Testing Riak for multiple datacenter support: a case study

Jim Englert
Gilt Groupe
jenglert@gilt.com

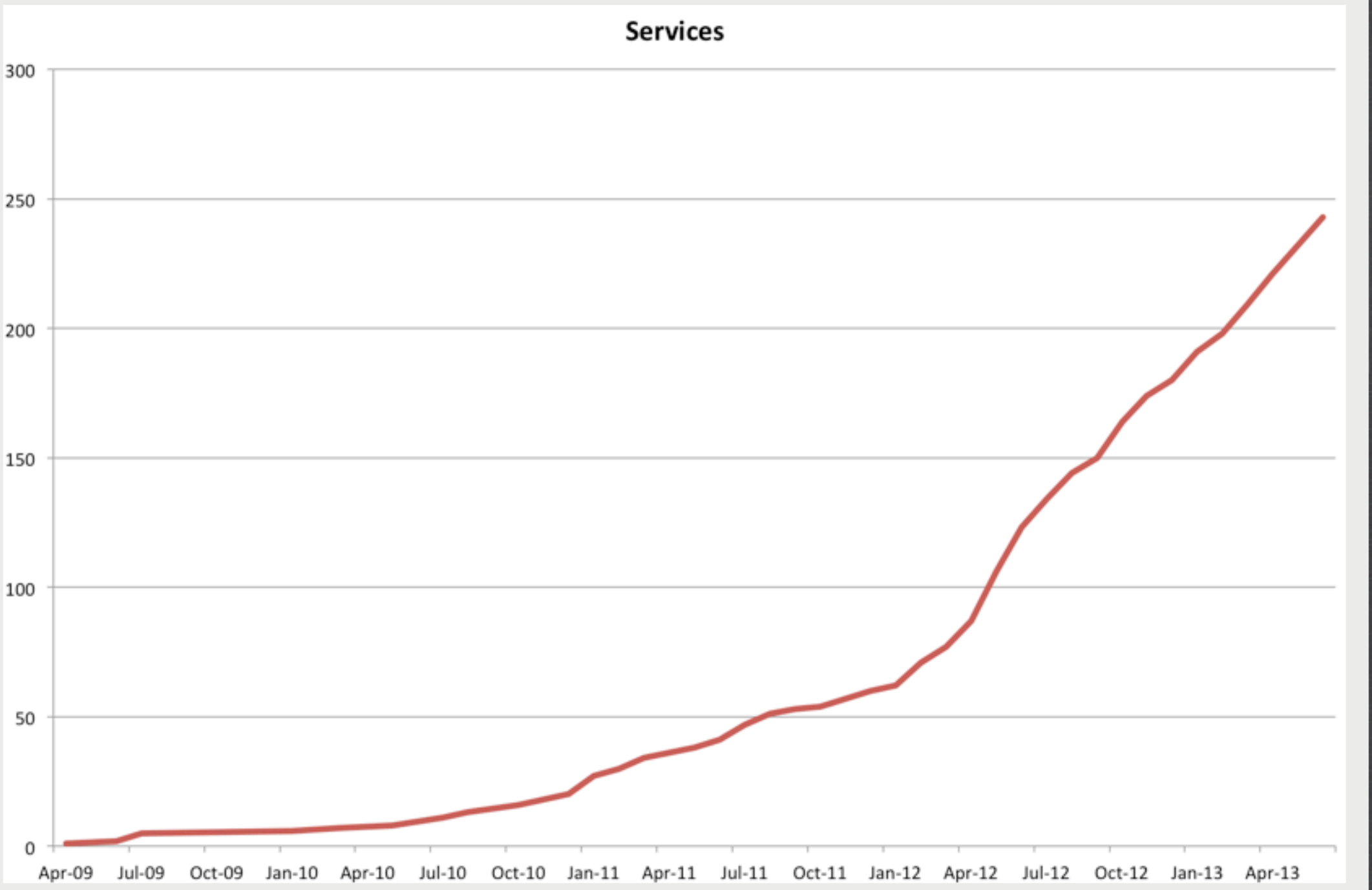
ONE WEEK HACKATHON

- Gilt's offices in Dublin
- 1 week (4 days really)
- From Basho: Seth Thomas and Steve Vinoski
- 3-4 engineers from Gilt

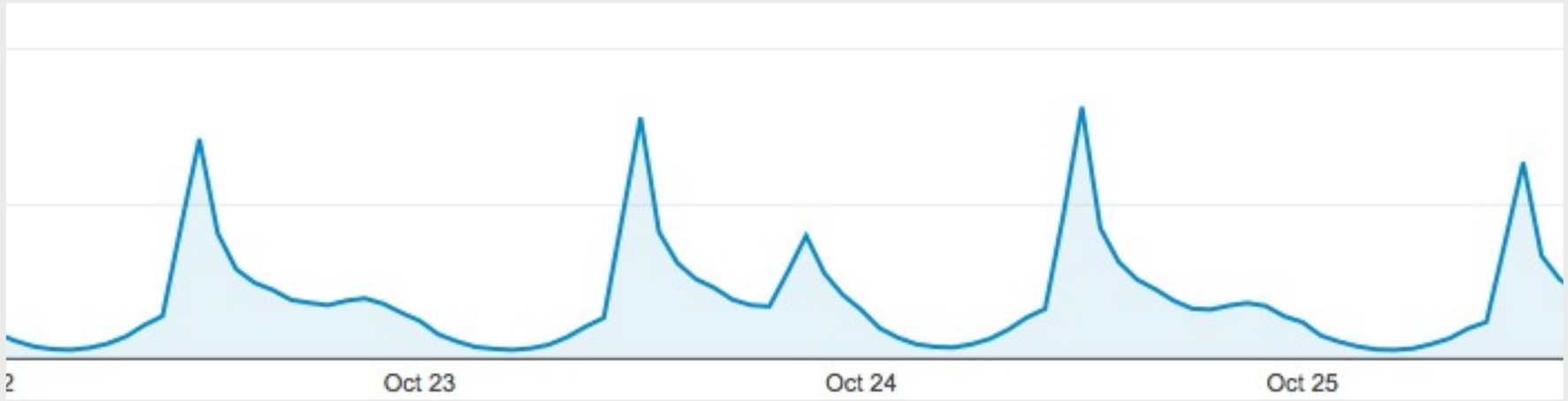
SETH AND STEVE

WHAT IS GILT.COM

- Flash Sales
  - show duration, limited inventory
- Lots of personalization
- Lots of traffic at Noon
- **Micro-service architecture**
  - Hundreds of services



SERVICE GROWTH OVER TIME



VISITORS AT NOON

WHAT ARE WE DOING?

- Realtime Personalization
  - Complex event processing w/ “big data”

K/V STORE NEEDS

- Multi-dc
- Avoid active/passive DataCenter
  - IMO, expecting a handover to work in an emergency is never going to happen. The only real solution is to actively serve traffic from multiple DC's
- Some functionality may only need to live in 1 data center
- Requests should not cross DC's

WHY RIAK?

- Supports Active-Active configuration across data centers
- High Performance (or we we had heard)
- People!

ACTIVE-ACTIVE IS IMPORTANT

- Riak manages communication between DC's for you
- Allows us to leverage other data centers for processing
  - EC2
  - Very quick access
  - No extra work

WHAT IS RIAK?

- KV Store
- Based on Amazon's Dynamo paper
- Supports secondary indexes

HOW TO TEST IT?

- Modify our most important service to use it of course

SVC-USER

- Scala service
- HTTP requests w/ JSON response
- Stores information the user
- At least one request per page load
- Performance and scale are an absolute must

SVC-USER

- Greater than 150K requests per minute at peak
- ~1ms response time at peak
- With Mongo, problems with performance under write load
  - Needed to throttle writes
- Active-Active mutli-DC deploy hard

INSTALLATION NOTES

- Used two production data-centers
  - Different sides of the country
- Two rings
- 5 nodes per ring
- Clusters in active-active configuration

BITCASK V. LEVELDB

- Two different storage options for Riak
- LevelDB supports secondary indexes, bitcask does not
- We need a bunch of secondary indexes
- Initially, we managed them in multiple buckets; multiple calls -> slower performance

REWRITE SVC-USER

- Scala services
- Uses cake pattern
- Much easier than anticipated
- Cake pattern made it easy to swap out components
- I actually hate the cake pattern, but it worked out pretty nicely here.

CAKE PATTERN

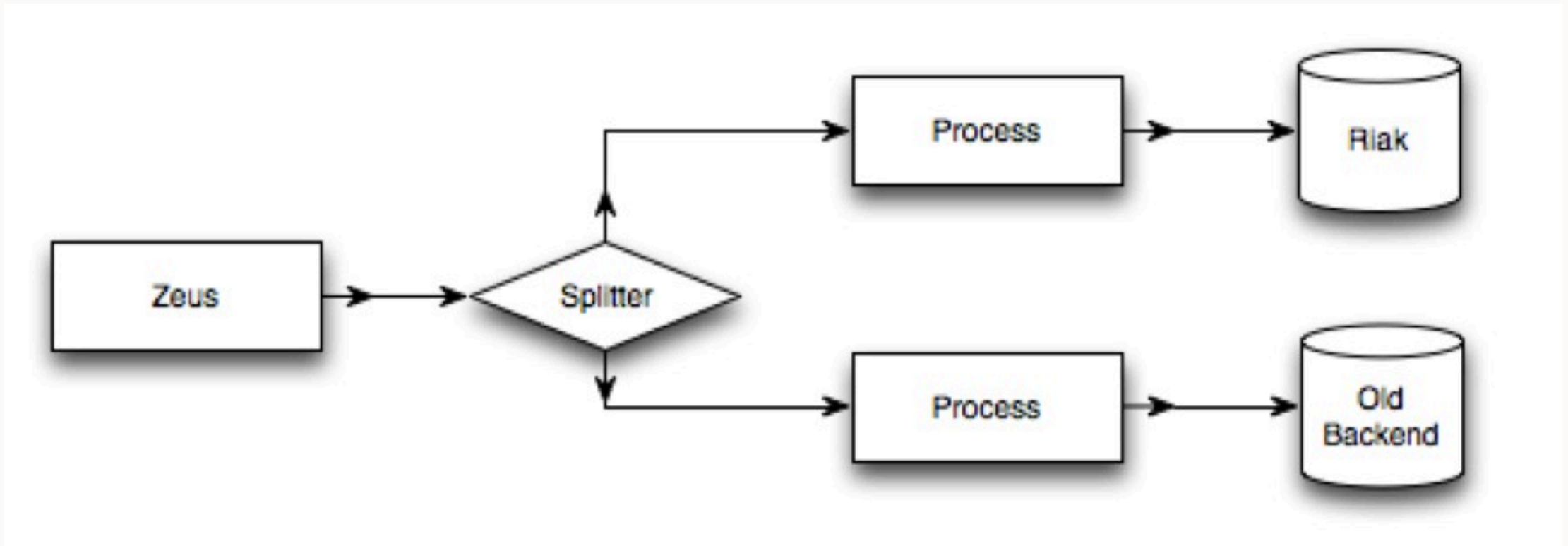
```
val svcUser = new SvcUser with MongoPersistence
val svcUser = new SvcUser with RiakPersistence
```

```
class SvcUser {
  abstract def saveUser(...)
  abstract def getUser: User
}
```

TESTING STRATEGY

- Use real traffic!
- Splitter!
- https://github.com/ebowman/splitter
- Takes traffic and sends it to the real backend and a fake one
- Discards responses from the fake backend

TESTING STRATEGY



BASH THE THING

- Simulate high read / write load
- Zeus Bench
  - Generates HTTP Load
- Simple scala script simulates 1,000's of inputs per second

FAILURE CASES

- Let's kill some nodes!
- kill -9 is your friend
- Performance degraded temporarily with dead nodes

WHAT WENT WRONG?

- We are bad research scientists
- Accidentally published UserCreated messages for fake users

QUESTIONS?