Just Add an Index

Cassandra in Production @ Digg

Quick Background

- over 40 million uniques
- Standard LAMP architecture
  - PHP, MySQL, Memcache, Gearman, Mogile
- Need better scalability & performance

Sharding MySQL

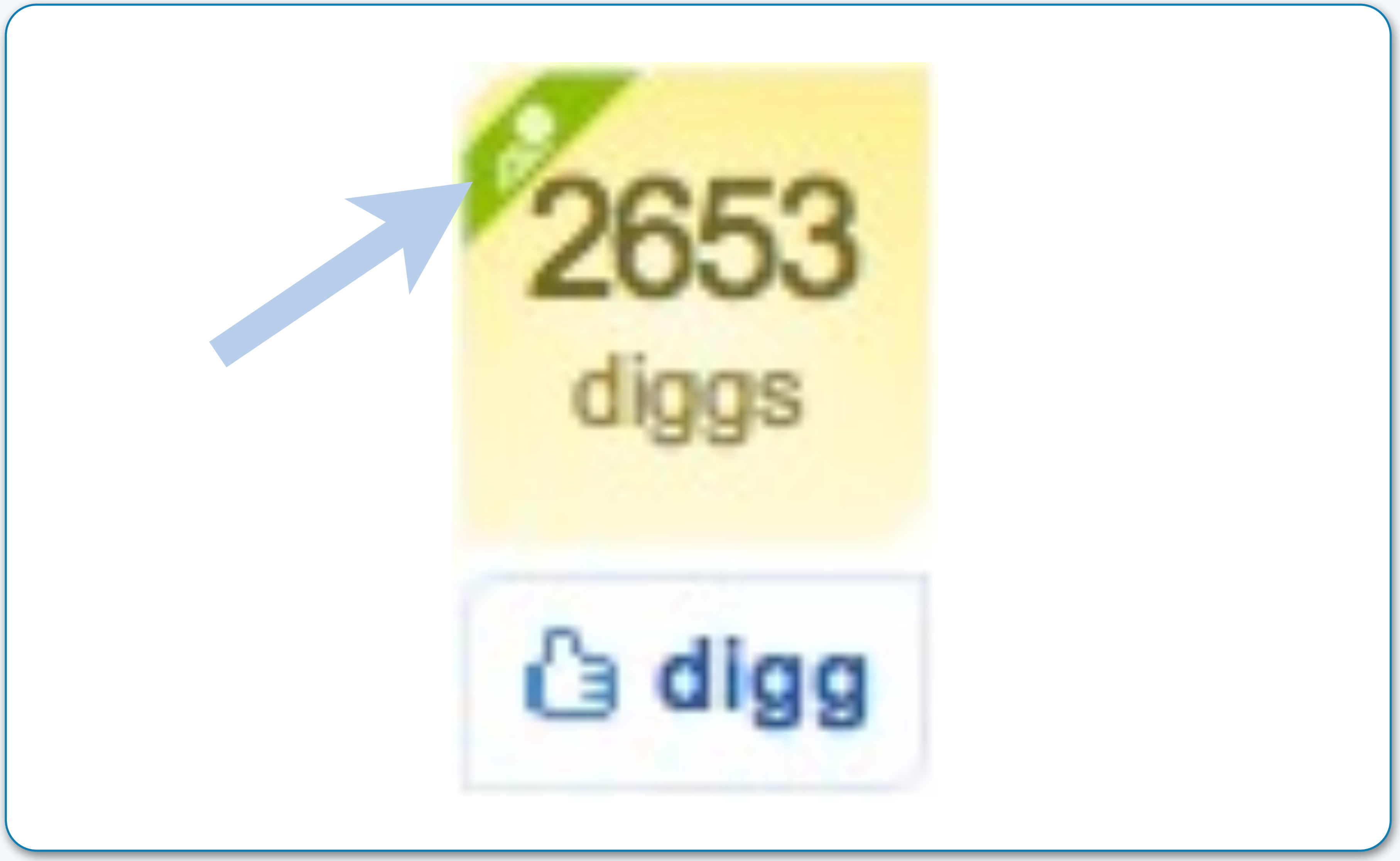
- Vertical partitioning is easy
- IDDB for horizontal partitioning
    - Home grown, PHP based
    - Needed a bunch of work to make administration easy
- Now you've tossed away the "R"

Check Out Alternatives

- Why?
  - You’ve tossed joins etc, why not?
  - Easier administration
  - Scalability & performance
  - Something open source
- Encouraged by management

Why Cassandra

- Easy administration
- No SPF
- More than just key/value, flexible schema
- Super fast writes
- Community is growing
- Java



Proof of Concept

Problem & Approach

- Need to know which (if any) of your friends have Dugg each and every story
- De-normalize the data in SuperColumns
- Reads should be super easy - no complicated queries needed

Friend_Diggs CF

```
Friend_Diggs { // Column Family
  12345 : { // story_id as Row key
    user_id: { // SuperColumns are User's IDs
      friend_id1: true,
      friend_id2: true,
    }
  }
}
```

Dark Launch

- Modify app to read/write from Cassandra based on config settings
- Flip "write to Cassandra" switch "on"
- Still reading & writing to/from MySQL
- Watch Cassandra as data rolls in
- Internally we use web servers with "Cassandra reads" set to "on"

Results: Good But...

- Race conditions
- Data corruption
- Ran out of file descriptors on server
- Needed better monitoring/metrics
- Hinted handoff bugs
- Needed root access. Ops doesn't like that

Sounds Scary: But...

- We've been working on the internals
  - 2 1/2 contributors, 1 Apache committer
  - In house ability to debug & fix issues
- Talk to the Facebook guys
- Brought Jonathan Ellis in to Digg to consult
- Being involved in the community helps!

Real Launch

- Fixed the issues w/ help of community
- Backfilled "old" data using Hadoop/BMT
    - ~3TB of data on a 12 node cluster
- Flip all web servers to read from Cassandra
- Blogged about it & got crap from ~~trolls~~ armchair engineers

It's Running Fine

- Cluster's all good
- Backed down to 8 machines
- Not using Memcache & doing well
  - < 1 ms writes
  - ~4-6 ms reads

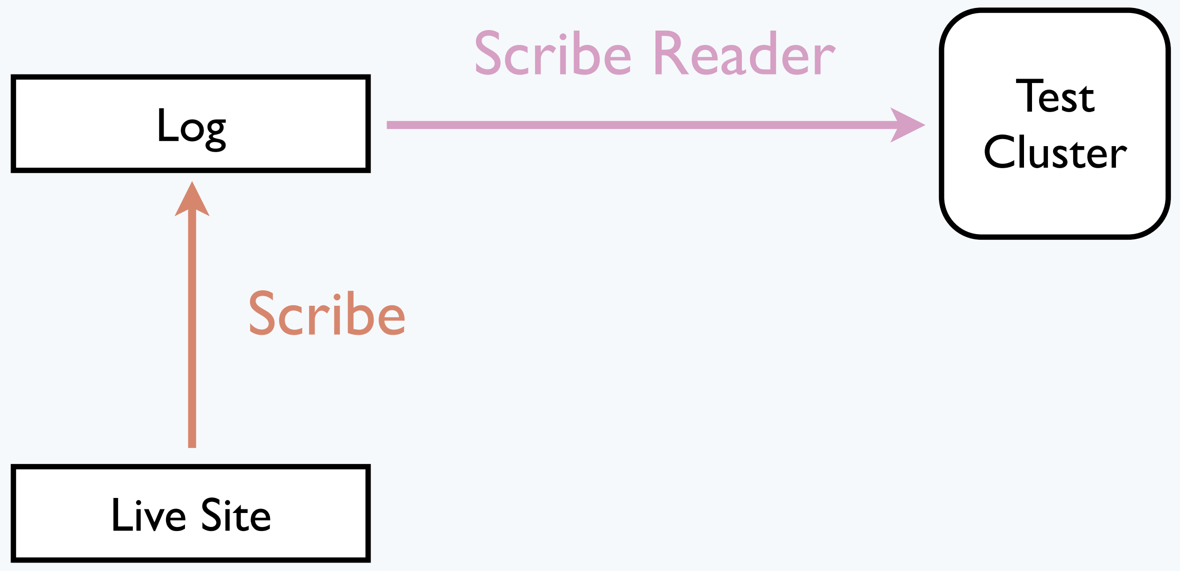
So Now What?

- Let’s port the rest of Digg to Cassandra
- Which parts?
  - All of it...
- New problems:
  - We need live data
  - We need real traffic to test performance

So Far, So Good

- We have an internal prototype now
  - Category pages: Apple, Politics etc
  - Front page
- Working on porting Users, Comments etc (aka: everything else)
- We'll have it all done... some day

Live Data to Test Site



- Log Stories, Diggs, Buries via Scribe
- Sip off the Scribe log
- Get data to test cluster in “real time”

Testing Performance

- Auto-Request tool
    - Production Apache requests logged via Scribe
    - With each request we hit the test cluster
- Impressive results w/o using Memcache
- Also have a “replay” tool with an optional speed-up/slow-down factor

Awesome... But

- There's always another problem
- This is probably the most important issue to address...

The Rest of the Team

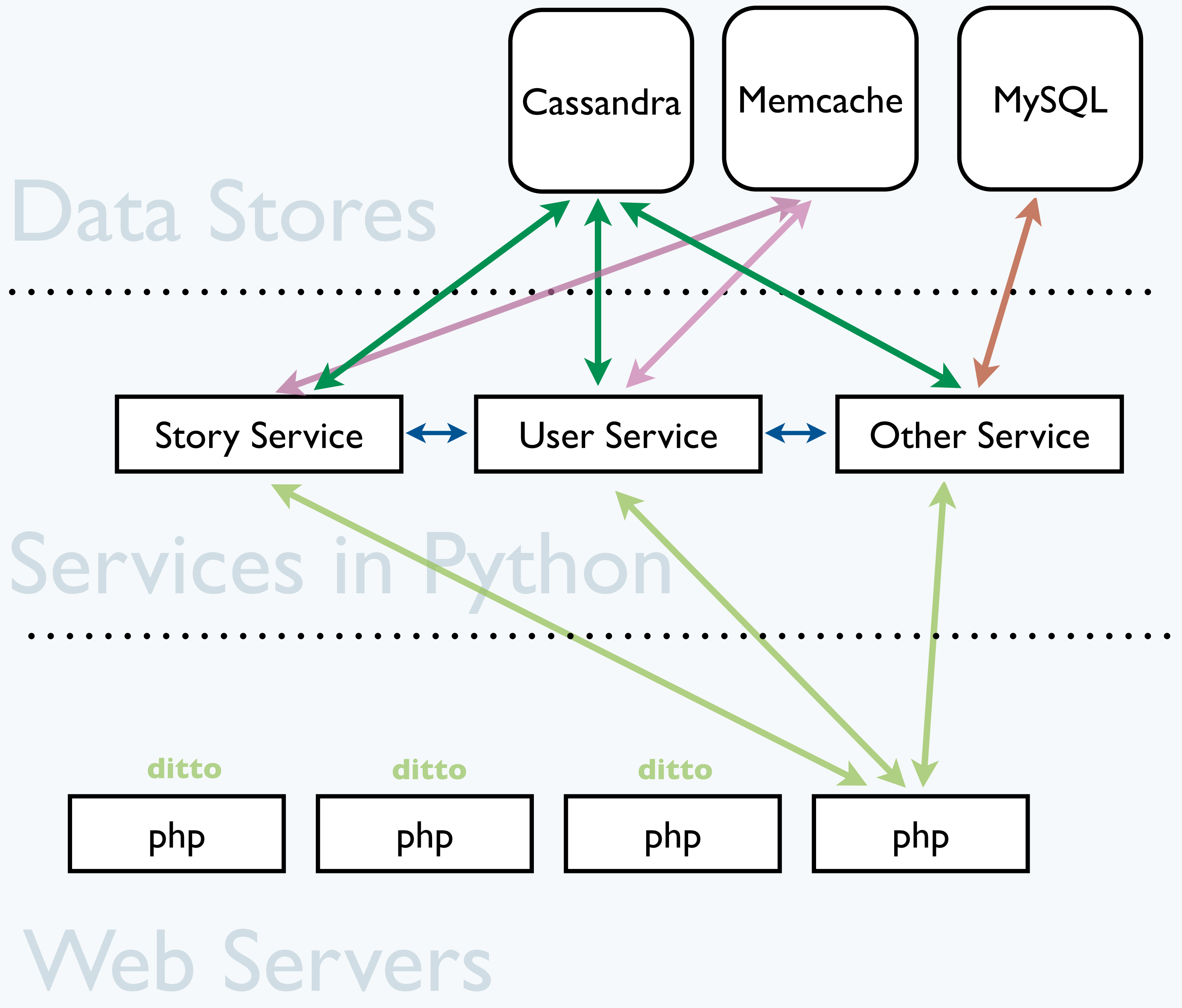
I Don’t Get It

- WTF is A SuperColumn?
- How do I query it?
- Is there a GUI to see the data?
- Can I sort?
- WTF is Thrift?
- I have to manage my own indexes? Really?

Education & Tools

- Lots of internal training, docs & examples
- Lazyboy
  - easy CRUD
  - “views” to manage secondary indexes
- New architecture
  - From LAMP to LAMPCRMTP

New Architecture



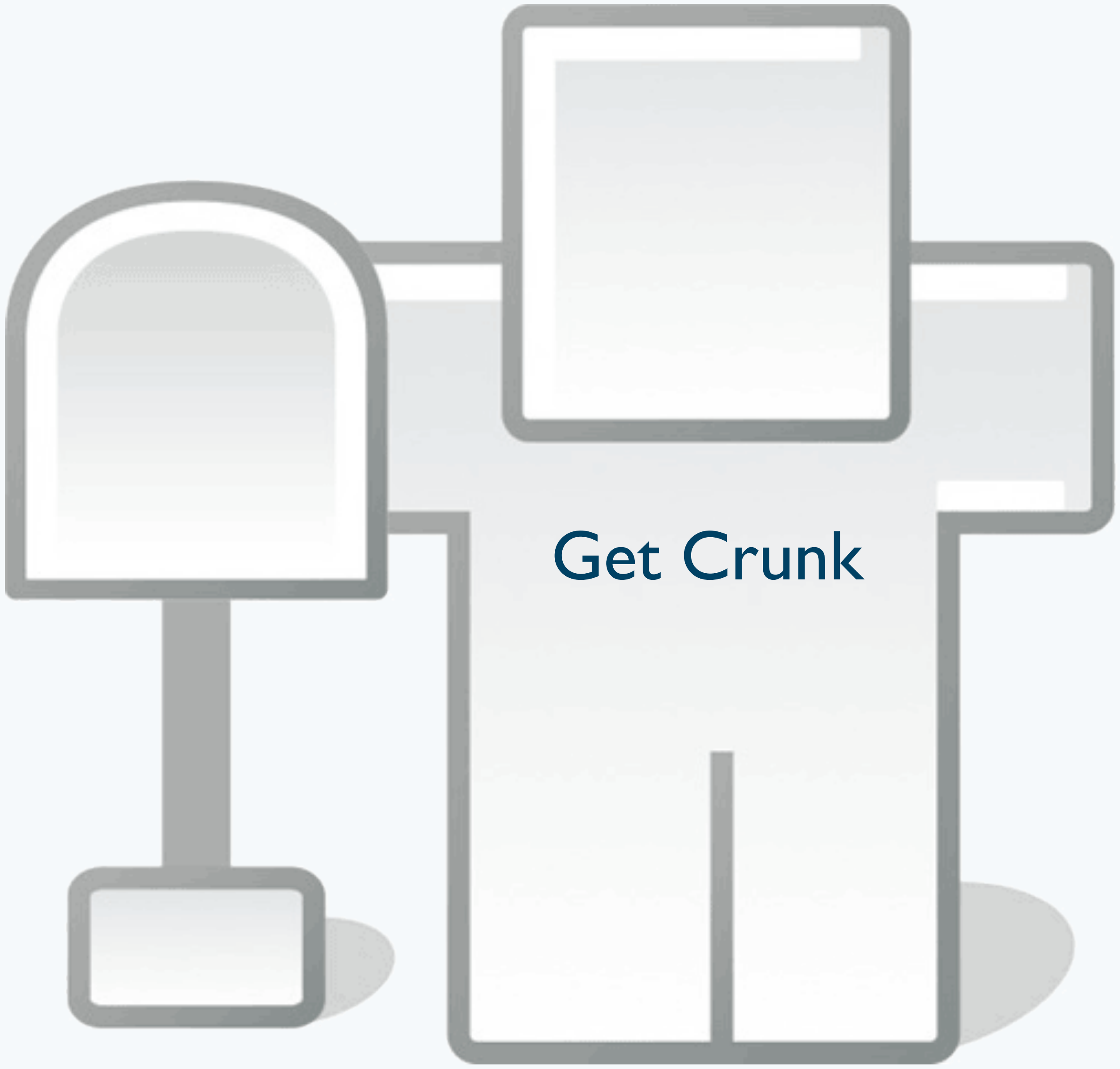
Yup, Still Using MySQL

- It doesn’t suck at queries
  - Very “controlled” applications
  - Make sure indexes always fit in memory
- Cassandra can’t do everything
- Right tool for the right job

A Different Mindset

- No, we don't hate SQL (MySQL etc)
  - select * from bla where x > 2 is nice
  - I wish scaling MySQL would just "work"
  - Sharding can scale but has it's own issues
- Different tools are good
  - A hammer and rock can both drive a nail

That's All Folks



Modeling Data

- Store “objects” in a Row
- Store indexes in another ColumnFamily
- Custom comparators turn out to be key
  - LongSting & FloatString
- Lots of multi-get
  - Didn’t exist. Contributed by Goffinet

Basic “Object”

```
Story { // Column Family
  // LongString Row Key
  20090802hhmmssmm:gibberish : {
    // Columns
    title: Gettin’ Crunk in ATL,
    description: NoSQL East geeks get hella drunk,
    user_id: 123456789,
    category:  Apple
  }
}
```

Index: Category & Date

```
Category_Date { // Column Family
  Apple : { // Row Key
    // Columns - names are LongSting type
    20090802hhmmssmm:gibberish: 1,
    20090802hhmmssmm:bla: 1,
  },
  Orange : { // Row Key
    20090809hhmmssmm:yadayada: 1,
  },
}
```