Scalable, Good, Cheap

a tale of sexiness, puppets, shell scripts, and python
From this...
...to this!
Get your infrastructure started right!

(not just preparing for incident and rapid event response)
Who we are?

Avleen Vig (@avleen)
- Senior Systems Engineer at Etsy
- Good at: Scaling frontends, python
- Previous companies: WooMe, Google, Earthlink

Marc Cluet (@lynxman)
- Senior Systems Engineer at WooMe
- Good at: Backend scaling, bash/python, languages
- Previous companies: RTFX, Tiscali, World Online
Overview

- Workflow
- Why planning for scaling is important
- How do you choose your software
- Setting up your infrastructure
- Managing your infrastructure
The background

- Larger startup, $32m in funding
- 6 million+ active users
- Dozens of developers
- 6 systems administrators
- 4 DBAs
- 10+ code releases every day
- Geographically distributed employees
  - Brooklyn HQ
  - Satellites in Berlin, San Francisco
  - Small number of remote employees
The background

- Small, funded start up
- 6 python developers
- 2 front end developers
- 3 systems administrators
- 1 DBA (moustache included)
- Multiple code releases every day
- Geographically distributed employees
Workflow

- Ticket systems
  - Ticket, or it didn't happen!

- Documentation
  - Wikis are good

- Don't Repeat Yourself
  - If you keep doing the same thing manually, automate

- Version control everything
  - All of your scripts
  - All of your configurations
Workflow

- Everything will change

- Technical debt vs Premature optimisation
  - If you try to be too accurate too early, you'll fail
Team integration

- Be sure to hire the right people
  - Beer recruitment interview

- Encourage speed
  - Release soon and release often

- Embrace mistakes as part of your day to day
  - Learn to work with it

- Ask for peer reviews for important components
  - Helps sanity checking your logic

- Developers, Sysadmins, DBAs, *one team*
Team communication

- Team communication is the most critical factor
- Make sure everyone is in the loop
- Useful applications
  - IRC
  - Skype
  - email
  - shout!
- Don't be afraid to use the phone to avoid miscommunication
Layering! Not just for haircuts.

Separate your systems

- Front end
- Application
- Database
- Caching
Choosing your software

- What does your software need to do?
  - FastCGI / HTTP proxy? Use nginx
  - PHP processing? Use apache

- What expertise do you already have?
  - Stick to what you're 100% good at

- Don't rewrite everything
  - If it does 70% of what you need it's good for you
Release management

- Fast and furious
- Automate, automate, automate
- Script your deploys and rollbacks
- Continuous deployment
- MTTR vs MTBF
MTTR vs MTBF
Maintainability

MTTR Optimized  Versus  MTBF Optimized

More info here: http://ti.arc.nasa.gov/projects/ishem/Papers/ONEill_Maintainability.doc
Logging

- Centralize your logging
  - syslog-ng
- Parsing web logs - the secret troubleshooting weapon
  - SQL
  - Splunk
CREATE TABLE access (  
ip inet,  
hostname text,  
username text,  
date timestamp without time zone,  
method text,  
path text,  
protocol text,  
status integer,  
size integer,  
referrer text,  
useragent text,  
clienttime double precision,  
backendtime double precision,  
backendip inet,  
backendport integer,  
backendstatus integer,  
ssl_cipher text,  
ssl_protocol text,  
scheme text
);
Web logs in a database!

**Number of 504 errors from all instances**

![Graph showing number of 504 errors from all instances over time]

- **5 min error level**
- **Graph title:** Number of 504 errors from all instances - by day
- **Error level from all hosts**
- **Cur:** 227.65
- **Min:** 19.72
- **Avg:** 568.73
- **Max:** 11.07k
- **Last update:** Thu Nov 11 23:05:15 2010
Monitoring

- Alerting vs Trend analysis
Monitoring

- Alerting vs Trend analysis
  - Nagios is great for raising alerts on problems
This host is up and running.

### Time and String Metrics

<table>
<thead>
<tr>
<th>Metric</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Last Boot Time</td>
<td>Tue, 31 Aug 2010 16:36:59 -0700</td>
</tr>
<tr>
<td>Gexec Status</td>
<td>OFF</td>
</tr>
<tr>
<td>Gmond Started</td>
<td>Mon, 06 Sep 2010 12:16:42 -0700</td>
</tr>
<tr>
<td>Last Reported</td>
<td>0 days, 0:00:03</td>
</tr>
<tr>
<td>Machine Type</td>
<td>x86</td>
</tr>
<tr>
<td>Operating System</td>
<td>Linux</td>
</tr>
<tr>
<td>Operating System Release</td>
<td>2.6.26-2-xen-686</td>
</tr>
<tr>
<td>Uptime</td>
<td>72 days, 1:08:55</td>
</tr>
</tbody>
</table>

### Constant Metrics

<table>
<thead>
<tr>
<th>Metric</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU Count</td>
<td>2 CPUs</td>
</tr>
<tr>
<td>CPU Speed</td>
<td>2795 MHz</td>
</tr>
<tr>
<td>Memory Total</td>
<td>917700 KB</td>
</tr>
<tr>
<td>Swap Space Total</td>
<td>0 KB</td>
</tr>
</tbody>
</table>
Monitoring

- Alerting vs Trend analysis
  - Nagios is great for raising alerts on problems
  - Ganglia is great at long term trend analysis
  - Know when something is out of the "ordinary"
Monitoring

- Alerting vs Trend analysis
  - Nagios is great for raising alerts on problems
  - Ganglia is great at long term trend analysis
  - Know when something is out of the "ordinary"

- What should you monitor?
  - Anything which breaks once
  - Customer facing services
Monitoring

- Alerting vs Trend analysis
  - Nagios is great for raising alerts on problems
  - Ganglia is great at long term trend analysis
  - Know when something is out of the "ordinary"

- What should you graph?
  - Everything! If it moves, graph it.
  - Customer facing rates and statistics
Monitoring

Get statistics from your logs:

- PostgreSQL: pgfouine
- MySQL: mk-query-digest
- Web servers: webalizer, awstats, urchin
- Custom applications: Do it yourself! Integrate with Ganglia
Monitoring

CPU usage - by year

- **system**: Cur: 23.95, Min: 2.11, Avg: 38.53, Max: 1094.64
- **user**: Cur: 165.48, Min: 13.04, Avg: 309.72, Max: 1466.83
- **nice**: Cur: 0.00, Min: 0.00, Avg: 0.00, Max: 0.00
- **idle**: Cur: 1301.55, Min: 53.49, Avg: 1171.19, Max: 1740.86
- **iowait**: Cur: 154.39, Min: 0.05, Avg: 136.57, Max: 909.52
- **irq**: Cur: 0.52, Min: 0.00, Avg: 0.71, Max: 1.55
- **softirq**: Cur: 8.44, Min: 0.04, Avg: 12.20, Max: 42.94
Caching

- Caches are disposable
Caching

- Caches are disposable
- But what about the thundering herd?
The importance of scaling
The importance of scaling

- August 2003 Northeastern US and Canada blackout
  - Caused by poor process execution
  - Lack of good monitoring
  - Poor scaling
The importance of scaling
The importance of scaling

- Massive destruction avoided!
  - 256 power stations automatically shut down
  - 85% after disconnecting from the grid
  - Power lost but plants saved!
Caching

- Caches are disposable
- But what about the thundering herd?
  - Increase backend capacity along with cache capacity
  - Plan for cache failure
  - Reduce demand when cache fails
Caching

- Find out how your caching software works
  - Memcache + peep!
  - Is it better with lots of keys and small objects?
  - Or fewer keys and large objects?
  - How is memory allocated?
Caching

- Caches are disposable
  - Solved!
- But what about the thundering herd?
  - Solved!
- Now we get into database scaling!
  - Over to Marc...
Databases

Databases...

or how to live and die dangerously
Databases

SQL or NoSQL?
Databases

- **SQL**
  - Gives you transactional consistency
  - Good known system
  - Hard to scale

- **NoSQL**
  - Transactionally consistent "eventually"
  - New cool system
  - Easy to scale
Databases

- **SQL**
  - Gives you transactional consistency
  - Good known system
  - Hard to scale

- **NoSQL**
  - Transactionally consistent "eventually"
  - New cool system
  - Easy to scale

You may end up using BOTH!
Databases

- Be smart about your table design
Databases

- Be smart about your table design
  - Keep it simple but modular to avoid surprises
You need to design your database right!
Databases

- Be smart about your table design
  - Keep it simple but modular to avoid surprises
  - Don't abuse many-to-many tables, they will just give you hell
Databases

• Be smart about your table design
  o Keep it simple but modular to avoid surprises
  o Don't abuse many-to-many tables, they will just give you hell

• YOU WILL GET IT WRONG
  o You'll need to redesign parts of your DB semi-regularly
  o Be prepared for the unexpected
Databases

The read dilemma

- As the tables grow so do read times and memory. Several options:
  - Check your slow query log, tune indexes
  - Partition to read smaller numbers of rows
  - Master / Slave, but this adds replication lag!
Databases

The read dilemma

- As the tables grow so do read times and memory.
  Several options:
    - Check your slow query log, tune indexes
      - Single most common problem with slow queries and capacity
      - Be careful about foreign keys
Databases

The read dilemma

- As the tables grow so do read times and memory. Several options:
  - Check your slow query log, tune indexes
  - Partition to read smaller numbers of rows
    - By range (date, id)
    - By hash (usernames)
    - By anything you can imagine!
Databases

The write conundrum

- As the database grows so do writes
- Writes are bound by disk I/O
  - RAID1+0 helps
- Don't shoot yourself in the foot!
  - Don't try to solve this early
  - Have monitoring ready to foresee this issue
  - Bring pizza
Databases

Divide writes!

- Remember about modular? This is it
Databases

How to give a consistent view to the servers?

Use a query director!

- pgbouncer on PostgreSQL
- gizzard on MySQL
Web frontend

- Hardware load balancers - Good but expensive!
- Software load balancers - Good and cheap! (more pizza)

  - Web server frontends
    - nginx, lighttpd, apache

  - Reverse proxies
    - varnish, squid

  - Kernel stuff
    - Linux ipvs
Web frontend

Which way should I go?

- Web servers as load balancers
  - Gives you nice add on features
  - You can offload some process in the frontend
  - Buffering problems

- Reverse proxies
  - Caching stuff is good
  - Fast reaction time
  - No buffering problems
Web frontend

Divide your web clusters!

- You can send different requests to different clusters
- You can use an API call to connect between them
Configuration management

- Be ready to mass scale
  - Keep all your machines in line

- Automated server installs
  - Use it to install new software
  - Also to rapidly deploy new versions
Writing tools

- If you do something more than 2 times it's worth scripting

- Write small tools when you need them

- Stick to one or two languages
  - And be good at them
Writing tools

- Even better
- Have your scripts repo in a cvs and push it everywhere
Backups

- It's important to have backups
Backups

- It's important to have backups
- It's even more important to exercise them!
  - Having backups without testing recovery is like having no backups
Backups

- It's important to have backups
- It's even more important to exercise them!
  - Having backups without testing recovery is like having no backups
- How can we exercise backups for cheap?
Backups

● It's important to have backups

● It's even more important to exercise them!
  ○ Having backups without testing recovery is like having no backups

● How can we exercise backups for cheap?
  ○ Cloud computing!
Cloud computing

- Cloud computing help us recreate our platform on the cloud
- Giving us a more than credible recovery scenario
- Also very useful to spawn more instances if we run into problems
Interesting things to read

Wikipedia
  • http://en.wikipedia.org/wiki/DevOps

Web Operations and Capacity Planning
  • http://kitchensoap.com

High scalability (if you get there)
  • http://highscalability.com/

If you really fancy databases, explain extended
  • http://explainextended.com/
Questions?

Work at Etsy!
http://etsy.com/jobs

Work at WooMe!

Etsy

@avleen

WooMe

@lynxman