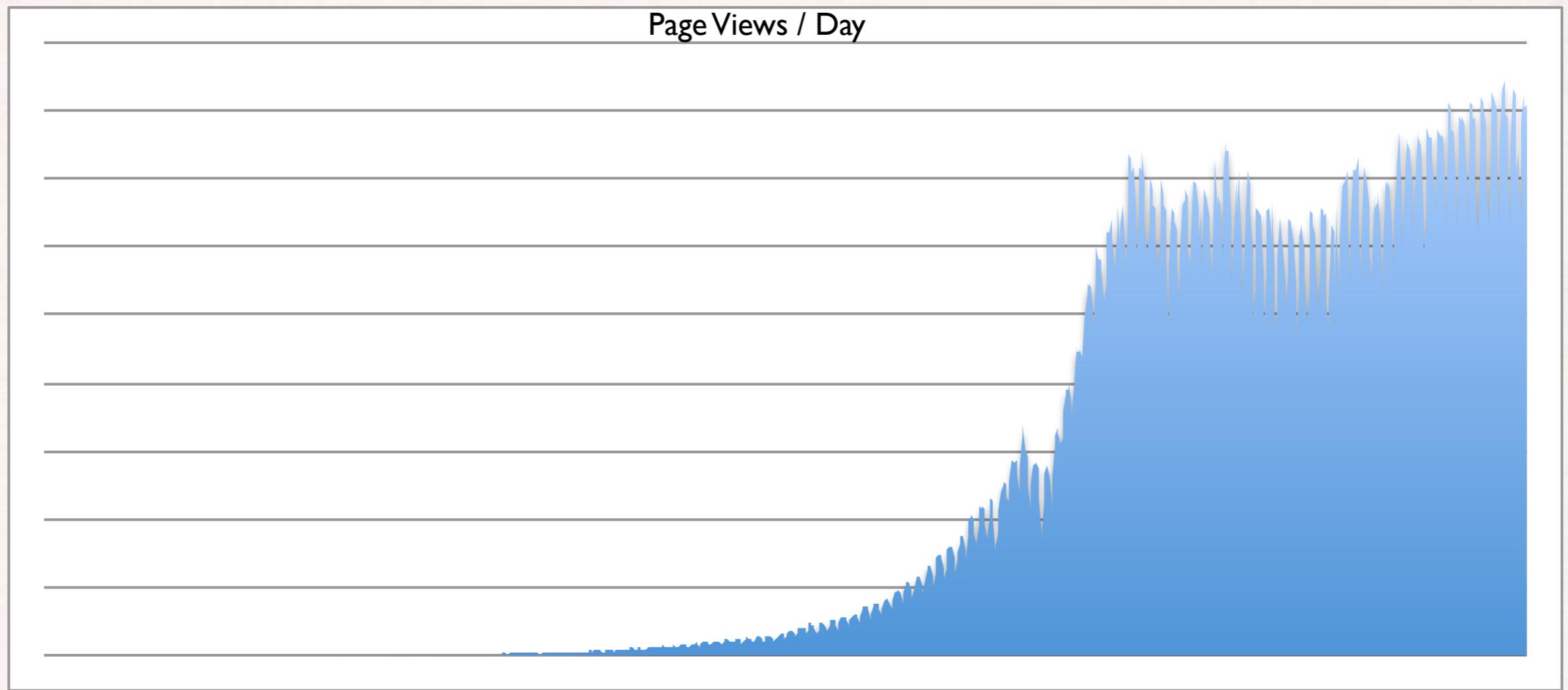
- Amazon EC2 + S3 + Akamai, ELB
- 90 Web Engines + 50 API Engines
- 66 MySQL DBs (m1.xlarge) + 1 slave each
- 59 Redis Instances
- 51 Memcache Instances
- 1 Redis Task Manager + 25 Task Processors
- Sharded Solr
- 6 Engineers

Mar 2010

Jan 2011

Jan 2012





Mar 2010

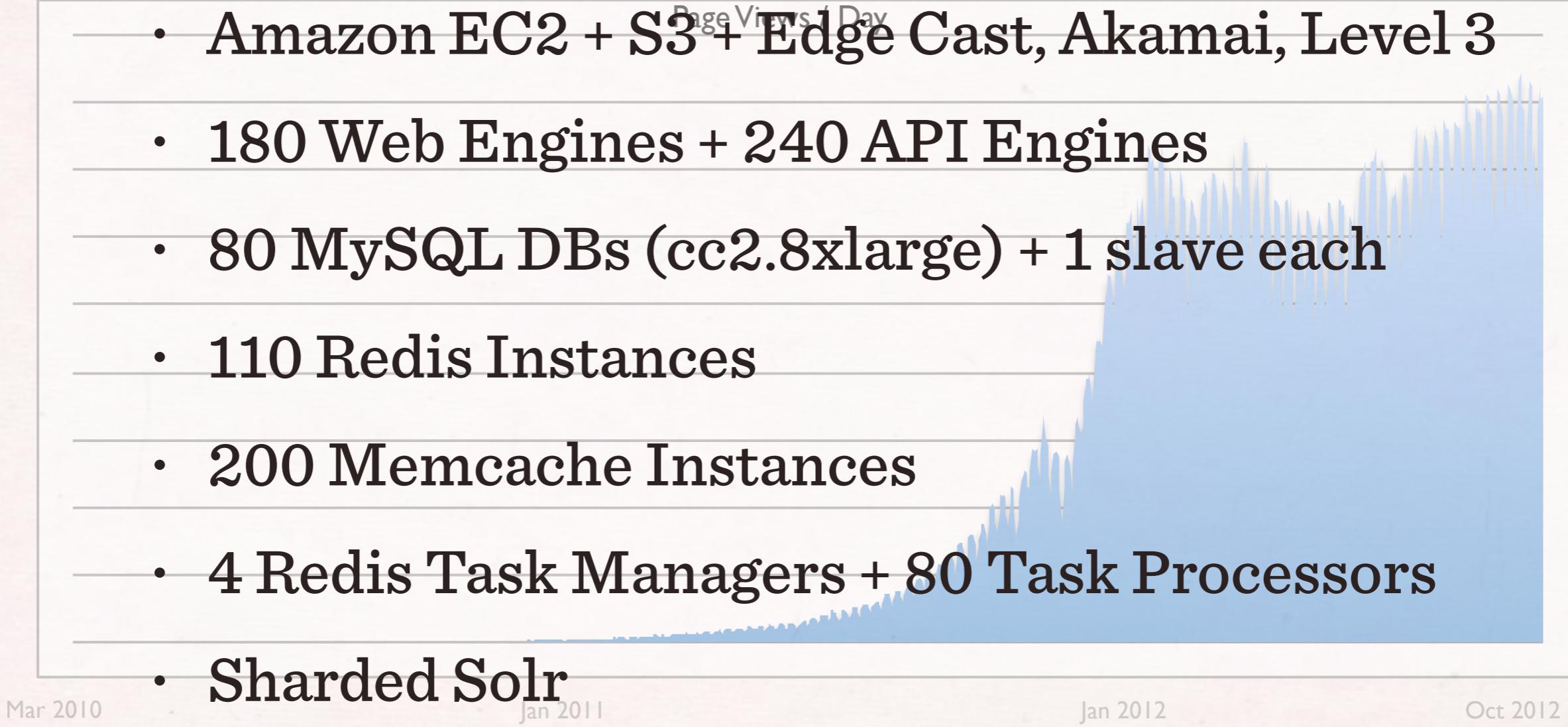
Jan 2011

Jan 2012

Oct 2012



- Amazon EC2 + S3 + Edge Cast, Akamai, Level 3
- 180 Web Engines + 240 API Engines
- 80 MySQL DBs (cc2.8xlarge) + 1 slave each
- 110 Redis Instances
- 200 Memcache Instances
- 4 Redis Task Managers + 80 Task Processors
- Sharded Solr
- 40 Engineers



# Why Amazon EC2/S3?

- Very good reliability, reporting, and support
- Very good peripherals, such as managed cache, DB, load balancing, DNS, map reduce, and more...
- *New instances ready in seconds*

# Why Amazon EC2/S3?

- Very good reliability, reporting, and support
- Very good peripherals, such as managed cache, DB, load balancing, DNS, map reduce, and more...
- *New instances ready in seconds*
- Con: Limited choice

# Why Amazon EC2/S3?

- Very good reliability, reporting, and support
- Very good peripherals, such as managed cache, DB, load balancing, DNS, map reduce, and more...
- *New instances ready in seconds*
- Con: Limited choice
- Pro: Limited choice

# Why MySQL?

- Extremely mature
- Well known and well liked
- Rarely catastrophic loss of data
- Response time to request rate increases linearly
- Very good software support - XtraBackup, Innotop, Maatkit
- Solid active community
- Very good support from Percona
- Free

# Why Memcache?

- Extremely mature
- Very good performance
- Well known and well liked
- Never crashes, and few failure modes
- Free

# Why Redis?

- Variety of convenient **data structures**
- Has persistence and replication
- Well known and well liked
- Consistently good performance
- Few failure modes
- Free

# **Clustering**

**vs**

# **Sharding**

## Clustering



- Data distributed automatically
- Data can move
- Rebalances to distribute capacity
- Nodes communicate with each other

## Sharding

# Clustering



- Data distributed manually
- Data does not move
- Split data to distribute load
- Nodes are not aware of each other

# Sharding



# Why Clustering?

- Examples: Cassandra, MemBase, HBase
- Automatically scale your datastore
- Easy to set up
- Spatially distribute and colocate your data
- High availability
- Load balancing
- No single point of failure

# What could possibly go wrong?

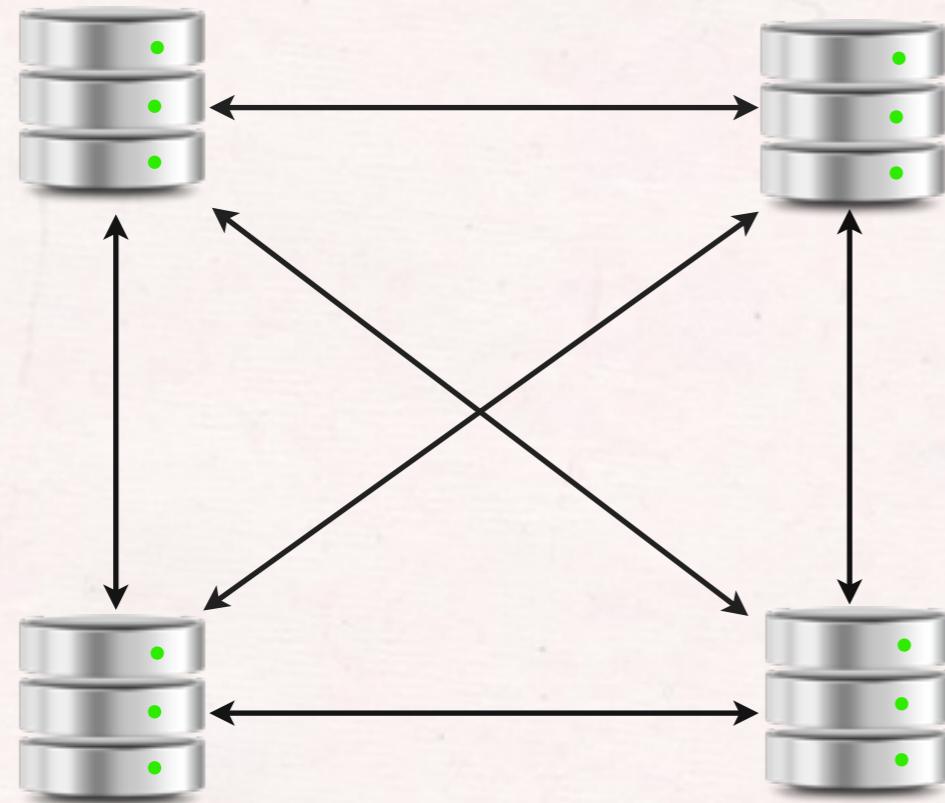


source: thereifixedit.com

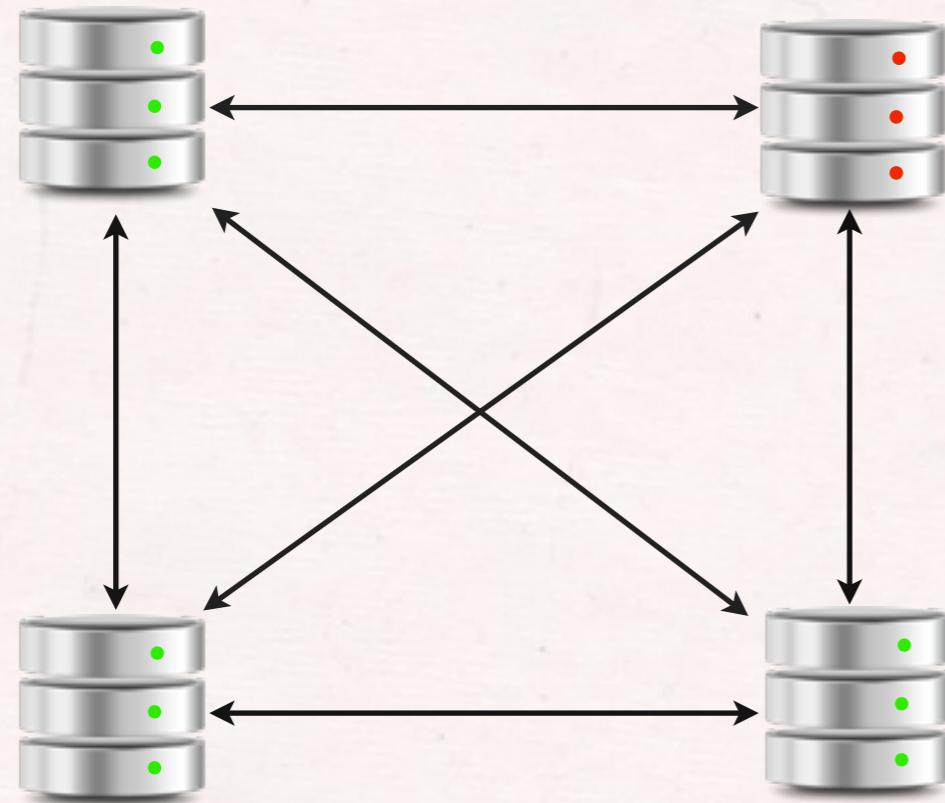
# Why Not Clustering?

- Still fairly young
- Fundamentally complicated
- Less community support
- Fewer engineers with working knowledge
- Difficult and scary upgrade mechanisms
- And, yes, there is a single point of failure. A BIG one.

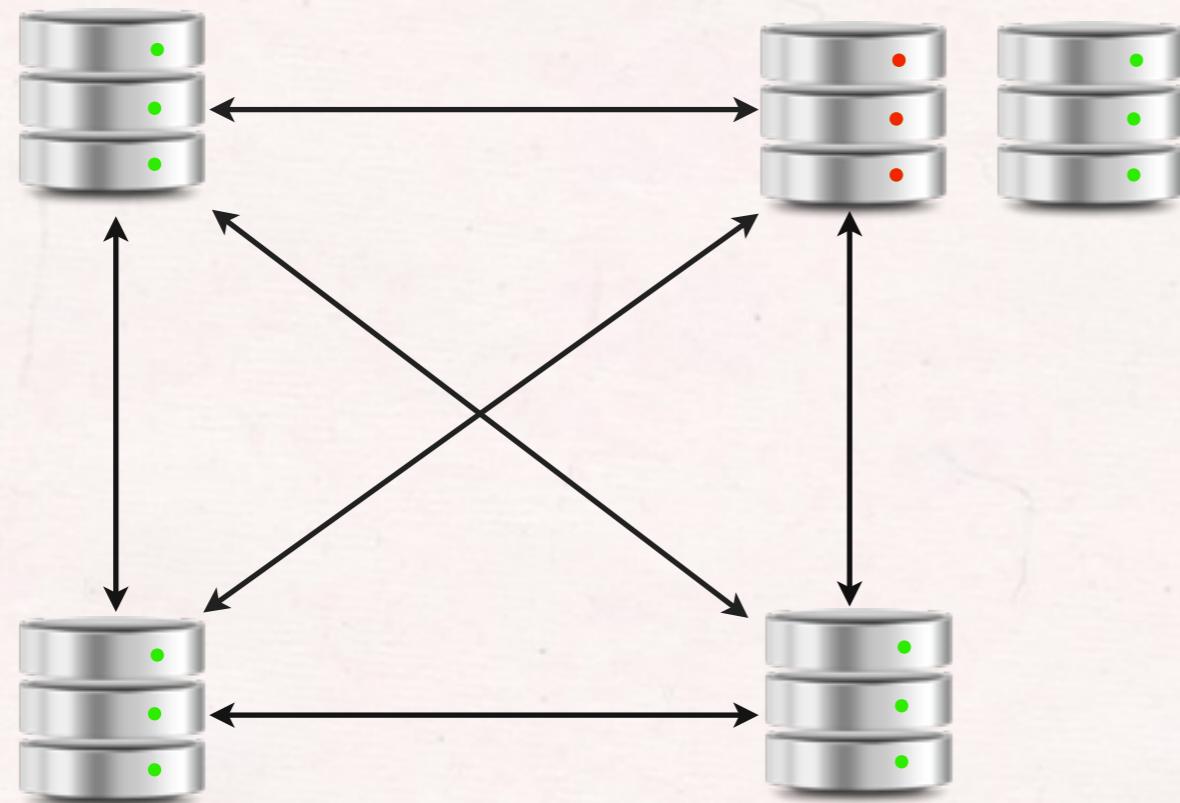
# Clustering Single Point of Failure



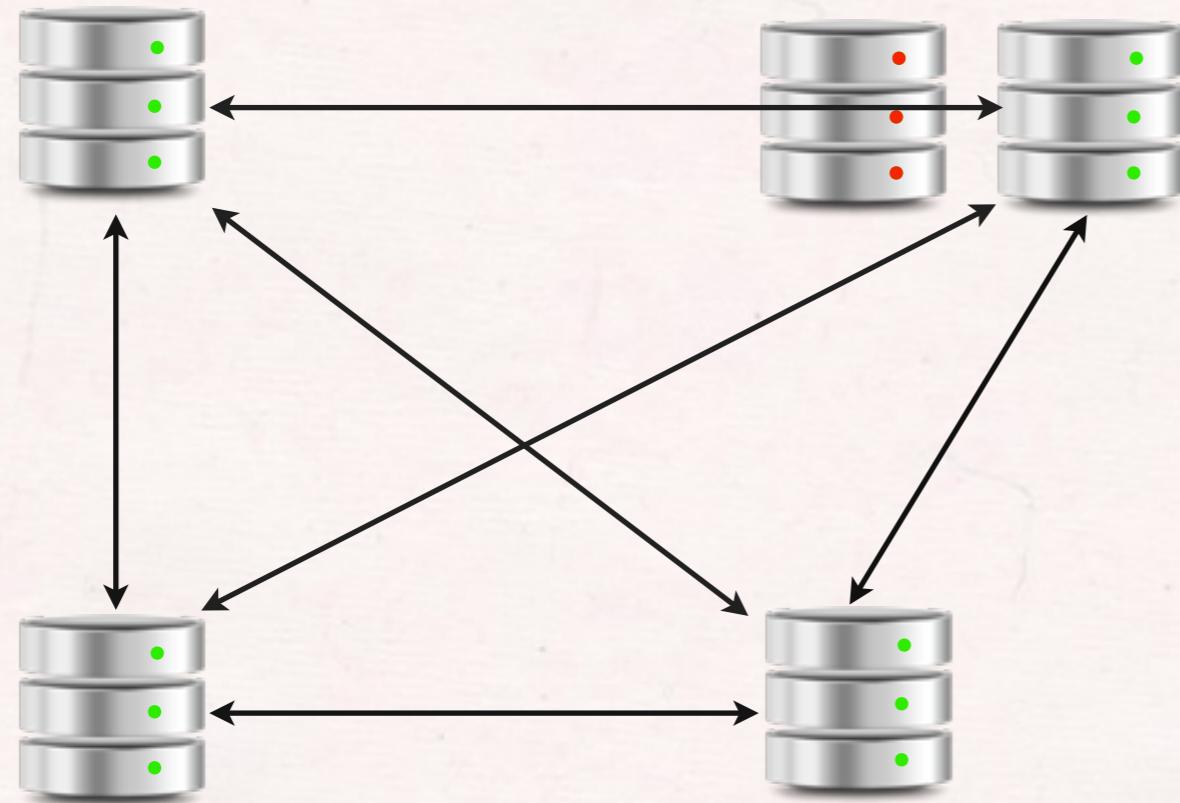
# Clustering Single Point of Failure



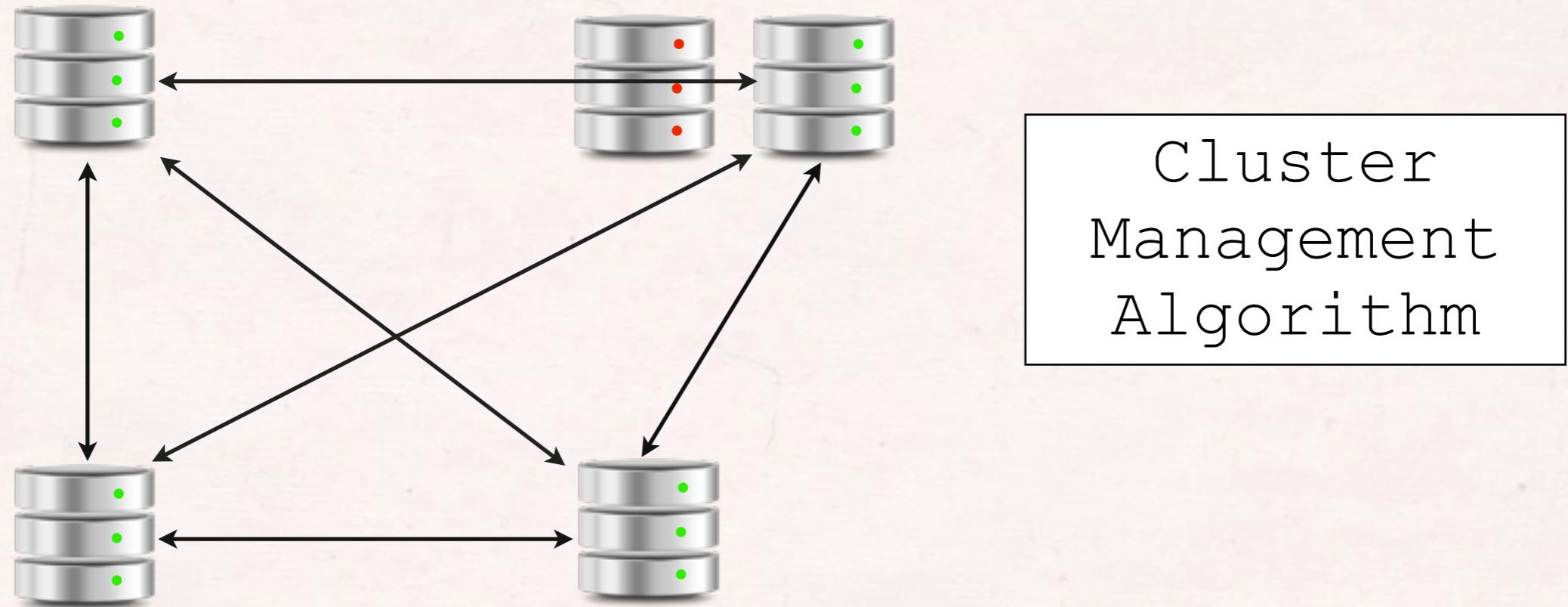
# Clustering Single Point of Failure



# Clustering Single Point of Failure



# Clustering Single Point of Failure



# Cluster Manager

- Same complex code replicated over all nodes
- Failure modes:
  - Data rebalance breaks
  - Data corruption across all nodes
  - Improper balancing that cannot be fixed (easily)
  - Data authority failure

# **Lesson Learned #2**

## Clustering is scary.

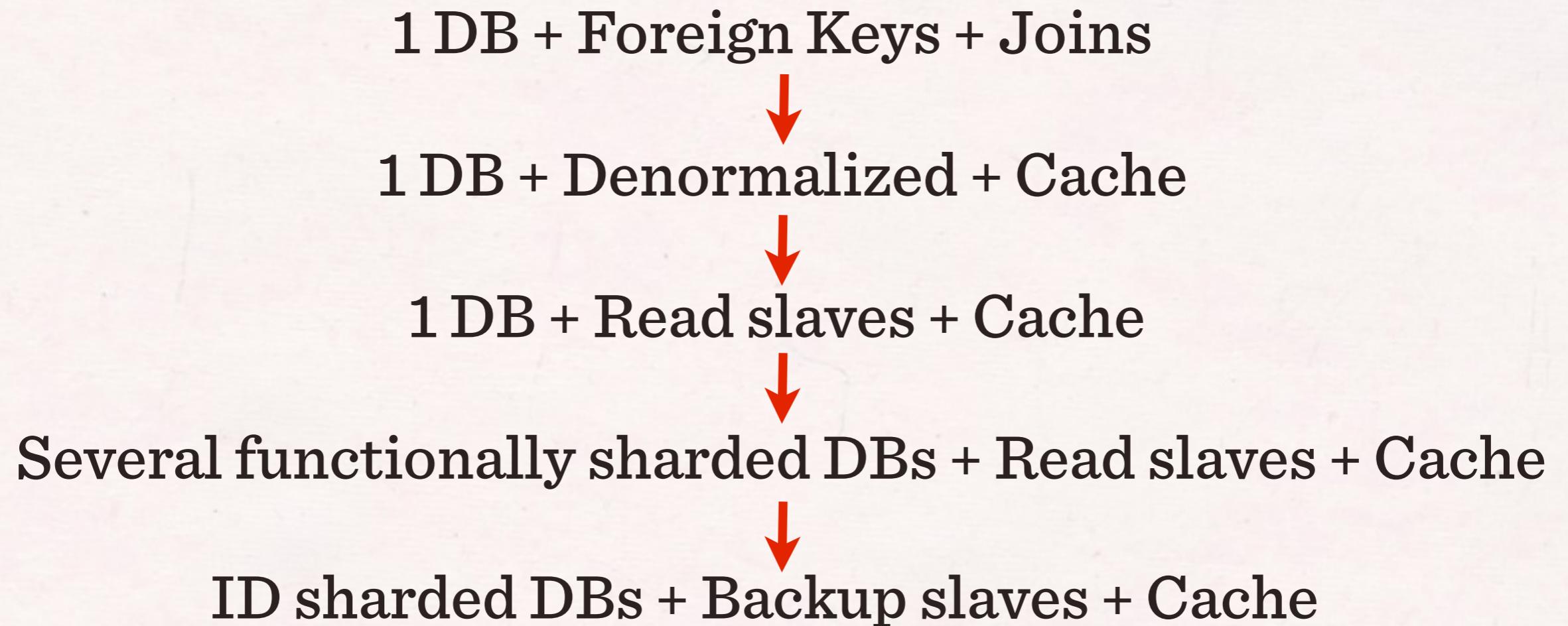
# Why Sharding?

- Can split your databases to add more capacity
- Spatially distribute and colocate your data
- High availability
- Load balancing
- Algorithm for placing data is very simple
- ID generation is simplistic

# When to shard?

- Sharding makes schema design harder
- Waiting too long makes the transition harder
- Solidify site design and backend architecture
- Remove all joins and complex queries, add cache
- Functionally shard as much as possible
- Still growing? Shard.

# Our Transition

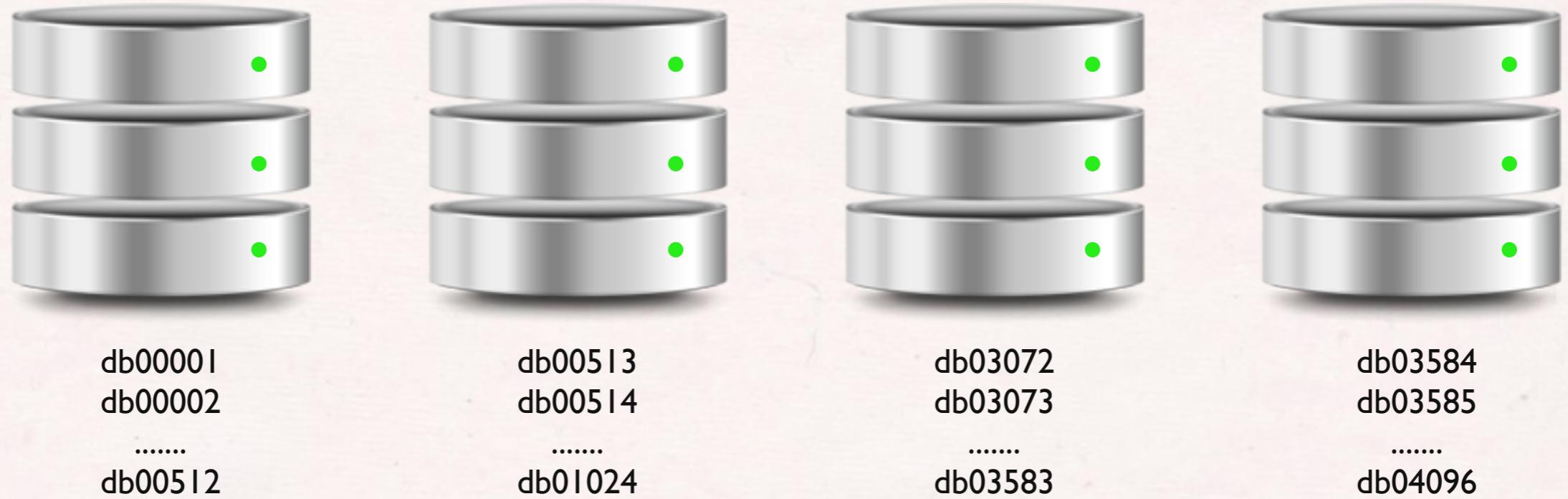


# Watch out for...

- Cannot perform most JOINS
- No transaction capabilities
- Extra effort to maintain unique constraints
- Schema changes requires more planning
- Reports require running same query on all shards

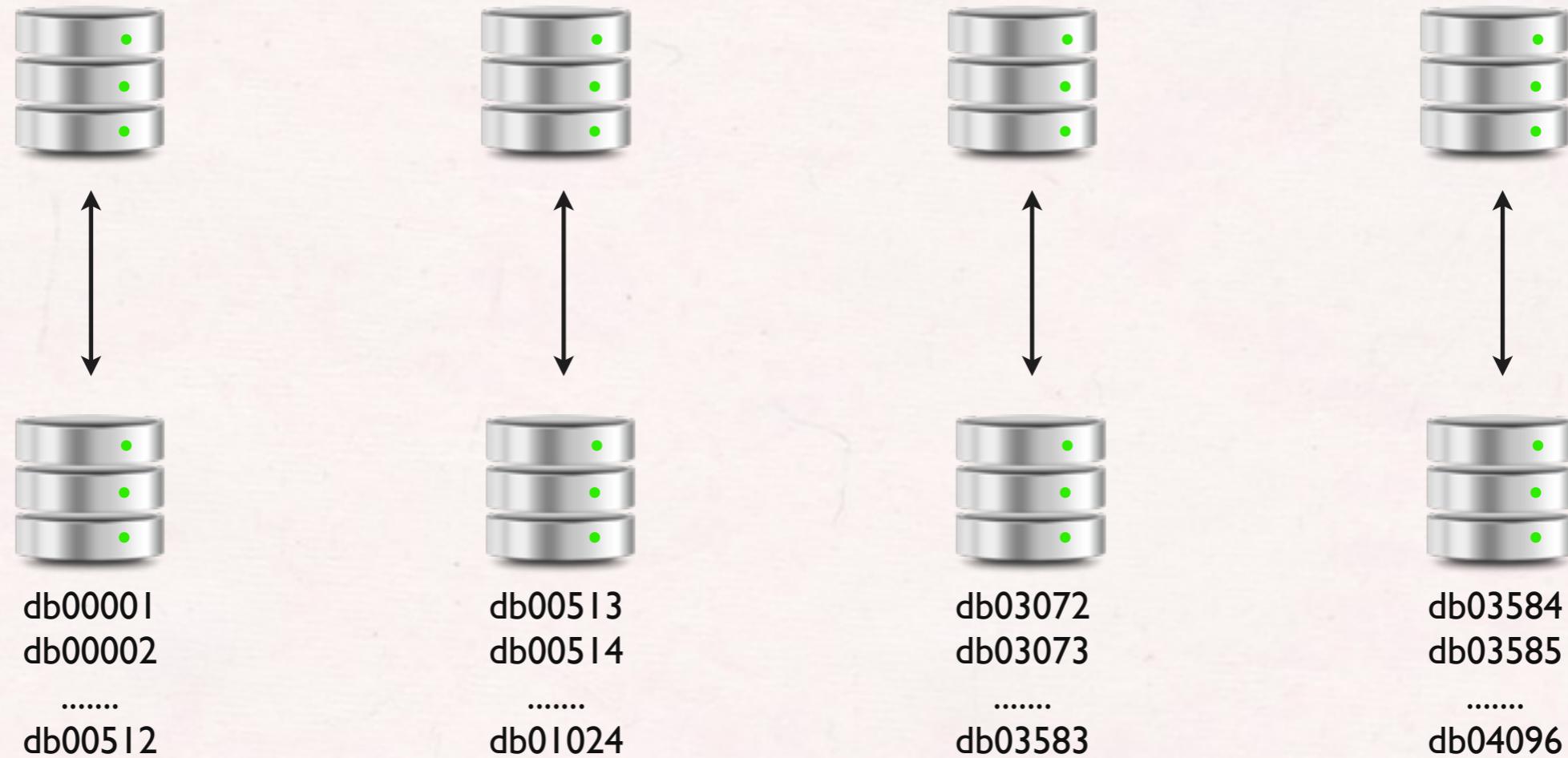
# How we sharded

# Sharded Server Topology



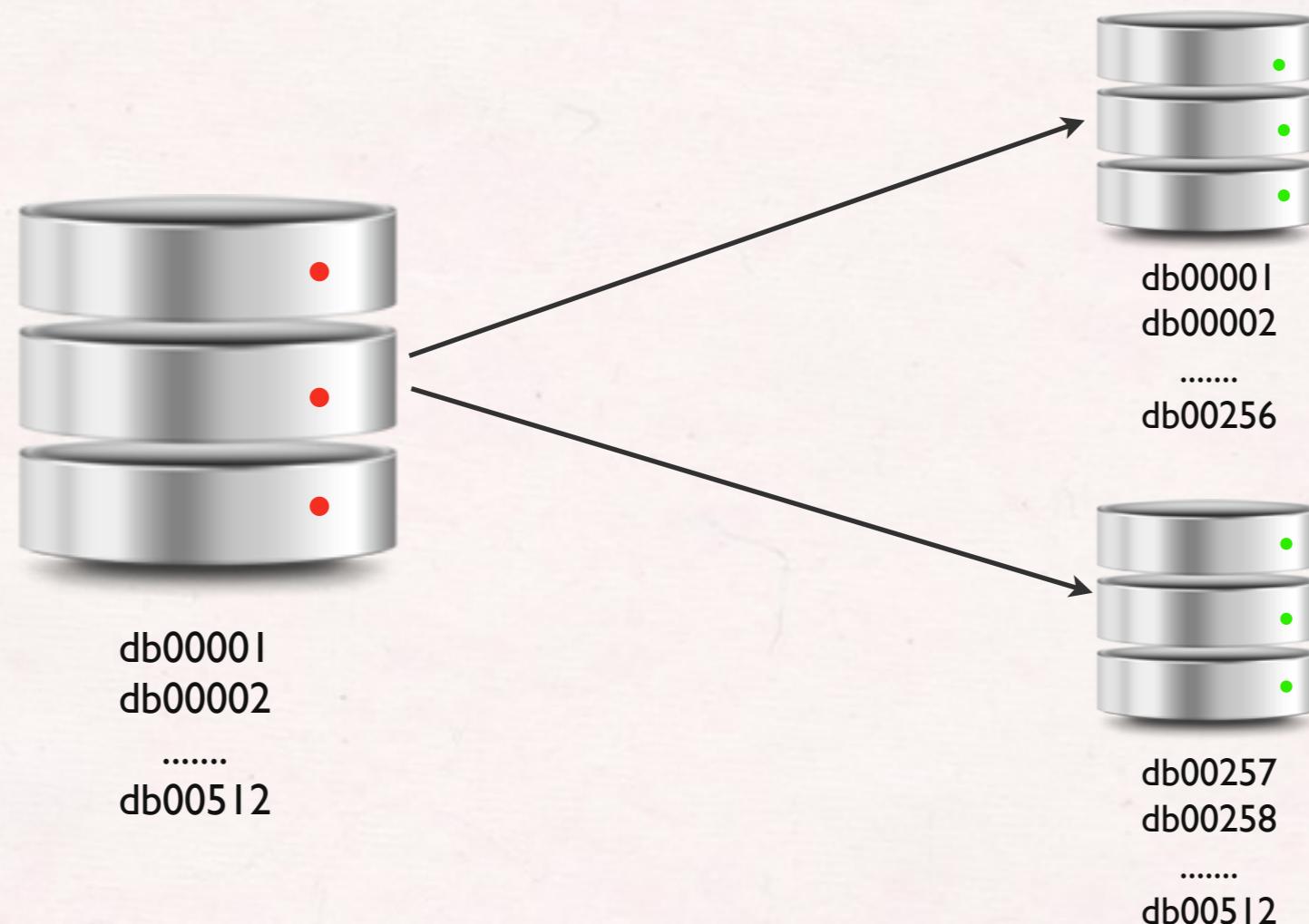
Initially, 8 physical servers, each with 512 DBs

# High Availability



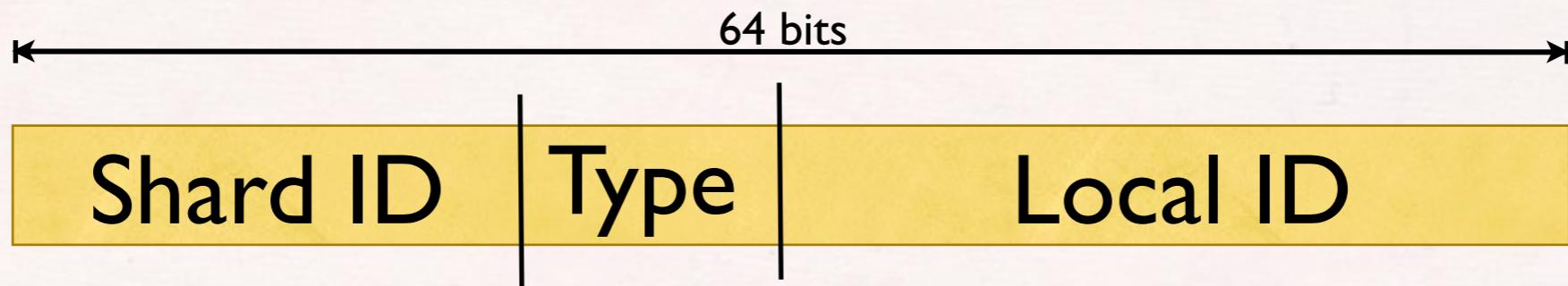
Multi Master replication

# Increased load on DB?



To increase capacity, a server is replicated and the new replica becomes responsible for some DBs

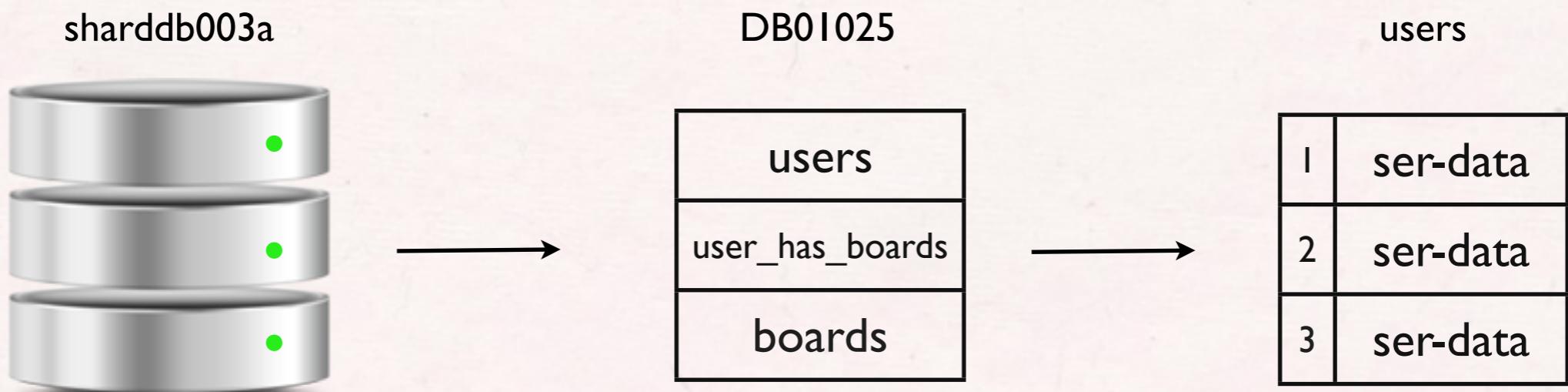
# ID Structure



- A lookup data structure has physical server to shard ID range (cached by each app server process)
- Shard ID denotes which shard
- Type denotes object type (e.g., pins)
- Local ID denotes position in table

# Lookup Structure

```
{"sharddb001a": ( 1, 512),  
 "sharddb002b": ( 513, 1024),  
 "sharddb003a": (1025, 1536),  
 ...  
 "sharddb008b": (3585, 4096)}
```



# ID Structure

- New users are randomly distributed across shards
- Boards, pins, etc. try to be collocated with user
- Local ID's are assigned by auto-increment
- Enough ID space for 65536 shards, but only first 4096 opened initially. Can expand horizontally.

# Objects and Mappings

- Object tables (e.g., pin, board, user, comment)
  - Local ID → MySQL blob (JSON / Serialized thrift)
  - Mapping tables (e.g., user has boards, pin has likes)
    - Full ID → Full ID (+ timestamp)
    - Naming schema is *noun\_verb\_noun*
  - Queries are PK or index lookups (no joins)
  - Data **DOES NOT MOVE**
  - All tables exist on all shards
  - **No schema changes required** (index = new table)

# Loading a Page

- Rendering user profile

```
SELECT body FROM users WHERE id=<local_user_id>
```

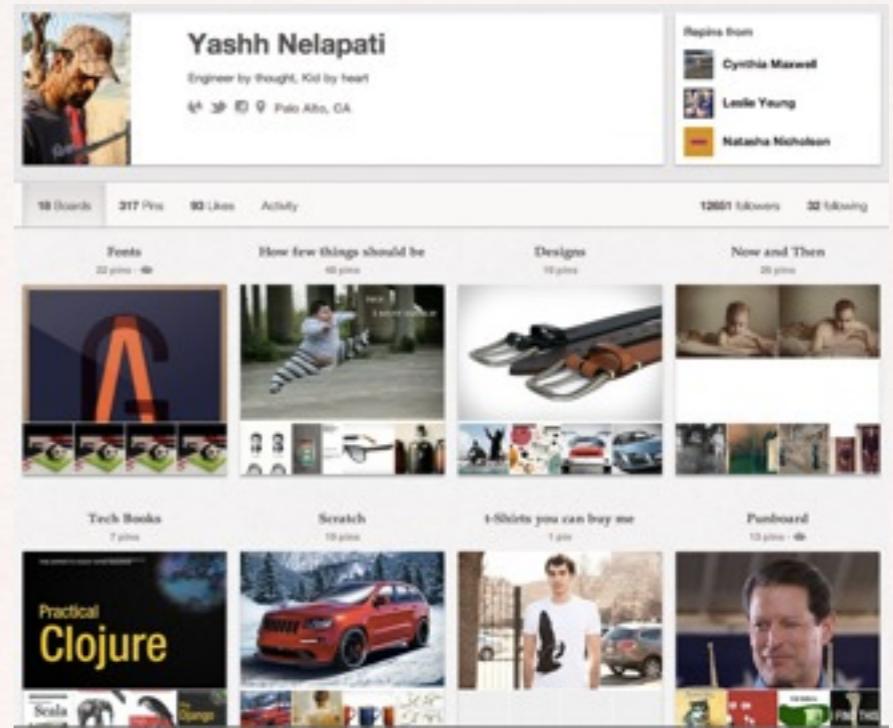
```
SELECT board_id FROM user_has_boards WHERE user_id=<user_id>
```

```
SELECT body FROM boards WHERE id IN (<board_ids>)
```

```
SELECT pin_id FROM board_has_pins WHERE board_id=<board_id>
```

```
SELECT body FROM pins WHERE id IN (pin_ids)
```

- Most of these calls will be a cache hit
- Omitting offset/limits and mapping sequence id sort



# Scripting

- Must get old data into your shiny new shard
- 500M pins, 1.6B follower rows, etc
- Build a scripting farm
  - Spawn more workers and complete the task faster
  - Pyres - based on Github's Resque queue

# In The Works

- Service Based Architecture
  - Connection limits
  - Isolation of functionality
  - Isolation of access (security)
- Scaling the Team
- New features

# **Lesson Learned #3**

## Keep it fun.

# NEED ENGIES

[jobs@pinterest.com](mailto:jobs@pinterest.com)



# Questions?

[marty@pinterest.com](mailto:marty@pinterest.com)

[yashh@pinterest.com](mailto:yashh@pinterest.com)