Scaling

Pinterest

Marty Weiner

Grayskull, Eternia

Yashh Nelapati

Gotham City

Pinterest is . . .

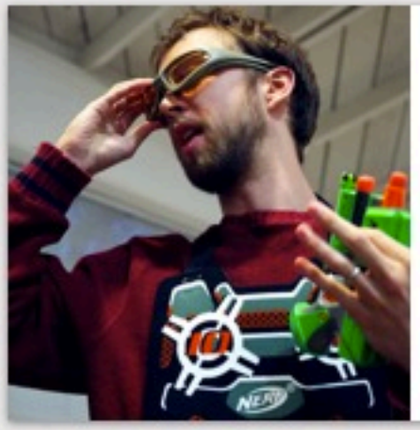
An online pinboard to organize and share what inspires you.

Pinterest

Search

Pinterest

Add + About Marty



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).. Any questions? Email away! ...

Grayskull, Eternia

Repins from


Justin Edmund


Candice Weiner

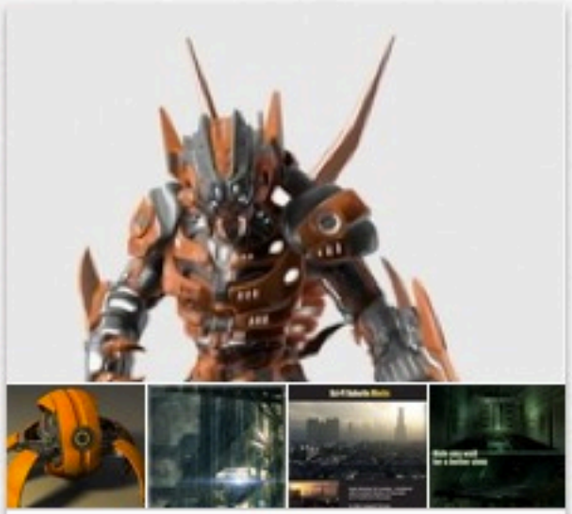

Enid Hwang

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

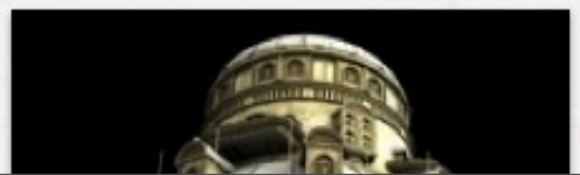
Edit

3D Models - Buildings
9 pins

3D Models - Space
14 pins

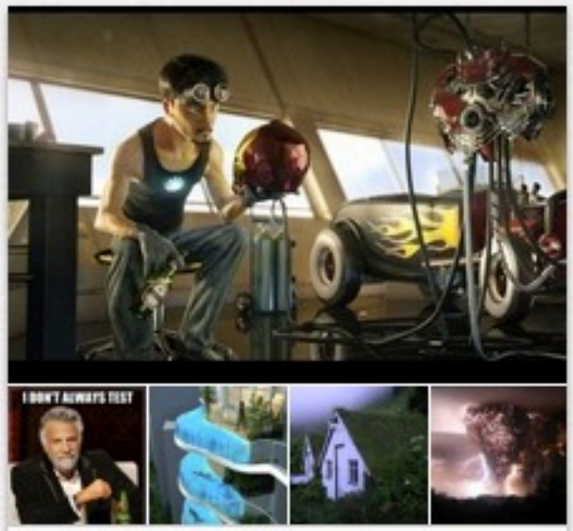
3D Models - Low Poly
8 pins

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

Search

Pinterest

Add + About Marty

Pinners you follow · Everything · Videos · Popular · Gifts

Invite Friends

Friends to Follow

See All

Gene Warren Follow

Jessica Spurling Follow

angela Follow

Recent Activity

Hemanth Pai repinned your pin.
1 hour ago

Luis Madrigal and **1 other** are now following your pins.
9 hours ago

myong greenspan and **2 others** liked your pin.
9 hours ago

Phyllis Weiner repinned your pin.
13 hours ago

Phyllis Weiner liked your pin.
13 hours ago

Phyllis Weiner repinned your pin.
14 hours ago

Phyllis Weiner liked your pin.
14 hours ago

Phyllis Weiner repinned your pin.
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**

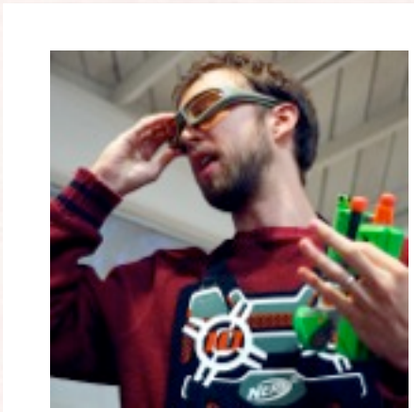
Susan Peck This looks delicious, but it's taking me to an add. I'm confused right now.

Susan Peck via **Ashley FitzSimmons** onto **clothing**

Ahh this looks fun, I have to try this!

Susan Peck via **Melissa Guffy** onto **Totally**

Relationships



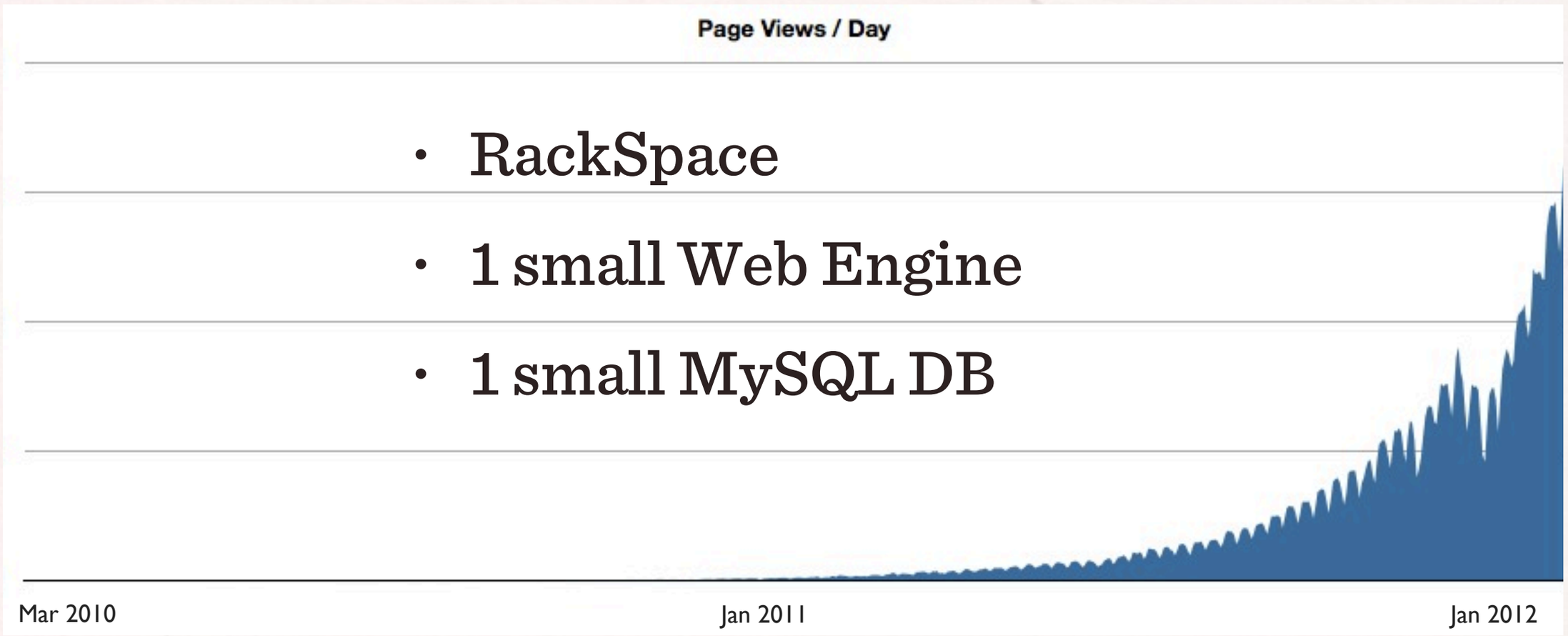
Marty Weiner

Grayskull, Eternia

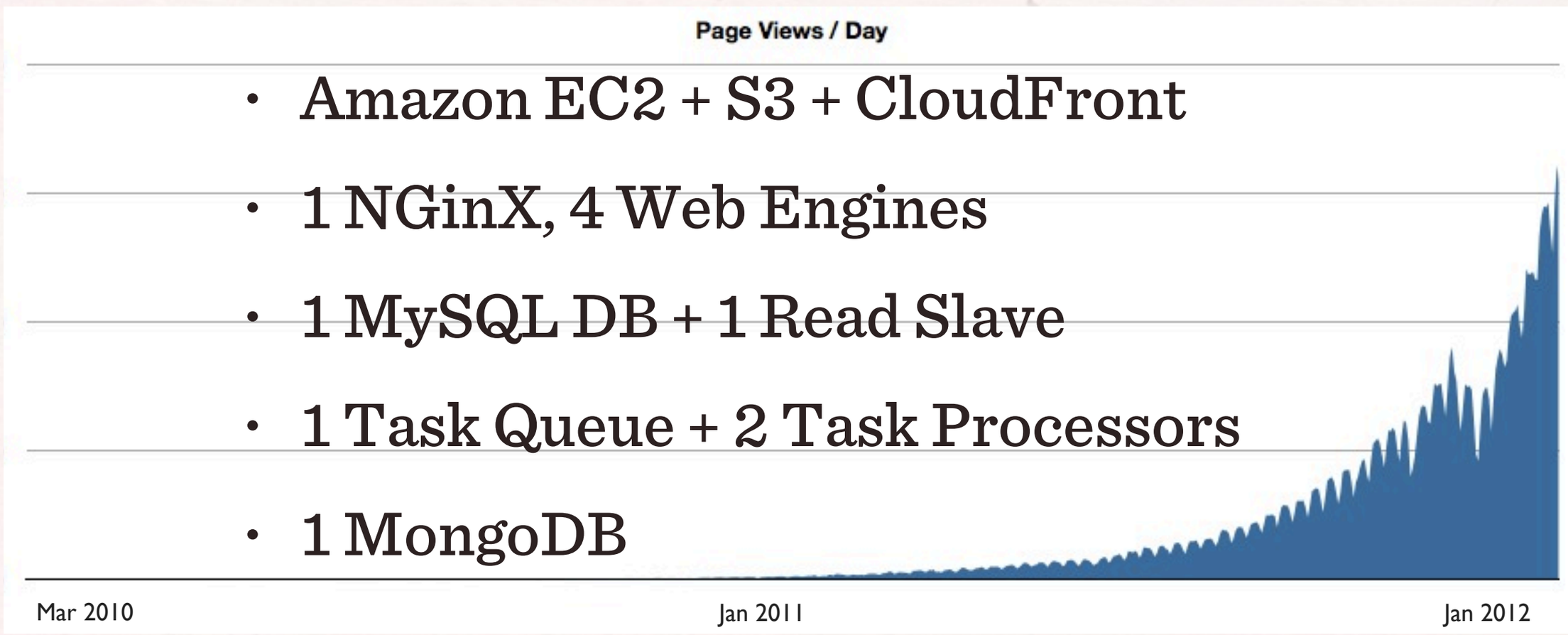
Yashh Nelapati

Gotham City

- RackSpace
- 1 small Web Engine
- 1 small MySQL DB

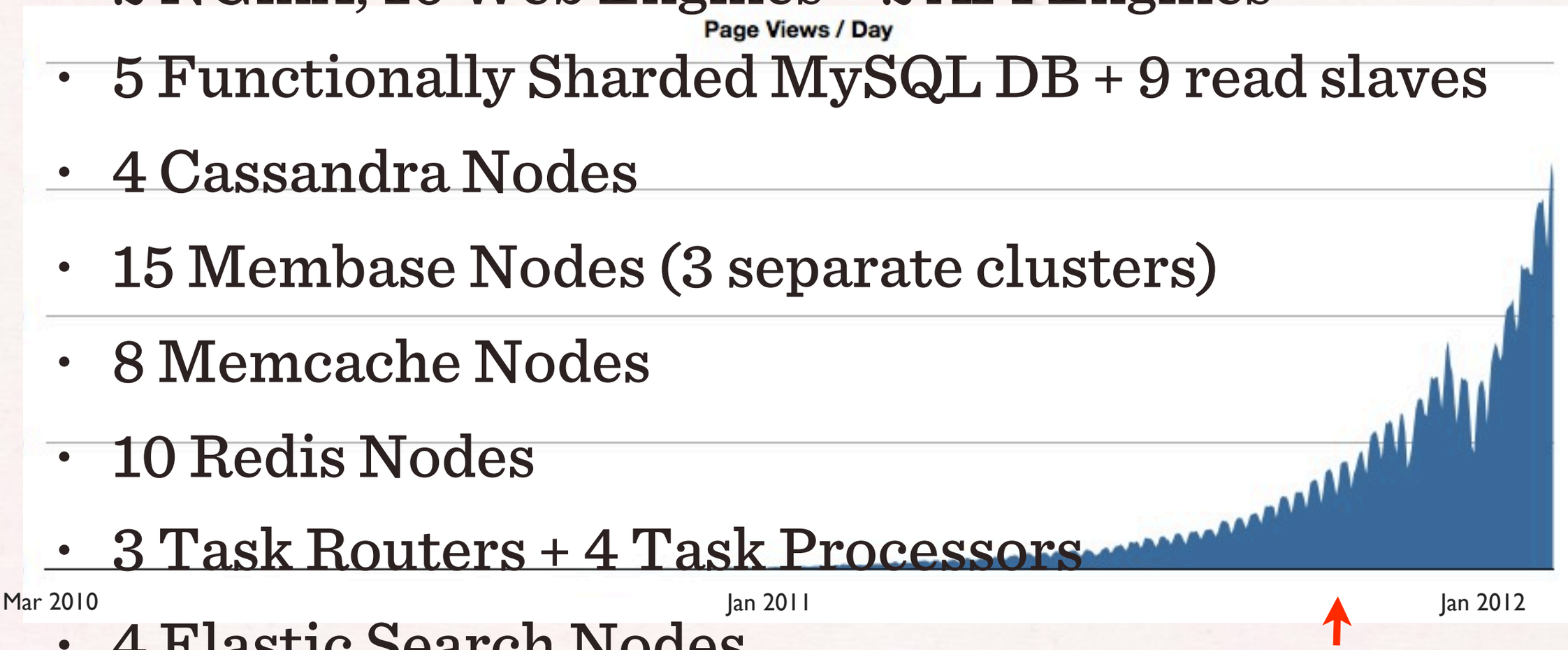


- Amazon EC2 + S3 + CloudFront
- 1 NGinX, 4 Web Engines
- 1 MySQL DB + 1 Read Slave
- 1 Task Queue + 2 Task Processors
- 1 MongoDB



Pinterest

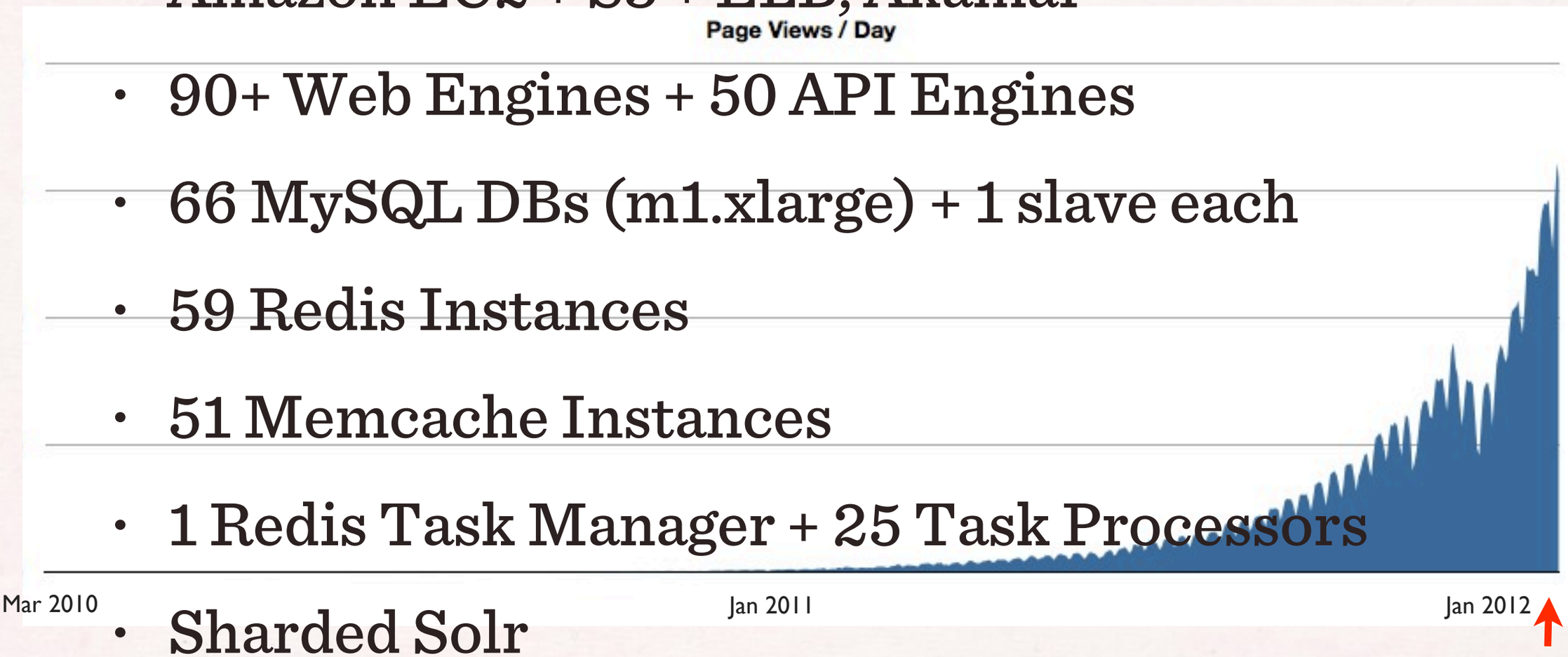
- Amazon EC2 + S3 + CloudFront
- 2 NGinX, 16 Web Engines + 2 API Engines
- 5 Functionally Sharded MySQL DB + 9 read slaves
- 4 Cassandra Nodes
- 15 Membase Nodes (3 separate clusters)
- 8 Memcache Nodes
- 10 Redis Nodes
- 3 Task Routers + 4 Task Processors
- 4 Elastic Search Nodes
- 3 Mongo Clusters



Lesson Learned #1

It will fail. Keep it simple.

- Amazon EC2 + S3 + ELB, Akamai
- 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



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
- Atomic operations
- Consistently good performance
- Free

What are the cons?

- They don’t do everything for you
- Out of the box, they wont scale past 1 server, won’t have high availability, won’t bring you a drink.

Clustering
vs
Sharding

Clustering vs Sharding

Distributes data across nodes automatically	Distributes data across nodes manually
Data can move	Data does not move
Rebalances to distribute load	Split data to distribute load
Nodes communicate with each other (gossip)	Nodes are not aware of each other

Why Clustering?

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

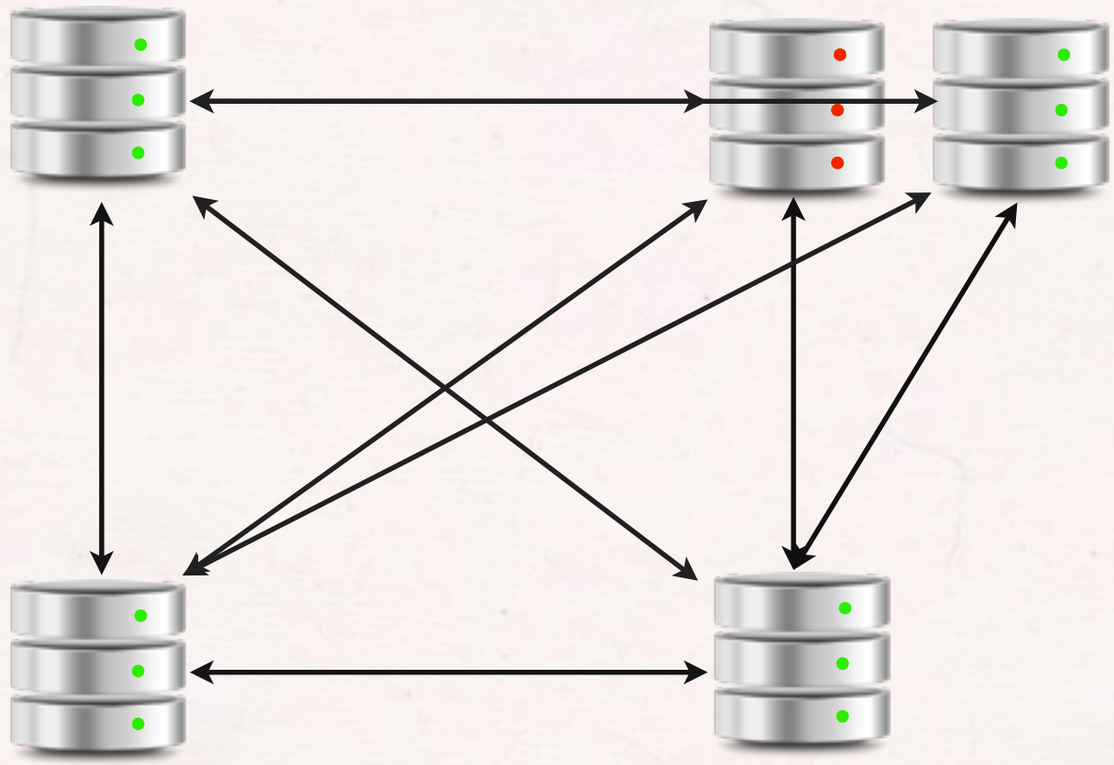
What could possibly go wrong?

source: thereifixedit.com

Why Not Clustering?

- Still fairly young
- 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



Cluster Management Algorithm

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
- 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

1 DB + Foreign Keys + Joins

1 DB + Denormalized + Cache

1 DB + Read slaves + Cache

Several functionally sharded DBs + Read slaves + Cache

ID sharded DBs + Backup slaves + Cache

Watch out for...

- Lost the ability to perform simple JOINS
- No transaction capabilities
- Extra effort to maintain unique constraints
- Schema changes requires more planning
- Single report requires running same query on all shards

How we sharded

Sharded Server Topology

db00001
db00002
.....
db00512

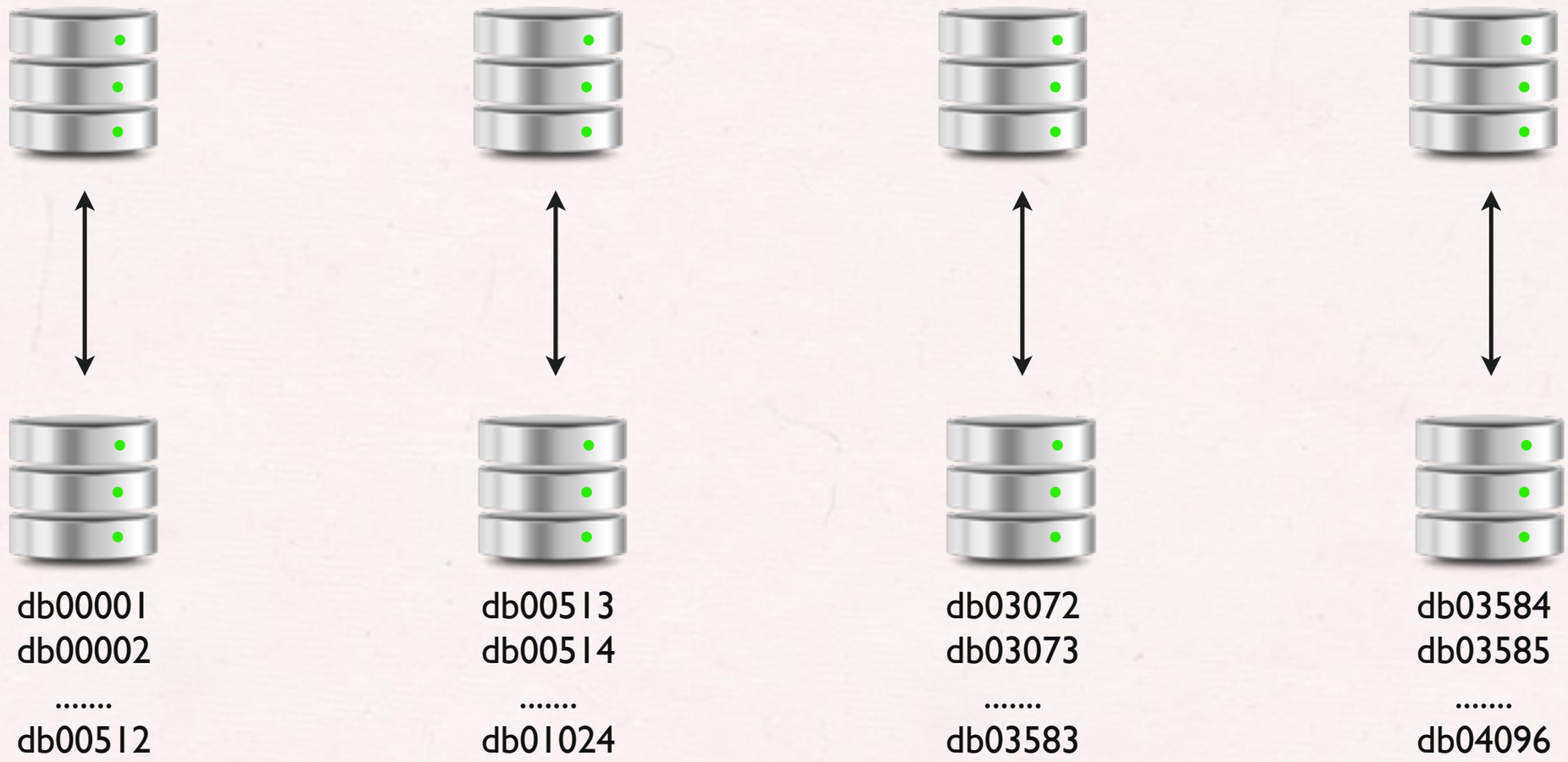
db00513
db00514
.....
db01024

db03072
db03073
.....
db03583

db03584
db03585
.....
db04096

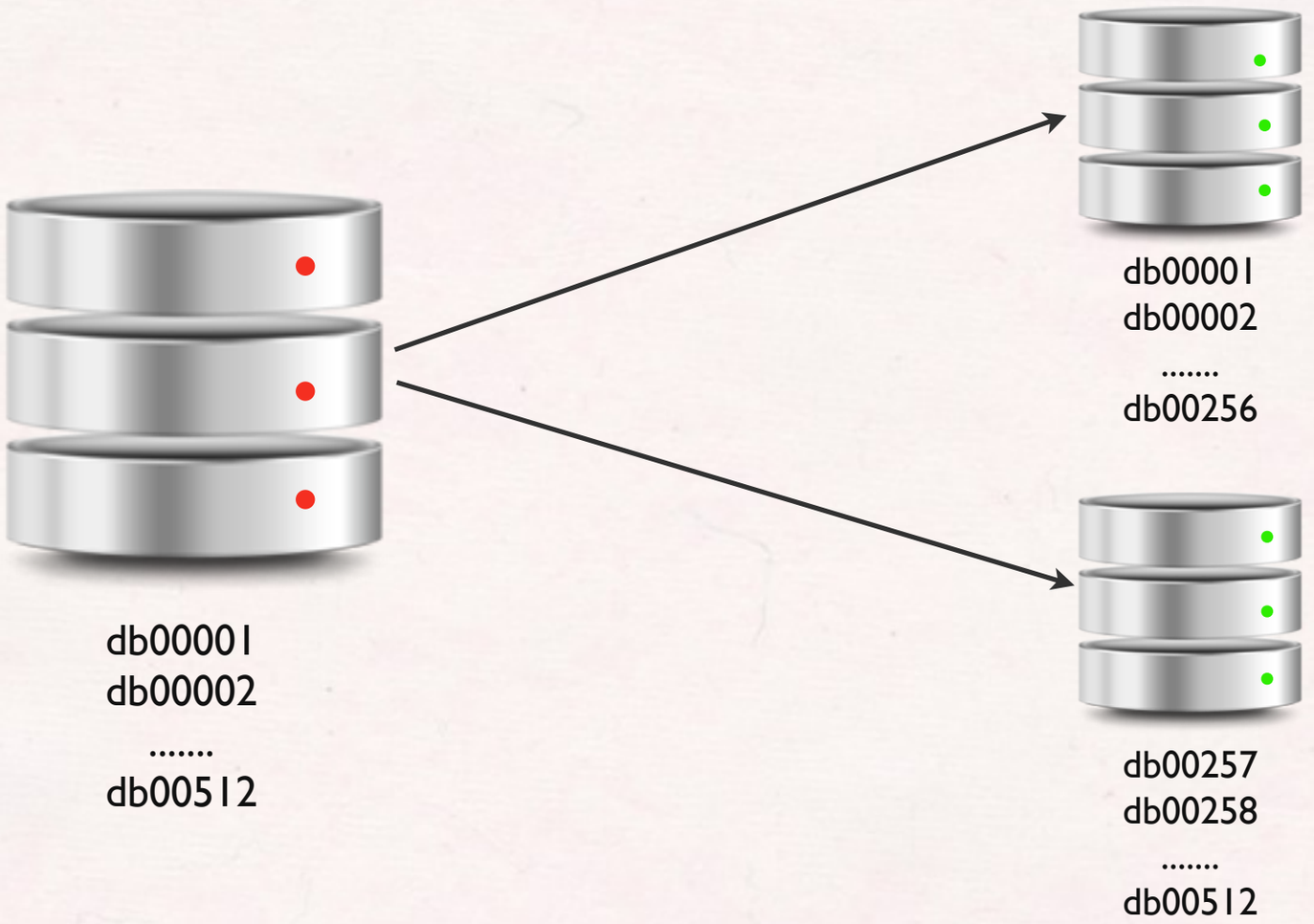
Initially, 8 physical servers, each with 512 DBs

High Availability



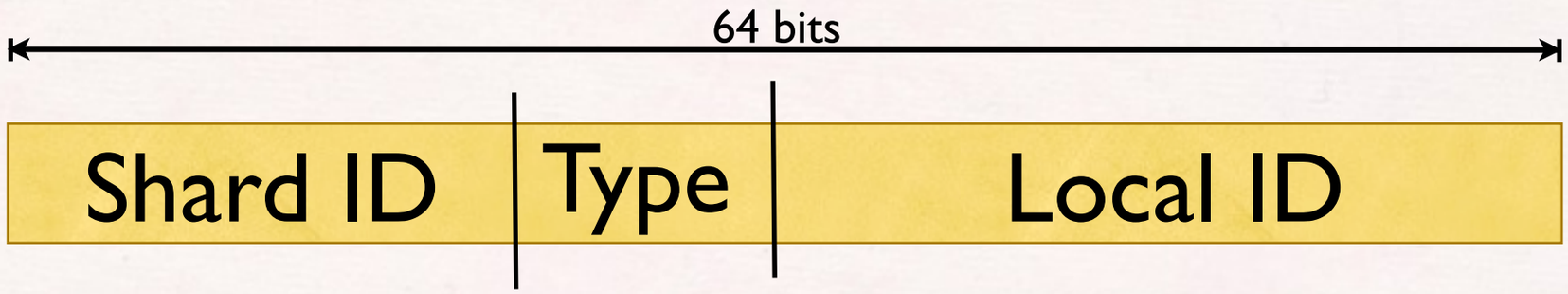
Multi Master replication

Increased load on DB?



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

ID Structure



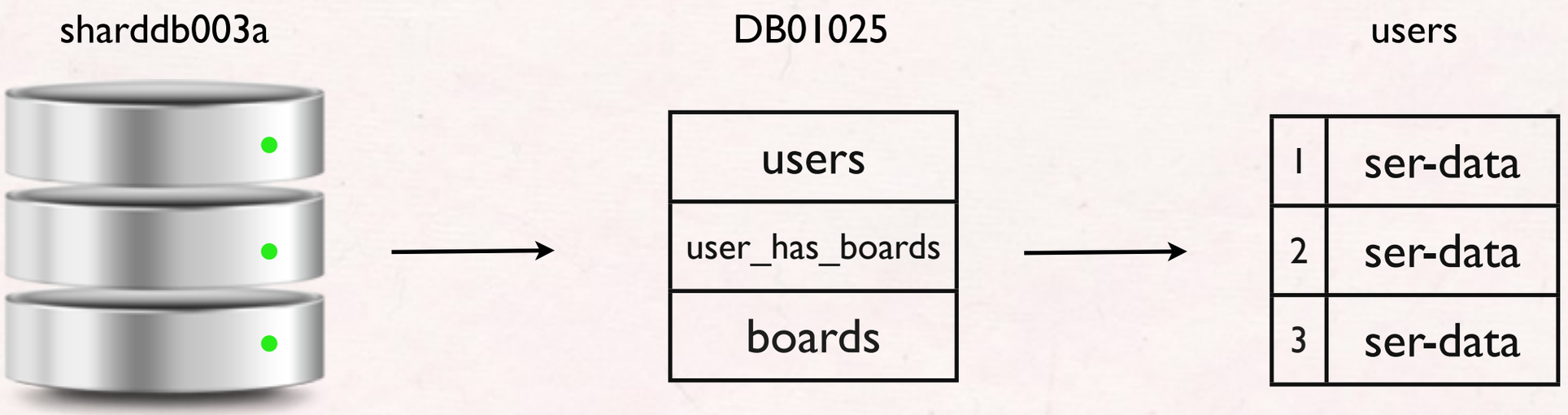
- A lookup data structure has physical server to shard ID range (cached by each app server process)
- Shard ID denotes which shard
- Type denotes object type (e.g., pins)
- Local ID denotes position in table

Why not a ID service?

- It is a single point of failure
- Extra look up to compute a UUID

Lookup Structure

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



ID Structure

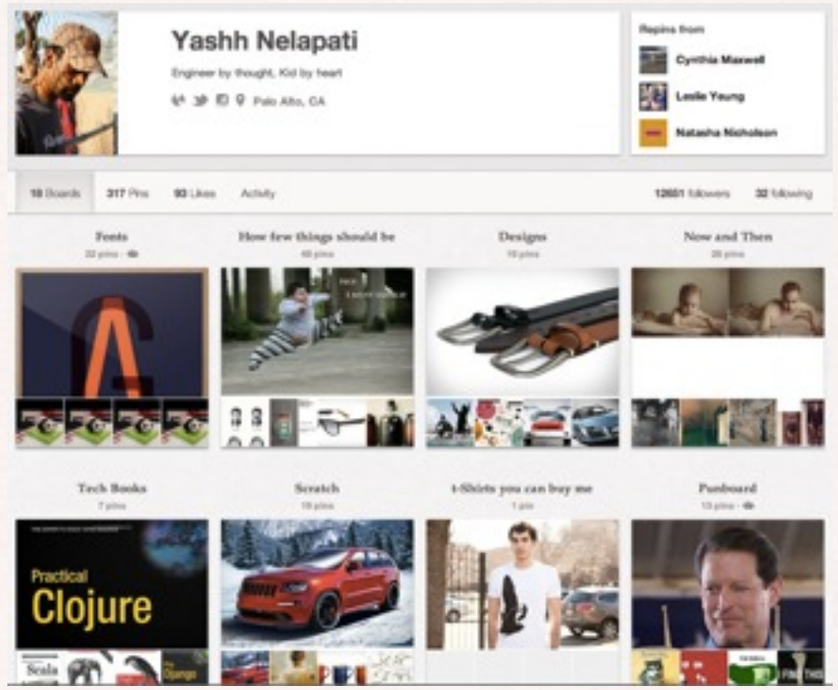
- New users are randomly distributed across shards
- Boards, pins, etc. try to be collocated with user
- Local ID's are assigned by auto-increment
- Enough ID space for 65536 shards, but only first 4096 opened initially. Can expand horizontally.

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)

```
def create_new_pin(board_id, data):
    shard_id, type, local_board_id = decompose_id(board_id)
    local_id = write_pin_to_shard(shard_id, PIN_TYPE, data)
    pin_id = compose_id(shard_id, PIN_TYPE, local_id)
    return pin_id
```

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

Future problems

- Connection limits
- Isolation of functionality

Interesting Tidbits

- Use read slaves for read only as a temporary measure
  - Lag can cause difficult to catch bugs
- Use Memcache/Redis as a feed!
  - **Append** new values. If the key does not exist, append will fail so no worries over partial lists
- Split background tasks by priority
- Write a custom ORM tailored to your sharding

Lesson Learned #3

Working at Pinterest is AWESOME

We are Hiring!

jobs@pinterest.com

Pinterest

Questions?

marty@pinterest.com

yashh@pinterest.com