Jeremy Edberg

Information Cowboy
reddit.com

Friday, February 19, 2010

RIDE THE SNAKE

THE REDDIT STORY

Friday, February 19, 2010

Last year, our founders came and gave a presentation called
“Ride the snake” where they
told you about our switch from lisp to python
and other fun stuff about reddit

RIDE THE SNAKE

THE REDDIT STORY

Part II

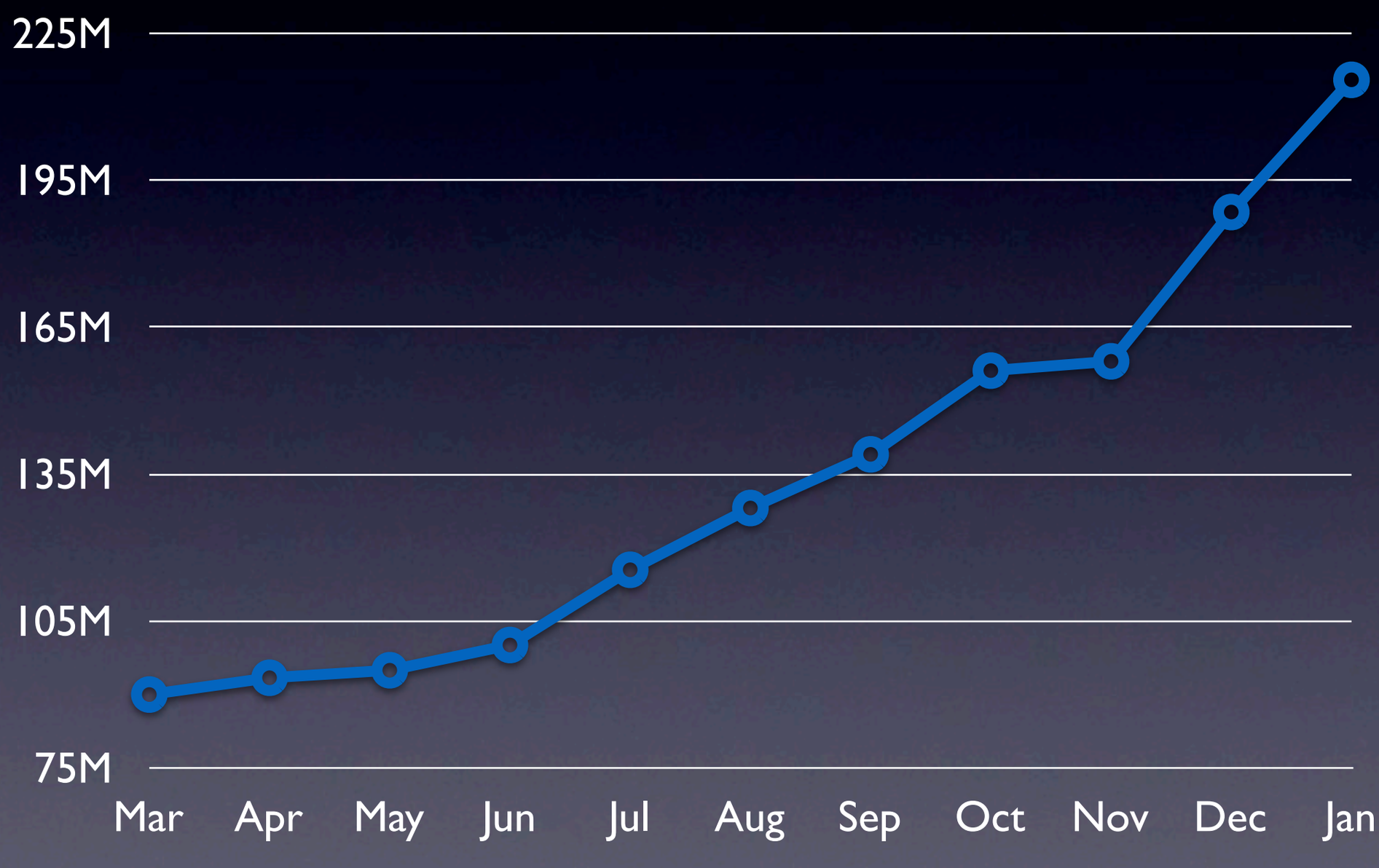
Stretching the Snake

Friday, February 19, 2010

This is part two of that presentation

called, “stretching the snake”

Monthly Page Views



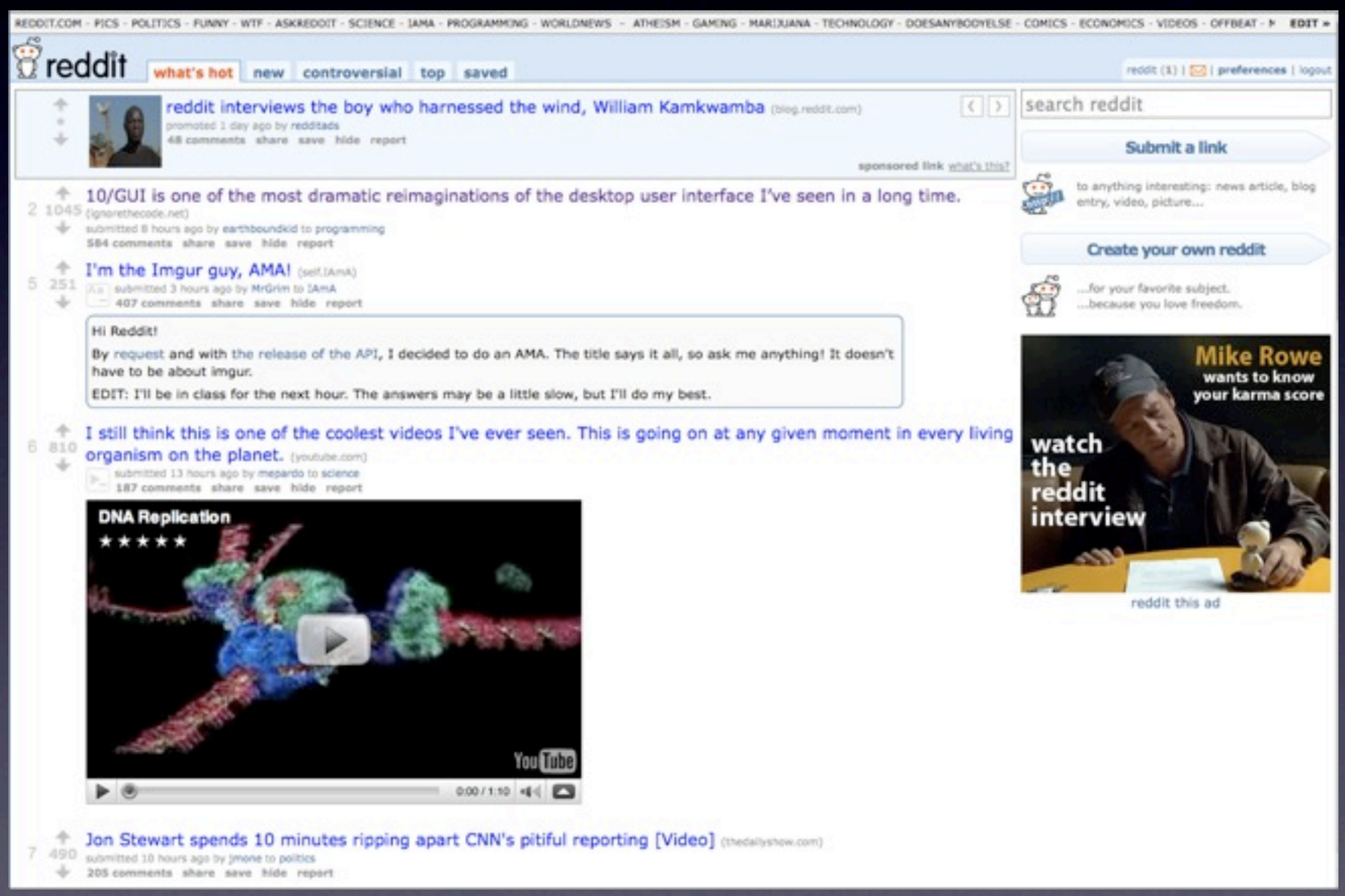
reddit

Friday, February 19, 2010

where I tell you about how we handle
more than double the traffic we did
a year ago

What is reddit?

reddit is an online community



reddit

Friday, February 19, 2010

How many of you use reddit?

How many have used it today?

reddit is a community where people come together
share and discuss interesting things on the internet
such as links to other stuff
or create their own content.

Timeline

April 2006 -- S3 for logos

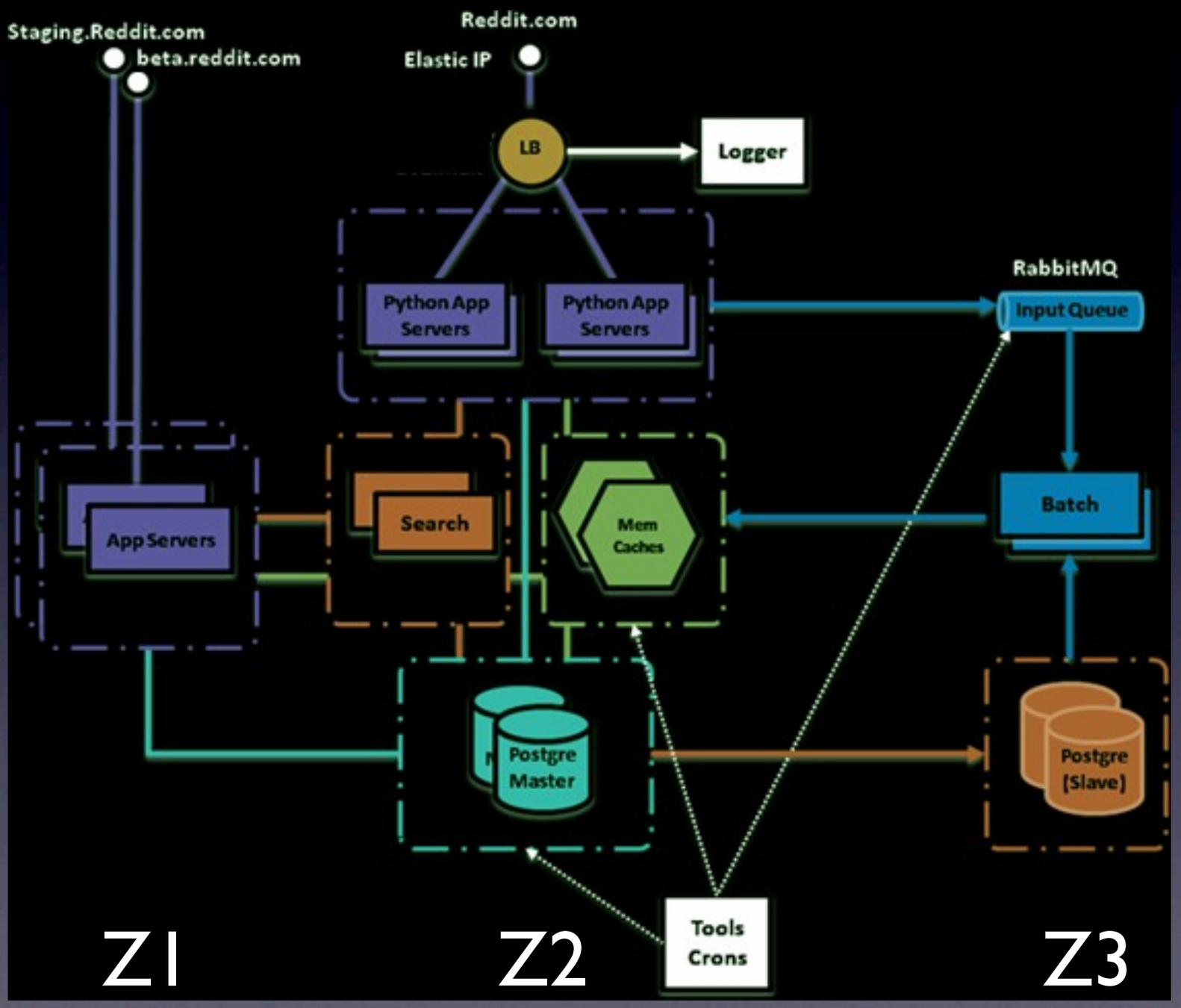
September 2007 -- S3 for thumbnails

November 2008 -- EC2 for batch processing

May 2009 -- EC2 for entire site

Been AWS user since almost the beginning

Architecture



multiple zones

db slaves are in different zones
from master for redundancy

beta/staging is elastic

app servers are elastic

Run 16 app processes per server (2 per virtual core)

Stats

- 256 Virtual CPUs
- 432GB of RAM
- 14TB of Elastic Block Storage
- 2TB of S3 Storage
- 6.5 TB / 3TB Data Out / In per month
- Over 2 comments per second at peak
- 220M+ Pageviews and just one sysadmin!

Friday, February 19, 2010

(Wait for display to finish)

256 Virtual CPUs

432 GB of RAM

220M+ page views

Cloud problems

Friday, February 19, 2010

So what kind of problems have we had in the cloud?

Threading and Virtual CPUs

- The security policies of a virtual server...
- Combined with the GIL's effect on CPUs...
- meant really bad performance for us

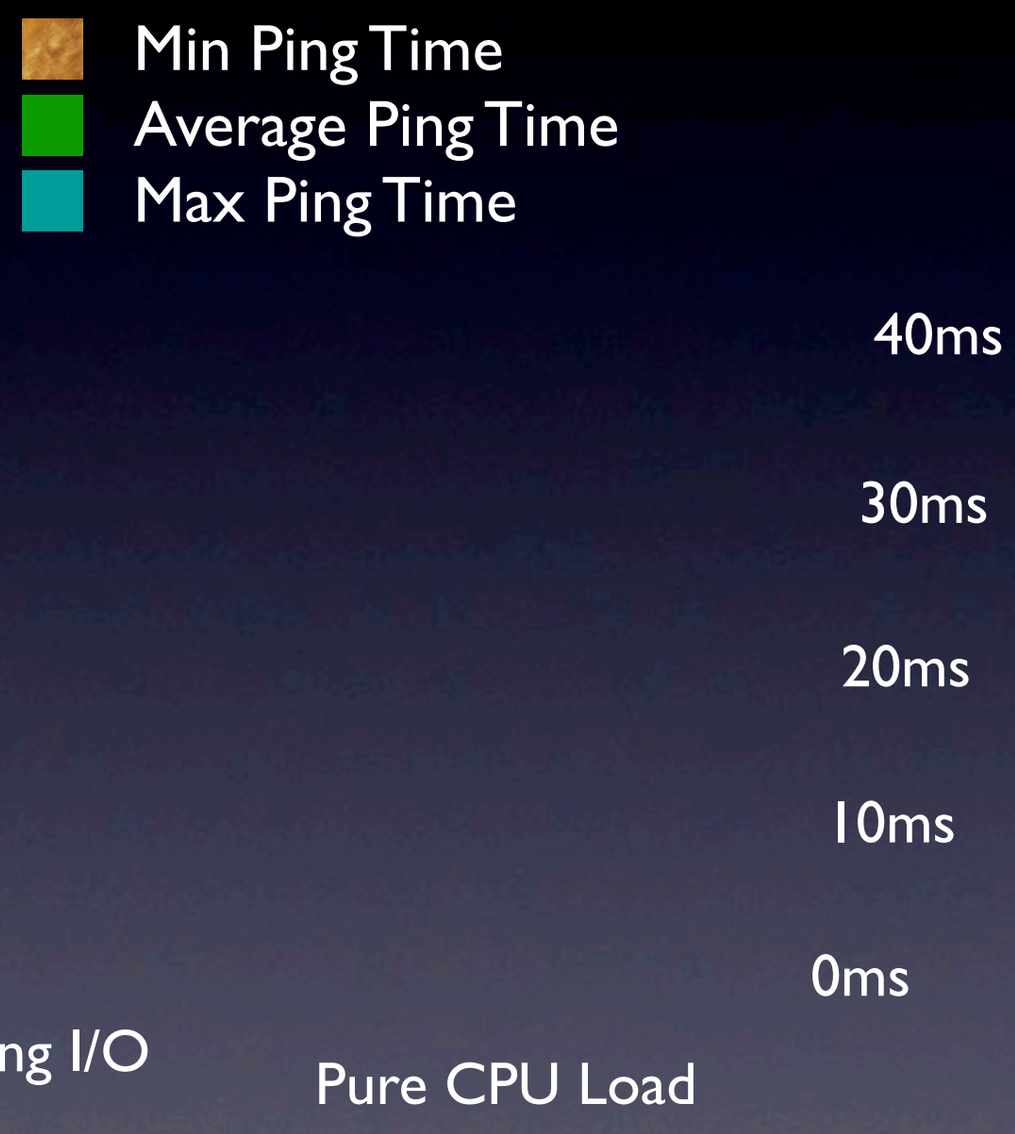
Friday, February 19, 2010

Since every packet has to go though the CPU for security checks, a loaded CPU slows down the network.

Slow network means more time spent switching between threads, loading the CPU even more.

Vicious cycle brings machine to its knees.

Latency vs. CPU load



reddit

Friday, February 19, 2010

Experiment description:

First did regular ping (left group)

Then made program that does a while 1 loop in 5 threads,
and 50% of the time it reads a file from disk into RAM and then dumps that object.

Ran 8 of those (1 per virtual core) while pinging. (middle group)

Then took out the I/O part (pure CPU while 1 loop). (Right group)

As you can see, a loaded CPU has a drastic effect
on network performance.

Pain for reddit

- Started by sending 5 requests to each application process.
- Server load would grow exponentially with traffic.
- Fixed it by changing load balancer to only send one request at a time.
- Let Linux divide up the CPU time, not Python.

Friday, February 19, 2010

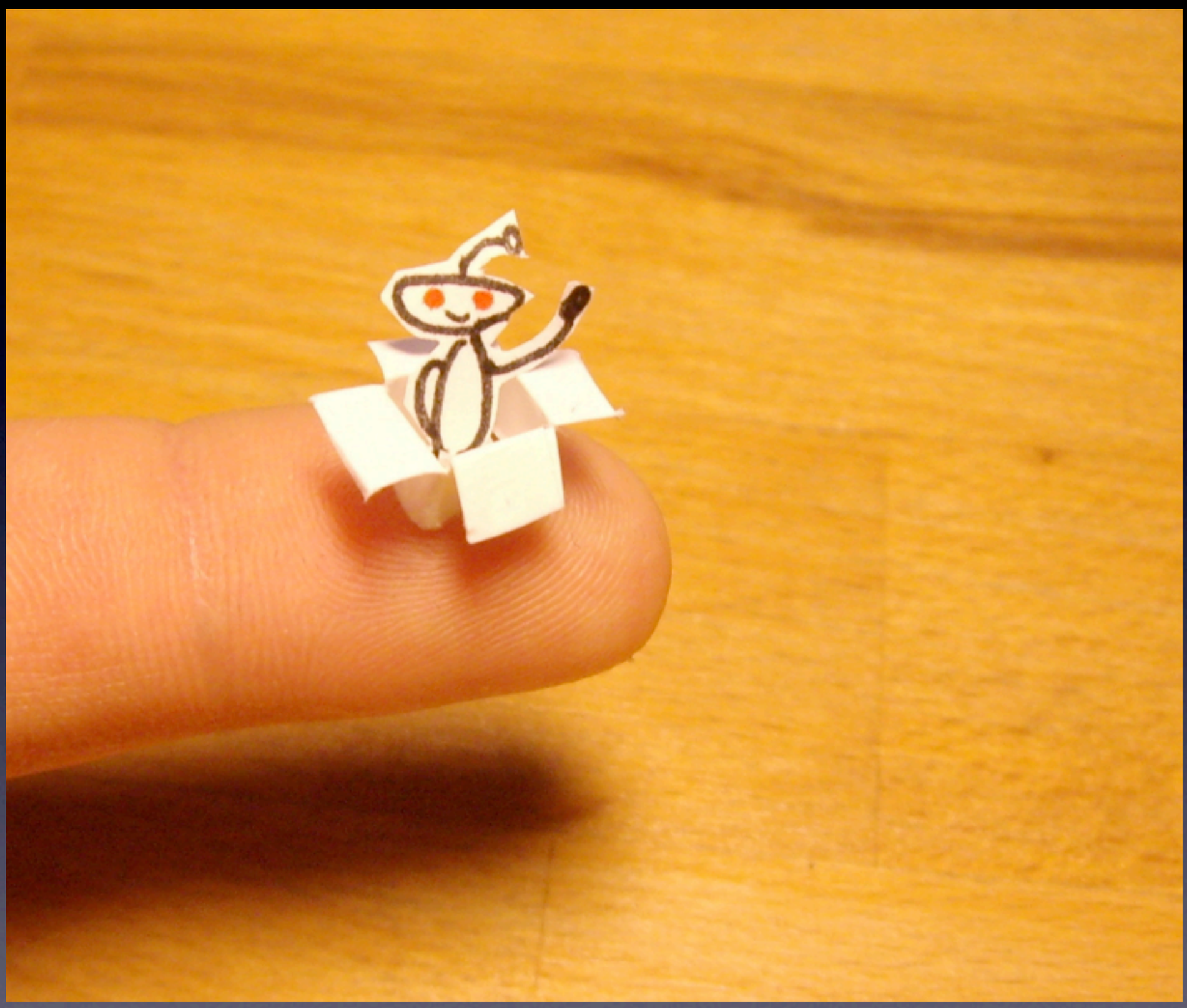
Step through one by one

When we first moved to EC2,
we were sending 5 requests to each
application process at once

We noticed that after the move,
server load would grow exponentially
with traffic

We solved the problem by changing the
load balancer to only send one request at
a time. Despite sending less requests
we got much better performance since
only one thread was active at a time

In other words, we let Linux tae care
of dividing the CPU time instead of Python.



Friday, February 19, 2010

An example of the fine work of the reddit community

Elastic Block Devices

EBS sometimes slows down a bit (and sometimes a lot)

Workaround: Use software RAID to minimize the effects of a slow disk

Downside: Can't take snapshots easily

Friday, February 19, 2010

(Step through)

Due to the redundant and network based nature,
sometimes the underlying drive has to remirror
or the network may be momentarily unavailable.

Use as many disks as reasonable.
We use 10 50GB disks for our database.
Amazon says failure rate is partly dependent on the size of the disk
so a raid with lots of small disks is in theory
less likely to fail

One downside to this strategy is that it makes it
almost impossible to utilize the built in snapshot feature
of the EBS.

Latency

Virtualized hardware with unknown physical proximity has inherent limits in latency.

Workaround: Fewer network calls, ask for more data at a time.

(step through)

There is an inherent latency in a virtual environment.

We had to rethink how we used memcached,
making less calls for more data at a time

A little pain is good

Fixing these issues made our app more reliable and highly available. We are better off than when we started.

(Activate words, then read them)

Overall, our experience with AWS and EC2 in particular has been good.

Scaling a Python App

Since EC2 is mostly like having your own hardware
The next part of this talk will be general scaling lessons learned by reddit

I'll point out some EC2 specifics as we go along.

1 > 2 > 3

Going from two to three is hard

Going from two

(keypress)

to three is hard

1 > 2 > 3

Going from one to two is harder

Friday, February 19, 2010

Going from one

(key press)

to two, is harder.

What do I mean by that?

anywhere you will need more than one of something
(application process, database, cache, queue, whatever)
It will be harder to go from one to two than two to three
and so on.

Especially relevant in a cloud setting, since getting more resources is so easy.

1 > 2 > 3

If possible, plan for 3 or more from the beginning.

(wait for animation)

If possible, plan for three or more from the beginning.

Sometimes your development cycle doesn't allow it
but at least keep it in mind.

Database Scaling with Sharding

reddit

Friday, February 19, 2010

And now some database scaling.

Sharding

- We split our writes across four master databases
- Links/Accounts/Subreddits, Comments, Votes and Misc
- Each has at least one slave
- We avoid reading from the master if possible
- Wrote our own database access layer, called the “thing” layer

Friday, February 19, 2010

We use 4 master databases

They are split up as Links/Accounts/Subreddit on the main db
Then separate db’s for the comments, votes, and everything else

Each has at least one slave -- The comments have 4 slaves

When they get busy, we add a slave. Thanks to EC2,
we don’t have to requisition servers!

Keep writes fast by not reading from the master when safe.

And lastly our data access layer,
the thing layer -- creative, isn’t it?
we wrote it because there was no good
database ORM at the time.

The thing layer

- Postgres is used like a key/value store
- Thing table has denormalized data
- Data table has arbitrary keys
- Lots of indexes tuned for our specific queries
- Thing and data tables are on the same box, but don't have to be

Friday, February 19, 2010

We use postgres like a key value store

our data access layer can be configured to find different tables in different places

this makes it somewhat easier to move data around based on load

We've never actually done that before because it still isn't straightforward.

Friday, February 19, 2010

Here’s a picture of the internet

We love memcache

We make heavy use of both memcached and memcachedb

reddit

Friday, February 19, 2010

Memcache is the heart of our scaling. We would be lost without it.

We have 30GB of cache, which is broken up like this:

(Next slide)

Caching

- Render cache (5GB)
  - Partially and fully rendered items
- Data cache (15GB)
  - Chunks of data from the database
- Permacache (10GB)
  - Precomputed queries

Friday, February 19, 2010

(step through)

render cache (5GB)
we store bits of html in here,
like the html for a single link in a listing
with holes for filling in the custom info
like the arrow direction and points,
as well as fully rendered pages for non-logged-in users

data cache (15GB)
any time we need data from the database,
we check the data cache first,
and if we don't find it, we put it there.
that means that the data for all of the popular items
will always be in the cache, which is great for a site
like ours, where certain data is usually popular for
a day, and then fall out of favor for newer data.
We also put memoized results in the data cache.

Permacache (10GB)
This is where we store the results of long database queries,
such as all of the listings, profile pages and such.

more on memcache and python in bit

Second class users

- Logged out users **always** get cached content.
- Akamai bears the brunt of our traffic
- Logged out users are about 80% of the traffic

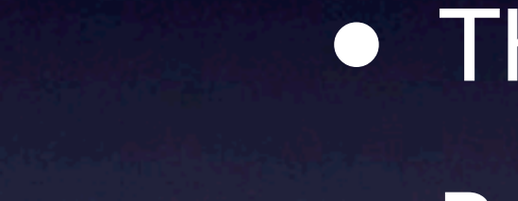
Since a logged out user isn't voting or commenting,
the pages looks the same for all of them, so we render
and cache the full page every 30 seconds.

Then akamai grabs it and caches it for 30 seconds as well.

akamai also accelerates the connection for our logged in users

Queues are your friend

- Votes
- Comments
- Thumbnail scraper
- Precomputed queries
- Spam
  - processing
  - corrections



Friday, February 19, 2010

Every vote generates a queue item for later processing. We just update the rendercache with the vote until it is processed. It also puts a job in the precomputer queue to recalculate that listing. That is why it sometimes seems like the vote totals aren't accurate. They are consistent in the database, it is just a rendering issue.

Every time a comment is written, it generates a job to have the comments tree for that page recalculated, which is then stored in the cache

Every time a link is submitted, it generates a job to go out and get a thumbnail
and it generates a job to recalculate the listing. It also generates a job for the spam filter.

When a moderator bans or unbans a link, it generates a job to
train the spam filter on that data.

Queues are extra important on EC2, because if you loose an instance
you don't have to worry about the lost work, as long as your queue is redundant.

C is faster than Python (sorry)

- filters
- discount (markdown)
- memcache (almost)

Friday, February 19, 2010

We use the ctypes library to interface to a couple of C libraries that make the site a lot faster.

(step through)

We've been using a c based whitespace filter for a long time.
in fact it had to be updated when we went from python 2.5 to 2.6
because the C interface changed

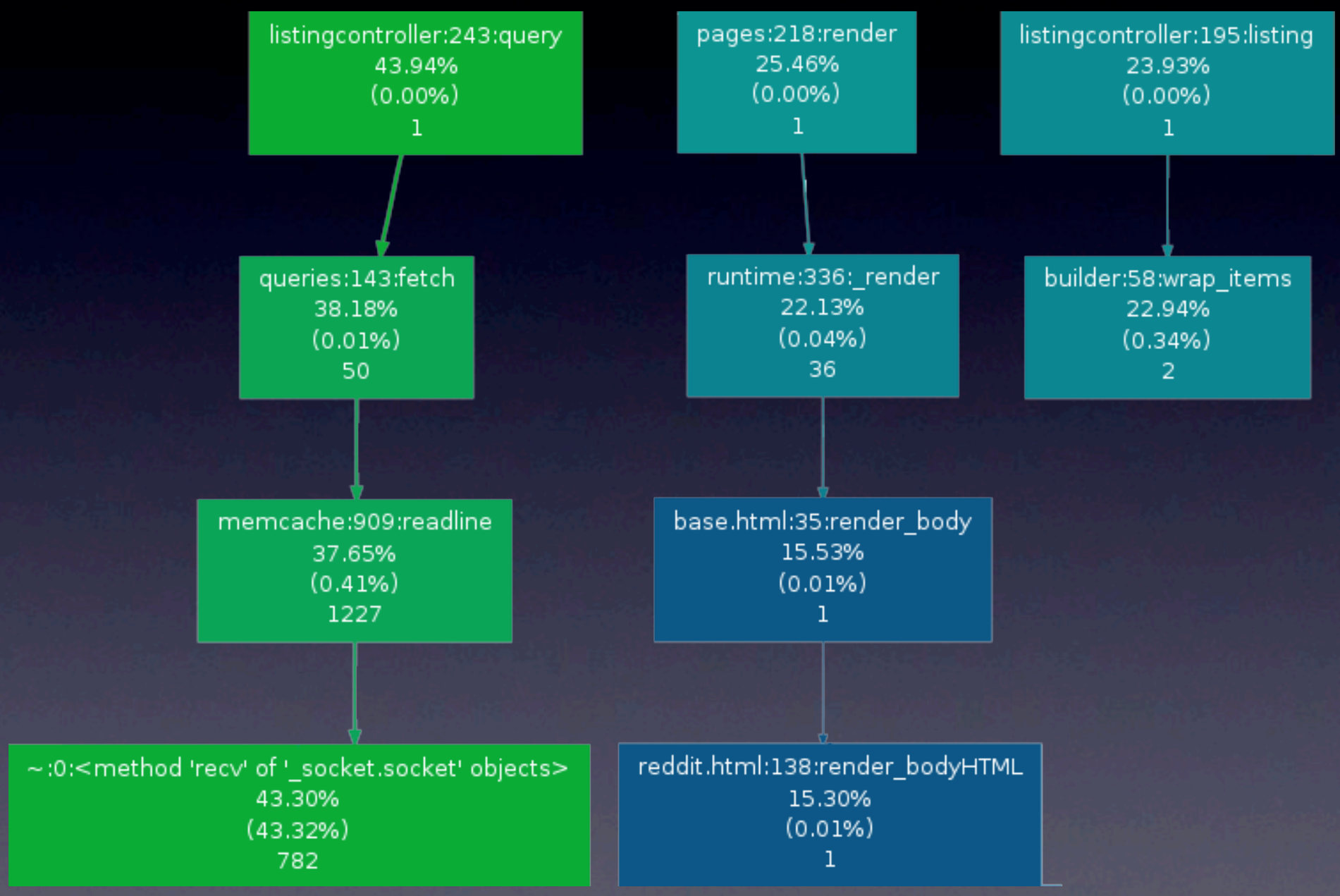
A few weeks ago we started using discount, which is a markdown library in C.
In testing, it was 15X faster than the equivalent python library, and had less edge cases.

In practice it was hard to see the benefit, because we rolled some other optimizations at the same time and traffic went up, but profiling showed a big improvement.

We're currently trying to move to C memcached library, which in testing has shown a huge improvement, and gives us access to the better binary protocol.

We've had some issues though related to thread safety and memory leaks, as well as places in our code where we had been lazy because the python library was more forgiving.

A silhouette of Profiling



Friday, February 19, 2010

We do a lot of profiling to help identify, and hopefully fix, bottlenecks.

Here is a piece of our profile graph

The key takeaway here is that we spend almost 1/2
our time on socket receive for memcache
which is why we are changing it out for the
C library. This is an effect of the higher network
latency in the virtualized environment.

A lot of the rest of the time is spent on rendering.

Why not Django?

- It was in early alpha at the time
- Support was hard to find
- Rendering, especially in a loop, was slow
- The sandbox was hampering

reddit

Friday, February 19, 2010

People ask us why we didn't (or don't) use Django.

When we started, it was still fairly alpha.
We actually tried using it before Pylons, but it was
hard to get support, especially for our biggest problem
which was that rendering was very slow, especially when you use nested templates like we do.

Also, the sandbox was hampering.
As my teammate Chris said,
"I shouldn't have to trick the templating engine into doing what I want."

You must construct additional Pylons

Friday, February 19, 2010

So we chose Pylons instead

What is Pylons?

- routing
- paste
- wsgi
- mako
- sqlalchemy
- c and g

It's really just some glue code between these pieces.

So what does it mean to "scale Pylons"?

Scaling Pylons

- pylons scaling == python scaling
- run lots of appservers and make them independent of each other
- We built our own caching
- We built our own database layer

Friday, February 19, 2010

Scaling pylons is the same as scaling python:
use lots of independent app servers and caching

There really isn't anything special about scaling pylons vs. any other framework or any other python application in general.

we built our own caching layer because it didn't really exist in pylons when we started

we built our own db access layer with raw sql via sqlalchemy because
the sqlalchemy ORM wasn't quite ready
for prime time when we started

Would we use Pylons again?

Yes

Maybe replace paste

(even Ian isn't using Paste anymore)

(click again)

In short, yes.

(wait for animation)

We might replace paste if we did,
since it causes some problems
that are hard to fix because they are so deep in the code
like setting cookies twice
and not enforcing limits on uploads
Also, we had to write our own "paster run",
which we felt it should have to go with "paster shell"

Key Takeaways

- EC2 makes things easier, but isn't a magic bullet.
- The higher network latency will be problematic -- expect to work around it.
- Scaling on EC2 is a lot like anywhere else, but you need to be more disciplined.

Friday, February 19, 2010

EC2 isn't a magic bullet --
you still have to configure the servers correctly
and write your apps correctly
your site won't magically scale by putting it on ec2

The higher network latency will effect you everywhere.
It could surprise you if you are coming from your own hardware.

The principals are the same, but you need to be more disciplined,
especially about where you put your data and how you back it up

Don’t change your API
(Or at least change your major version number)

- sqlalchemy
- beautiful soup
- routes
- pylons

Friday, February 19, 2010

A short rant while I’ve got you here

Please stop making major changes to your API
without making major changes to your version numbers

Here are the packages that we are currently holding back
because it would be difficult to upgrade them
due to an API change

(click next)

don’t get me wrong, we love these packages and the work their maintainers do
we just wish you would make the version number changes reflect the underlying changes

Just a quick reminder...

reddit is open source

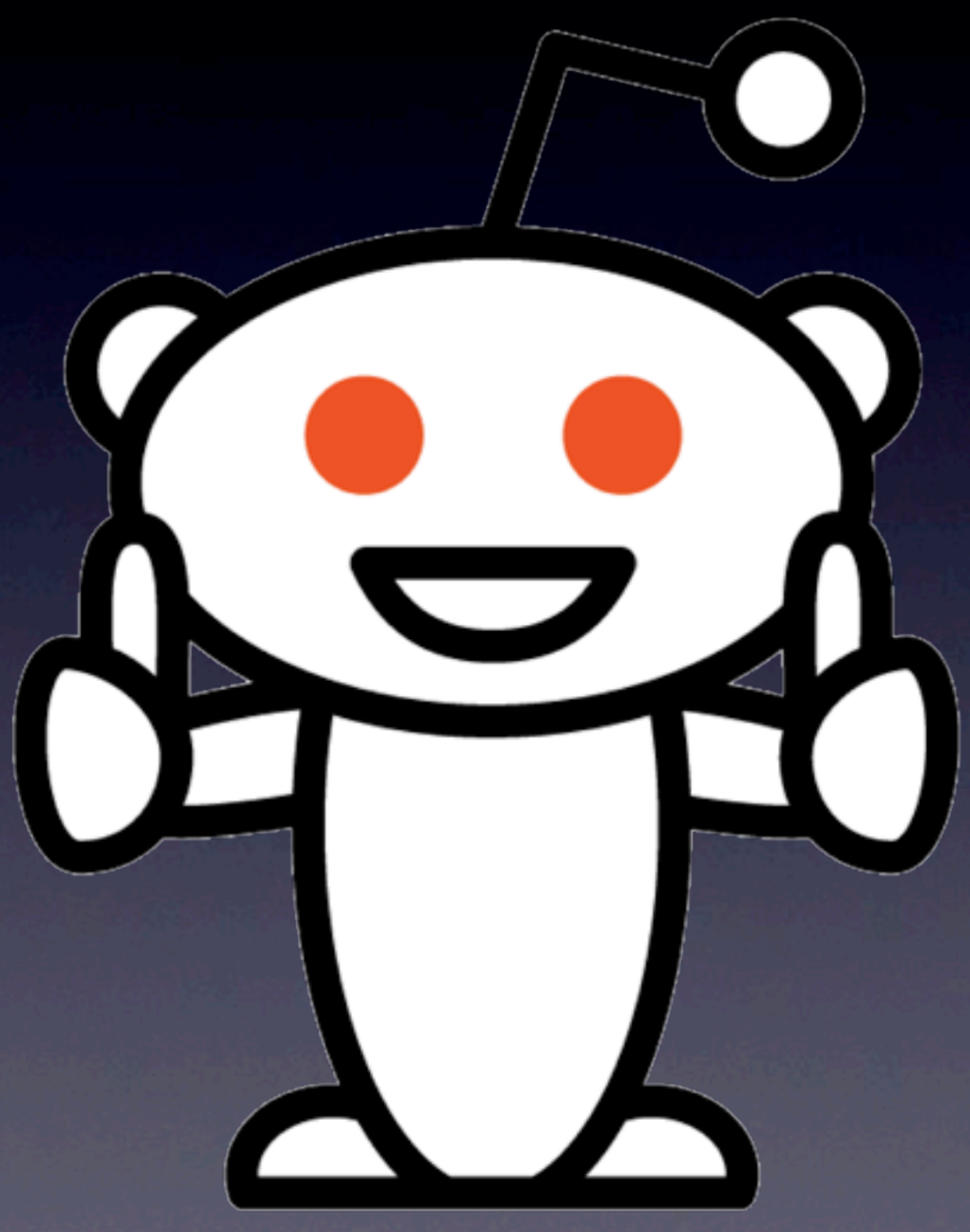
http://code.reddit.com

patches are now being accepted!

and also...

Available at the Breadpig booth outside. All proceeds go to charity

Questions?



Thanks for listening!

If anyone has any questions

(next slide)

Getting in touch

Email: jedberg@reddit.com

reddit: www.reddit.com/user/jedberg

Twitter: @jedberg

Web: www.edberg.org/jeremy

Facebook: facebook.com/jedberg

Linkedin: www.linkedin.com/in/jedberg

You can contact me in one of these ways,
 or ask your question now.

thank you.