The Future of Microblogging

Jack Magiera, Jon Richelsen
Idea

The human race **needs** a way to share every aspect of one’s life at all times.

If you’re not sharing your pictures and words with the entire world, then what’s even the point?

Twitter is good, but not good enough. Tweets are wordy and cumbersome.
Problem

Need a **simple, distributed, scalable** system to allow for constant microblogging and social validation.

Sharing pictures and text **at all times**

Deliver an infrastructure that can manage **high traffic usage** and scale to **large numbers of users**
Koolack

Microblogging all the time, even right now.
Koolack

Microblogging service - Twitter, but with dynamics and synergy

Users post Kools, can follow/unfollow other users, and can “ack” others’ Kools
Unscaled Infrastructure
Detail of Unscaled Infrastructure

Incoming request

EC2

httpd

GET, POST

django

disk

static

media

sqlite3
Scaled Infrastructure
# Django ORM with NoSQL?

<table>
<thead>
<tr>
<th>SQL</th>
<th>Django API</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT * FROM kools WHERE id = 12343</td>
<td>my_kool = Kool.objects.get(id=12343)</td>
</tr>
<tr>
<td>SELECT author FROM kools WHERE id = 12343</td>
<td>my_author = my_kool.author</td>
</tr>
<tr>
<td>UPDATE kools SET content = ‘updated content’ WHERE id = 12343</td>
<td>my_kool.content = ‘updated content’</td>
</tr>
<tr>
<td></td>
<td>my_kool.save()</td>
</tr>
<tr>
<td>SELECT * FROM kools, follows WHERE kools.author = follows.followee AND follows.follower = ‘username’</td>
<td>Kool.objects.filter(author__followed_by=my_author)</td>
</tr>
</tbody>
</table>
**Attack methodology**

- Use PhantomJS to test high traffic
- Create scripts that automate **standard user processes**
- Randomize elements of these scripts and run them simultaneously and/or subsequently
- Use internal testing features to measure performance on scaled vs unscaled
Performance Metrics

- client-side performance or server-side performance?
- Elastic Load Balancing + Amazon CloudWatch = metrics!
- req/s measures overall performance
- increase number of clients until req/s reaches limit
- Elastic Beanstalk to add more EC2
One more thing...

Future plans

Neaten up interface - aesthetic and functional purpose

Finalize scaling

Complete testing