References


Cloud computing news
EC2 Announcements

No longer in Beta

Windows Machine images in beta

Upcoming features

Load Balancing
Automatically balance requests among EC2 instances

Auto-scaling
Grow and shrink EC2 capacity to meet your demand

Monitoring
Realtime monitoring of your EC2 usage

Management Console
Web interface configure, manage & access AWS

Late next year Microsoft will make available Azure, a cloud OS

http://www.microsoft.com/azure/default.mspx
Scaling Web Applications

http://highscalability.com/
Planing for Scaling

ORM for Data Partitioning and Query Splitting
Monitoring process, resources, and uptime
Performance Testing and Capacity Planning
Static vs. Dynamic Content splitting
Bundling and Compressing JS and CSS
Logging
Pragmatic Caching
Functional Decomposition
Deployment
Asynchronous Practices
Lean Applications

ORM for Data Partitioning and Query Splitting

Split queries between updates and deletes from the start

ORM - Object Relational Mapping

Maps between objects and database tables
ORM Example - Tables

students

<table>
<thead>
<tr>
<th>firstname</th>
<th>lastname</th>
<th>phone</th>
<th>code_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>Smith</td>
<td>555-9876</td>
<td>2000</td>
</tr>
<tr>
<td>Ben</td>
<td>Oker</td>
<td>555-1212</td>
<td>9500</td>
</tr>
<tr>
<td>Mary</td>
<td>Jones</td>
<td>555-3412</td>
<td>9900</td>
</tr>
</tbody>
</table>

codes

<table>
<thead>
<tr>
<th>code</th>
<th>major</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>Art</td>
</tr>
<tr>
<td>3000</td>
<td>History</td>
</tr>
<tr>
<td>9500</td>
<td>Electrical engineering</td>
</tr>
<tr>
<td>9900</td>
<td>Computer Science</td>
</tr>
</tbody>
</table>
ORM Example - SQL

INSERT
    INTO students (firstname, lastname, phone, code_id)
SELECT
    'Roger' AS firstname,
    'Whitney' AS lastname,
    '594-3535' AS phone,
    codes.code AS code_id
FROM
    codes
WHERE
    codes.major = 'Art'
ORM Example - Using Glorp

person := Person first: 'Roger' last: 'Whitney'.
person phone: '594-3535'.
person major: 'Art'.
session beginUnitOfWork.
session register: person.
session commitUnitOfWork.
Monitoring process, resources, and uptime

Process Monitoring
This makes sure things are running
Example tools
God, Monit, SMF

Resource Monitoring
Monitor CPU, Memory, Disk Space, Networking, etc.
Example tools
Nagios, Ganglia, Munin, ZenOSS

UpTime Monitoring
Example Tools
webmetrics and pingdom
Performance Testing and Capacity Planning

How does your application perform under load

What sort of load do you expect
Static vs. Dynamic Content splitting

Web frameworks have overhead

Use web server for static pages

Web framework handles only dynamic pages
Bundling and Compressing JS and CSS

Bundle and compress javascript and CSS files in web application

Bundling reduces number of TCP connects needed

Compressing reduces total size sent
Logging

Log your application

access
request information
errors
problems

Monitor logs
Pragmatic Caching

Web frameworks support caching of pages/data

Understand and use caching available
Functional Decomposition

Decompose application into separate functions

- app servers, monitoring, log aggregation,
  databases, message queues

As required use separate machines for each function
Asynchronous Practices

When possible make functions asynchronous

Does logging have to be done in real time?

Send logging info to logger
Application continues
Logger queues requests

Beware of the CAP theorem
CAP Theorem

You can only have two of the three CAP properties at the same time

Consistency
All clients see the same view even with updates

Availability
All clients can find some replica of the data

Partition-tolerance
The system properties hold even when the system is partitioned

Deployment

Automate your deployment

Have roll back capability
Amazon Simple Queue Service
Message Queues

Unlimited queues and messages
   Each queue is named

Messages up to 8 KB in size

Multiple readers & writers allowed per queue

Redundant infrastructure
   Guarantees delivery of message at least once
Access to Queue

Via Soap or REST

From any location
Best effort made to keep messages in order

If exact order is important add sequencing information to message

Not easy
  Multiple machine can write
  Timestamps between machine not synched
At-Least-Once Delivery

Remember the CAP theorem

Messages are stored on multiple machines

A delete may not delete all copies of a message
   If a server is down during the delete
   You may get the message a second time

Your application has to handle duplicate messages
Message Sample
CAP again

When you query for messages

A subset of the SQS servers are queried

May get no messages when messages exist
  When have small number of messages (under 1000)

May not have access to all messages
Visibility Timeout

Reading a message does not delete the message
   The reader could die before processing the message

Reader has to delete message when done processing

After being read a message is invisible in the queue

Message becomes available after the visibility timeout ends
   Unless the reader deletes the message
Operations

CreateQueue
ListQueue
GetQueueAttributes
SetQueueAttributes
DeleteQueue

SendMessage
ReveiveMessage
DeleteMessage
Queue Attributes

Visibility Timeout