

CS 188/219

Scalable Internet Services

Andrew Mutz
April 4, 2015



Today's Agenda

Motivation

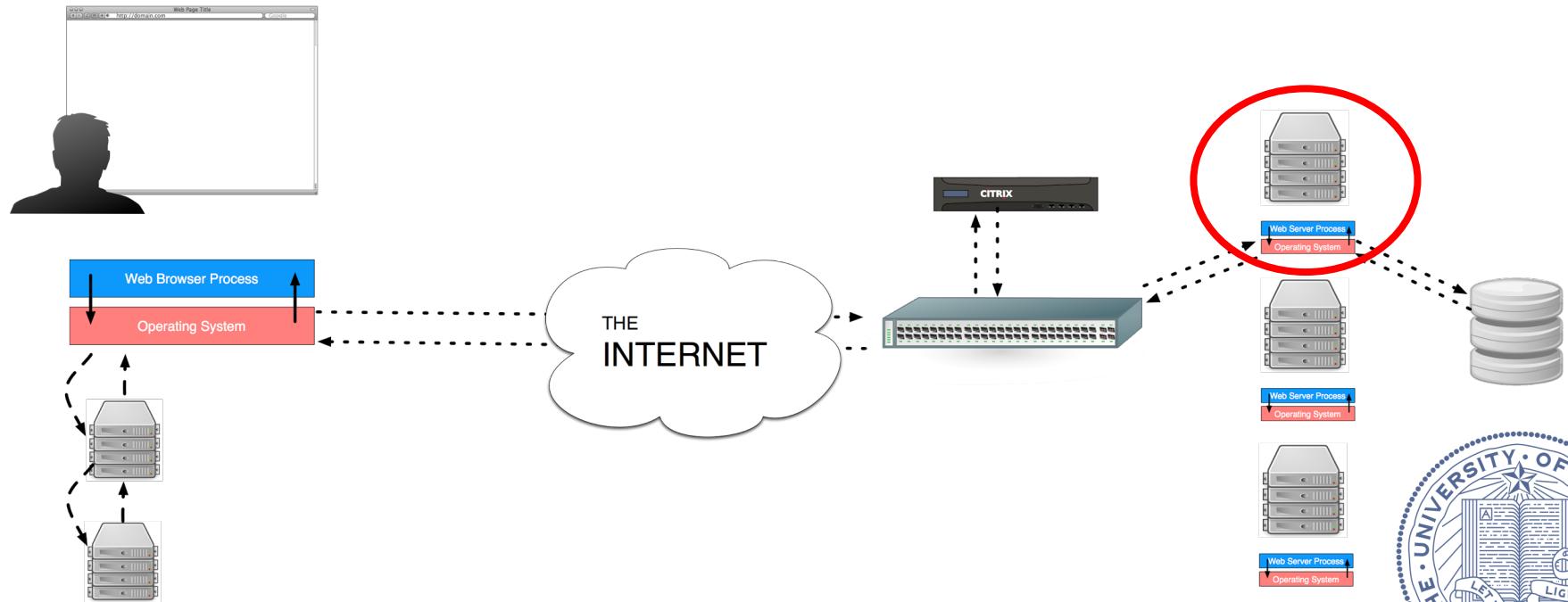
Server-side Caching

Deploying on AWS

For Next Time



Server-side Caching



Motivation

After today you should understand

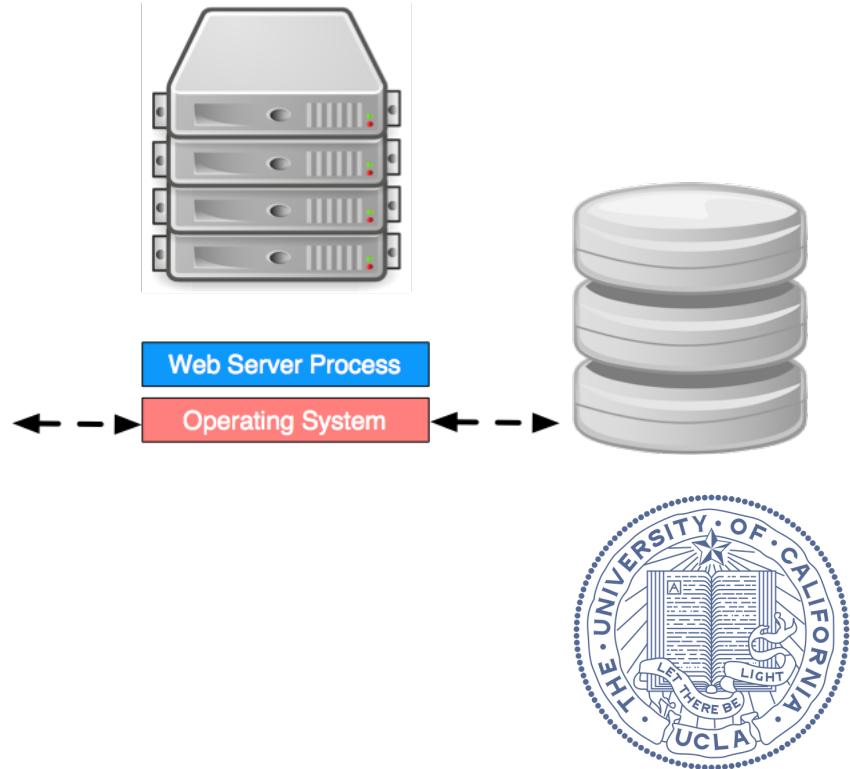
- Why server side caching exists
- What options you have when using server side caching
- How to use this in your projects
- How to deploy on AWS using CloudFormation



Server-side Caching

We have a web server process that is repeatedly responding to HTTP requests from a variety of clients.

Responding to each request requires computation and I/O to be performed, and this can be expensive.

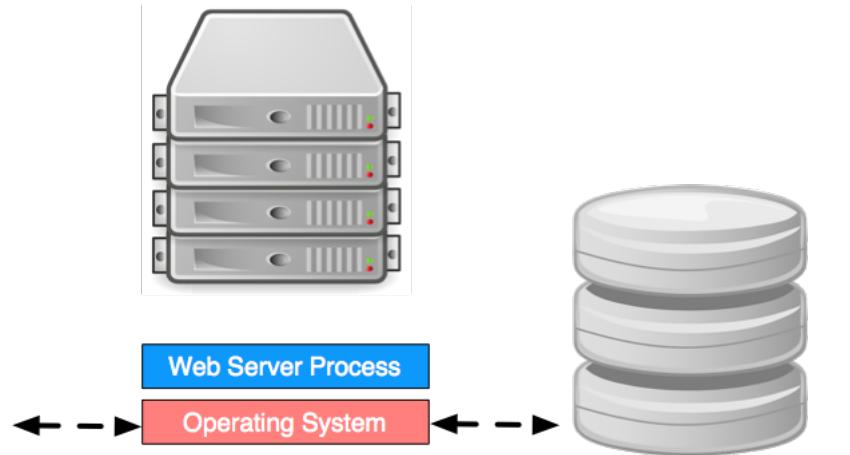


Server-side Caching

In practice, there is a great deal of similarity between responses.

In the last lecture (HTTP Caching) we looked at optimizing scenarios where repeated responses are identical.

In this lecture we will look at optimizing scenarios where repeated responses are not identical, but are similar.

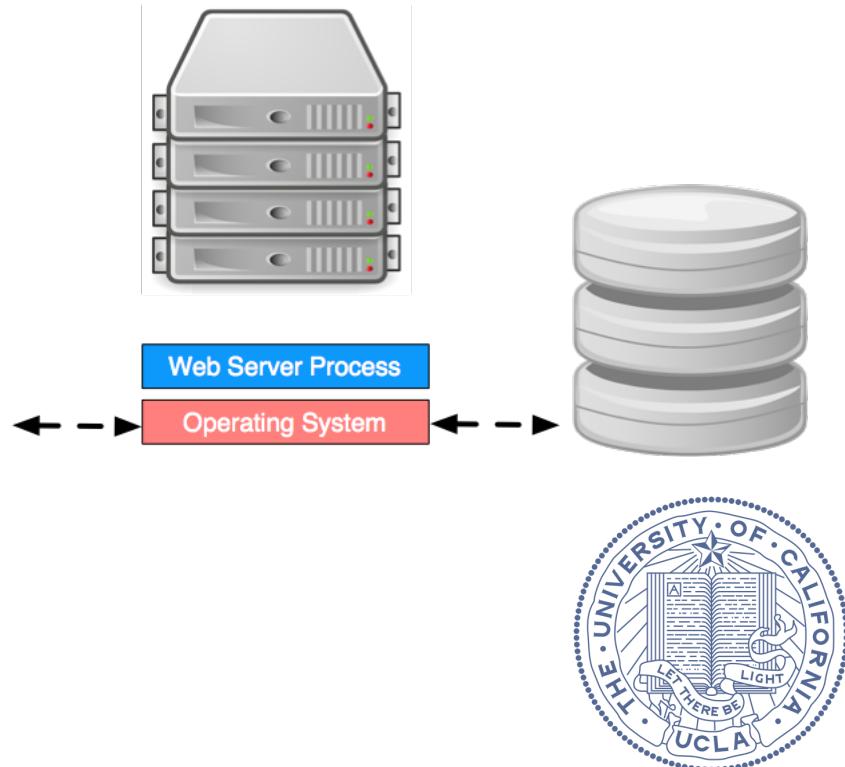


Server-side Caching

There are many parts of a response that are similar.

There are many steps to creating a response that are repeated.

What can you think of?



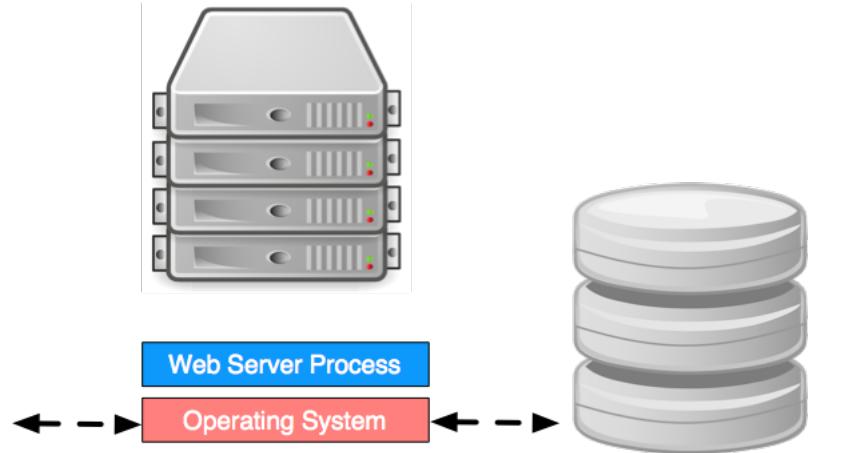
Server-side Caching

There are many parts of a response that are similar.

There are many steps to creating a response that are repeated.

What can you think of?

- View Fragments
- Rarely modified ORM objects
- Any summarized data that is expensive to compute



Server-side Caching

View Fragments: Similarity between pages

```
<html><head><meta name="referrer" content="origin"><link rel="stylesheet" type="text/css" href="news.css?Dc4WHhDIHIntiL20h45C">
<link rel="shortcut icon" href="favicon.ico">
<link rel="alternate" type="application/rss+xml" title="RSS" href="rss">
<script type="text/javascript">
    function hide(id) { var el = document.getElementById(id); if (el) { el.style.visibility = 'hidden'; } }
    function vote(node) { var v = node.id.split('_'); var item = v[1]; hide('up_' + item); hide('down_' + item); var ping = new Image(); ping.src = node.href; return false; }

</script><title>Hacker News</title></head><body><center><table id="hnmain" op="news" border="0" cellpadding="0" cellspacing="0" width="85%" bgcolor="#f6f6ef">
<tr><td bgcolor="#ff6600"><table border="0" cellpadding="0" cellspacing="0" width="100%" style="padding:2px"><tr><td style="width:18px;padding-right:4px;"><a href="http://www.ycombinator.com"></a></td>
<td style="line-height:12pt; height:10px;"><span class="pagetop">
    <b><a href="news">Hacker News</a></b><a href="newest">new</a> | <a href="newcomments">comments</a> | <a href="show">show</a> | <a href="ask">ask</a> | <a href="jobs">jobs</a> | <a href="submit">submit</a></span></td><td style="text-align:right;padding-right:4px;"><span class="pagetop">
        <a href="login?goto=news">login</a></span></td></tr></table></td></tr><tr style="height:10px"></tr><tr><td>
        <table border="0" cellpadding="0" cellspacing="0" style="height:10px"></td><td><table border="0" cellpadding="0" cellspacing="0" style="height:10px"></td>
        <tr><td align="right" valign="top" class="title"><span class="rank">1.</span></td>
        <td><center><a id="up_9403571" href="vote?for=9403571&dir=up&goto=news"><div class="votearrow" title="upvote"></div></a></center></td><td class="title">
            <span class="deadmark"></span><a href="http://fossdroid.com/">Fossdroid.com: Free and open source Android applications</a><span class="sitebit comhead"> (fossdroid.com)</span>
        </td></tr><tr><td colspan="2" style="text-align:center; vertical-align:middle; padding-top:10px;">
            <span class="score" id="score_9403571">164 points</span> by <a href="user?id=SnaKeZ">SnaKeZ</a> <a href="item?id=9403571">4 hours ago</a>
            | <a href="item?id=9403571">38 comments</a></td></tr>
<tr class="spacer" style="height:5px"></tr><tr>.....
```



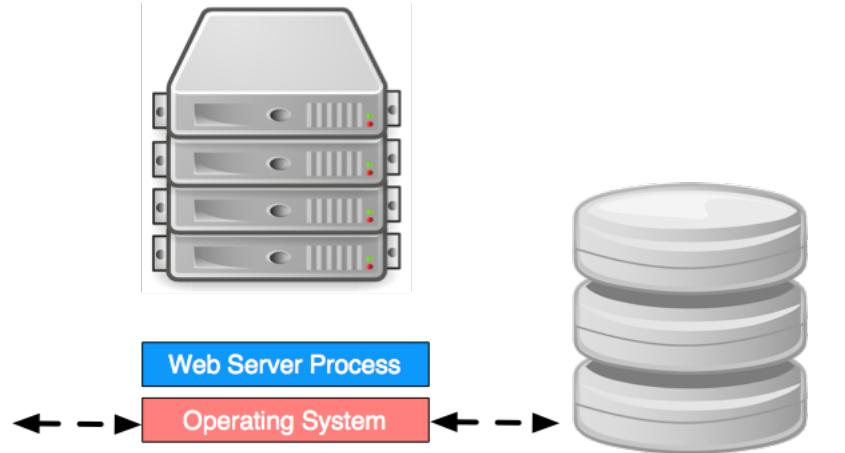
Server-side Caching

Rarely modified ORM objects?

- User permissions
- Configuration options
- Any database-backed data that changes rarely

Summarized data that is difficult to compute

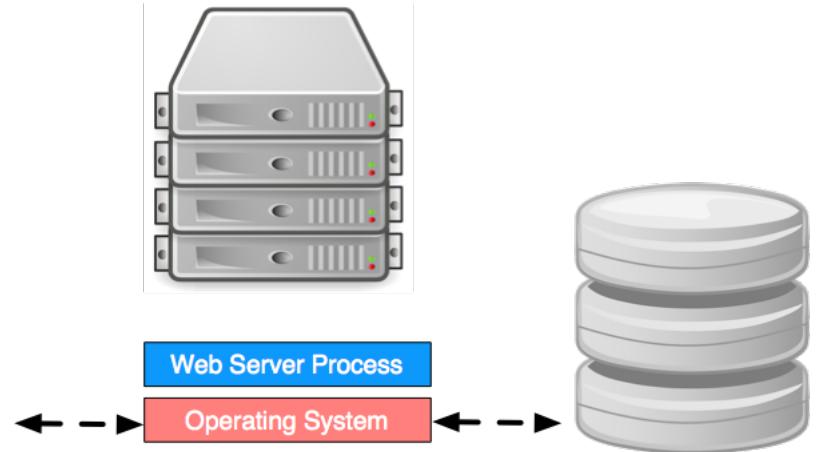
- Any particularly heavyweight SQL query
- Example: Total account balance on Mint.com



Server-side Caching

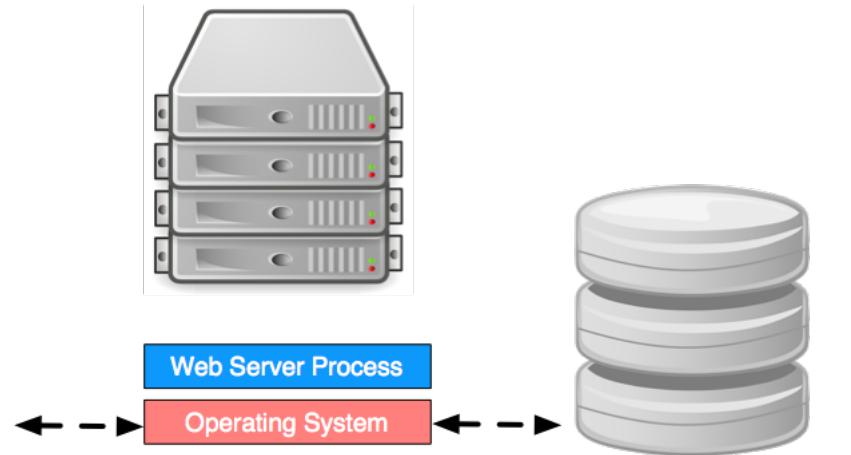
Some of these things are expensive to materialize

- View fragments are produced by extensive string manipulation.
 - Ruby optimizes humans over CPU
- The database can be a bottleneck in our current architecture
- Some SQL queries are necessarily heavyweight



Server-side Caching

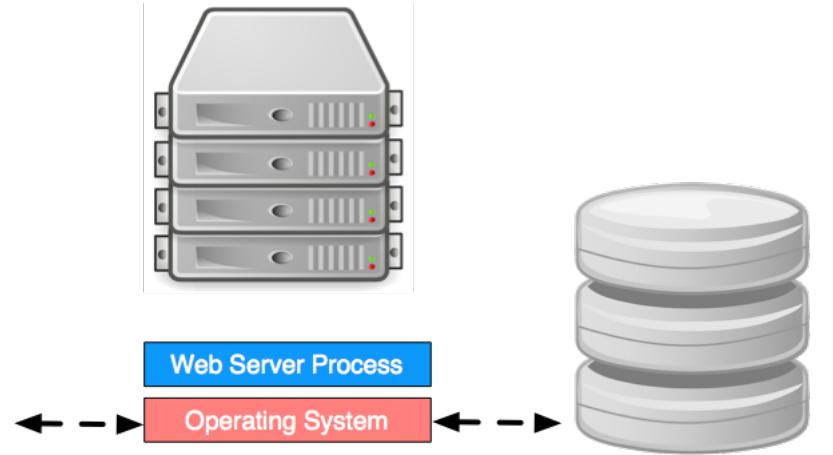
So if we want to keep previously computed results around between requests, how should we do it? Where should we put it?



Server-side Caching

So if we want to keep previously computed results around between requests, how should we do it? Where should we put it?

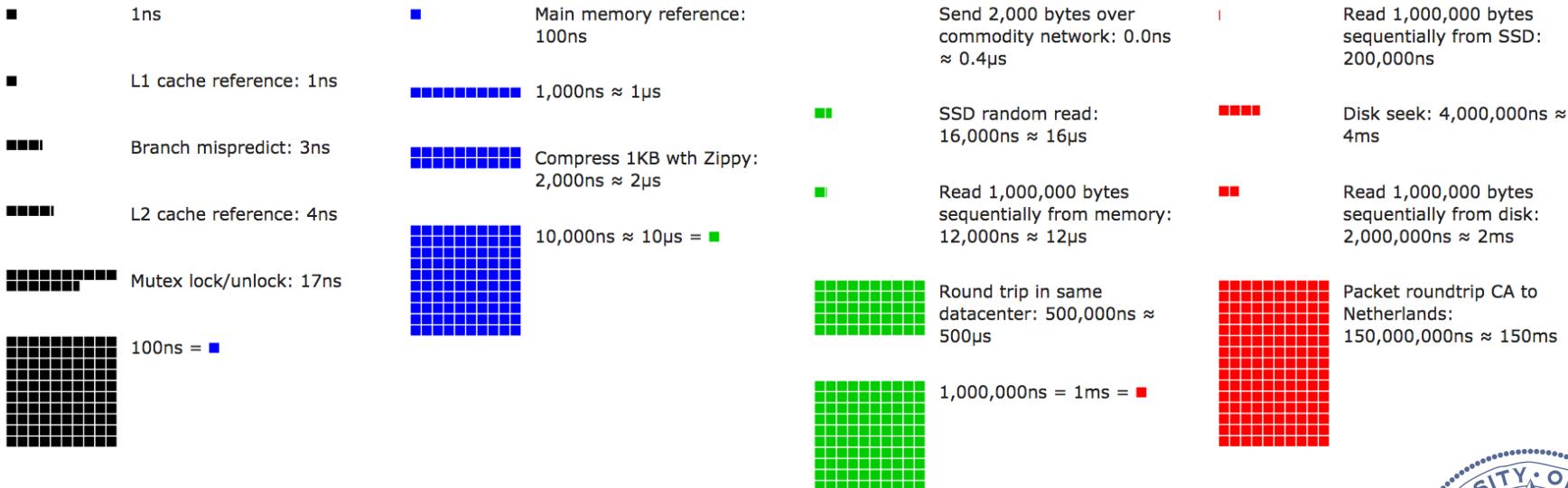
- Just keep it in memory between requests?
- Store it on the filesystem?
- Store it in memory on another machine?



All of these are reasonable options. Lets look into each in more depth.



Server-side Caching



Server-side Caching

What can we conclude from these numbers?

- Storing in memory and reading later is fast:
 - Random reads from memory will be $0.1\mu\text{s}$, reading 1MB will be $12\mu\text{s}$
- Storing on disk is slow *without SSD*:
 - Disk seek is $4000\mu\text{s}$, subsequent sequential read of 1MB is $2000\mu\text{s}$
- Storing on disk *with SSD* is much more reasonable:
 - Random read is $16\mu\text{s}$, sequential read of 1MB will be $200\mu\text{s}$
- Storing on another machine is reasonable:
 - Round trip within datacenter is $500\mu\text{s}$.



Server-side Caching

Summary

- In memory: tens of μs
- On SSD: hundreds of μs
- On Disk: thousands of μs
- And if its on a remote machine, add hundreds of μs .

Conclusion

- Always use SSD
- Memory > SSD > Remote?



Server-side Caching

It's not that simple. Why?



Server-side Caching

It's not that simple.

What effect on the cache hit rate does each of these designs have?

- In memory: Cache per process
- On SSD: Cache per machine
- On (single) remote machine: Cache per cluster



Server-side Caching

Conclusion:

- **In memory:** highest performance, lowest hit rate
- **On SSD:** lower performance, higher hit rate
- **On remote cache server:** lowest performance, highest hit rate

There is no silver bullet. How will each of these affect system performance:

- Number of processes per machine?
- Concurrency model of Application Server: threads vs. processes?
- Number of machines per cluster?



Memcached

Memcached is a commonly used implementation of a remote cache server

- Keeps a cache in memory
- Accepts TCP connections and returns lookup requests
- Distributed key-value store
 - Keys can be up to 250 bytes, values can be up to 1MB
 - Can scale horizontally
- When it runs out of space, it uses a simple LRU mechanism to make more space
- Lightweight features, everything is constant time.
- Originally developed at LiveJournal

Another commonly used tool is Redis



Rails Caching

The good news for your projects is that Rails has great support for server-side caching.

Rails emphasizes three types of caching:

- HTTP caching
- Fragment caching
- Low level caching

We covered HTTP caching in the last lecture, so today we will talk about fragment caching and low-level caching



Rails Caching

By default, caching is disabled in development and test, and enabled in production

- If you want to use it in development mode, add this to your environment:

```
config.action_controller.perform_caching = true
```

Rails can be configured to store cached data in a few different places:

- In memory
- Local file system
- Remote in-memory store



Rails Caching

ActiveSupport::Cache::MemoryStore

- Cached data is stored in memory, in the same address space as the ruby process and is retained between requests.
- Defaults to 32 megs, but is configurable.

ActiveSupport::Cache::FileStore

- Cached data is stored on the local file system.
- Can configure the location of the storage in Rails environment:
 - `config.cache_store = :file_store, "/path/to/cache/"`



Rails Caching

ActiveSupport::Cache::MemcacheStore

- Cached data is stored in memory on another machine.
- Can configure the location of the server in Rails environment:
 - config.cache_store = :mem_cache_store, "cache-1.example.com"



Rails Caching - Fragment Caching

Fragment caching caches a portion of a rendered view for reuse on future requests.

Let's take a look at the demo app...



Rails Caching - Fragment Caching

Demo App			
Submissions			
Title	Url	Community	
A id ea officia.	http://williamson.net/rebekah	Est distinctio qui aut quis doloribus dignissimos sit sed. 20 comments	
Earum sequi veniam libero quibusdam est corporis omnis voluptatem.	http://dare.biz/adolph_pouros	Est distinctio qui aut quis doloribus dignissimos sit sed. 20 comments	
Deleniti vel expedita sint voluptate repellat.	http://schamberger.org/raphaelle	Est distinctio qui aut quis doloribus dignissimos sit sed. 20 comments	
Neque eum sint cum magni.	http://vonruedenborer.name/edythe	Est distinctio qui aut quis doloribus dignissimos sit sed. 20 comments	

We can cache each line of this markup.

Regardless of anything else that changes on the page, we can rerender this if it stays fresh



Rails Caching - Fragment Caching

```
<tbody>
  <% @submissions.each do |submission| %>
    <tr>
      <td><%= link_to(submission.title, submission.url) %></td>
      <td><%= submission.url %></td>
      <td><%= submission.community.name %></td>
      <td><%= link_to "#{submission.comments.size} comments", submission, class: 'btn btn-primary btn-xs' %></td>
    </tr>
  <% end %>
</tbody>
</table>
```



Rails Caching - Fragment Caching

```
<tbody>
  <% @submissions.each do |submission| %>
    <% cache(cache_key_for_submission_row(submission)) do %>
      <tr>
        <td><%= link_to(submission.title, submission.url) %></td>
        <td><%= submission.url %></td>
        <td><%= submission.community.name %></td>
        <td><%= link_to "#{submission.comments.size} comments", submission, class: 'btn btn-primary btn-xs' %></td>
      </tr>
    <% end %>
  <% end %>
</tbody>
</table>
```



Rails Caching - Fragment Caching

How should we choose a cache key?

```
module SubmissionsHelper
  def cache_key_for_submission_row(submission)
    "submission-#{submission.id}"
  end
end
```

What are the weaknesses with the above approach?



Rails Caching - Fragment Caching

How should we choose a cache key?

```
module SubmissionsHelper
  def cache_key_for_submission_row(submission)
    "submission-#{submission.id}"
  end
end
```

What are the weaknesses with the above approach?

- Invalidation will be annoying: clear out the cache on possible action causing staleness?



Rails Caching - Fragment Caching

Instead, let's make the key change whenever the data gets stale.

```
module SubmissionsHelper
  def cache_key_for_submission_row(submission)
    "submission-#{submission.id}-#{submission.updated_at}-#{submission.comments.count}"
  end
end
```

There is no action needed to invalidate the cache: the cache key changes.



Rails Caching - Fragment Caching

Submissions			
Title	Url	Community	
A id ea officia.	http://williamson.net/rebekah	Est distinctio qui aut quis doloribus dignissimos sit sed.	20 comments
Earum sequi veniam libero quibusdam est corporis omnis voluptatem.	http://dare.biz/adolph_pouros	Est distinctio qui aut quis doloribus dignissimos sit sed.	20 comments
Deleniti vel expedita sint voluptate repellat.	http://schamberger.org/raphaelle	Est distinctio qui aut quis doloribus dignissimos sit sed.	20 comments
Neque eum sint cum magni.	http://vonruedenborer.name/edythe	Est distinctio qui aut quis doloribus dignissimos sit sed.	20 comments

If we step back and look at this page, we can observe that the whole table is expensive to compute and stays fresh for awhile.



Rails Caching - Fragment Caching

```
<h3>Submissions</h3>
<table class="table">
  <thead>
    <tr>
      <th>Title</th>
      <th>Url</th>
      <th>Community</th>
      <th colspan="3"></th>
    </tr>
  </thead>

  <tbody>
    <% @submissions.each do |submission| %>
      <% cache(cache_key_for_submission_row(submission)) do %>
        <tr>
          <td><%= link_to(submission.title, submission.url) %></td>
          <td><%= submission.url %></td>
          <td><%= submission.community.name %></td>
          <td><%= link_to "#{submission.comments.size} comments", submission, class: 'btn btn-primary btn-xs' %></td>
        </tr>
      <% end %>
    <% end %>
  </tbody>
</table>

<br>
<%= link_to 'New Submission', new_submission_path, class: 'btn btn-primary' %>
<%= link_to 'New Community', new_community_path, class: 'btn btn-primary' %>
```



Rails Caching - Fragment Caching

```
<% cache(cache_key_for_submission_table) do %>
<h3>Submissions</h3>
<table class="table">
<thead>
<tr>
<th>Title</th>
<th>Url</th>
<th>Community</th>
<th colspan="3"></th>
</tr>
</thead>

<tbody>
<% @submissions.each do |submission| %>
<% cache(cache_key_for_submission_row(submission)) do %>
<tr>
<td><%= link_to(submission.title, submission.url) %></td>
<td><%= submission.url %></td>
<td><%= submission.community.name %></td>
<td><%= link_to "#{submission.comments.size} comments", submission, class: 'btn btn-primary btn-xs' %></td>
</tr>
<% end %>
<% end %>
</tbody>
</table>

<br>
<%= link_to 'New Submission', new_submission_path, class: 'btn btn-primary' %>
<%= link_to 'New Community', new_community_path, class: 'btn btn-primary' %>
<% end %>
```



Rails Caching - Fragment Caching

```
module SubmissionsHelper
  def cache_key_for_submission_row(submission)
    "submission-#{submission.id}-#{submission.updated_at}-#{submission.comments.count}"
  end
  def cache_key_for_submission_table
    "submission-table-#{Submission.maximum(:updated_at)}-#{Comment.maximum(:updated_at)}"
  end
end
```

This technique of nesting cache fragments is known as “Russian Doll” caching.



Rails Caching - Low-level Caching

You can use the same mechanisms to cache anything:

```
class Product < ActiveRecord::Base
  def competing_price
    Rails.cache.fetch("#{cache_key}/competing_price", expires_in: 12.hours) do
      Competitor::API.find_price(id)
    end
  end
end
```



Rails Caching - Low-level Caching

Let's compare the demo app's performance!

For these tests I will compare the performance of the branch master, with the branch "server_side_caching", which implements the caching shown in the previous slides.

Master intentionally includes no optimizations to the way we interact with the database.

We will use the default (memory) caching.



Rails Caching

We'll use an M3-medium instance with the usual workload.

When we test we will have 12 phases, of 60 seconds each:

Phase	1	2	3	4	5	6	7	8	9	10	11	12
Users/sec	1	1.5	2	4	6	10	16	20	25	35	45	55

Deployed using nginx & passenger.



Rails Caching

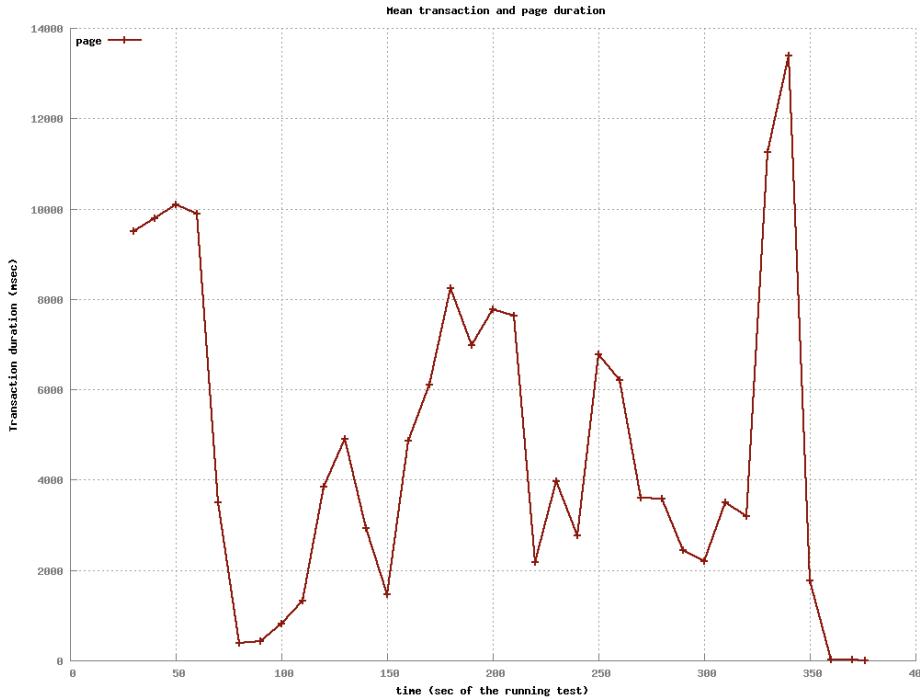
We will use the same testing script from before...

1. Going to the homepage
2. Waiting for up to 2 seconds
3. Requesting a form to create a new community
4. Waiting for up to 2 seconds
5. Submitting the new community
6. Requesting a form to create a new link submission
7. Waiting for up to 2 seconds
8. Submitting the new link
9. Waiting for up to 2 seconds
10. Delete the link
11. Waiting for up to 2 seconds
12. Delete the community



Rails Caching

Performance on master:

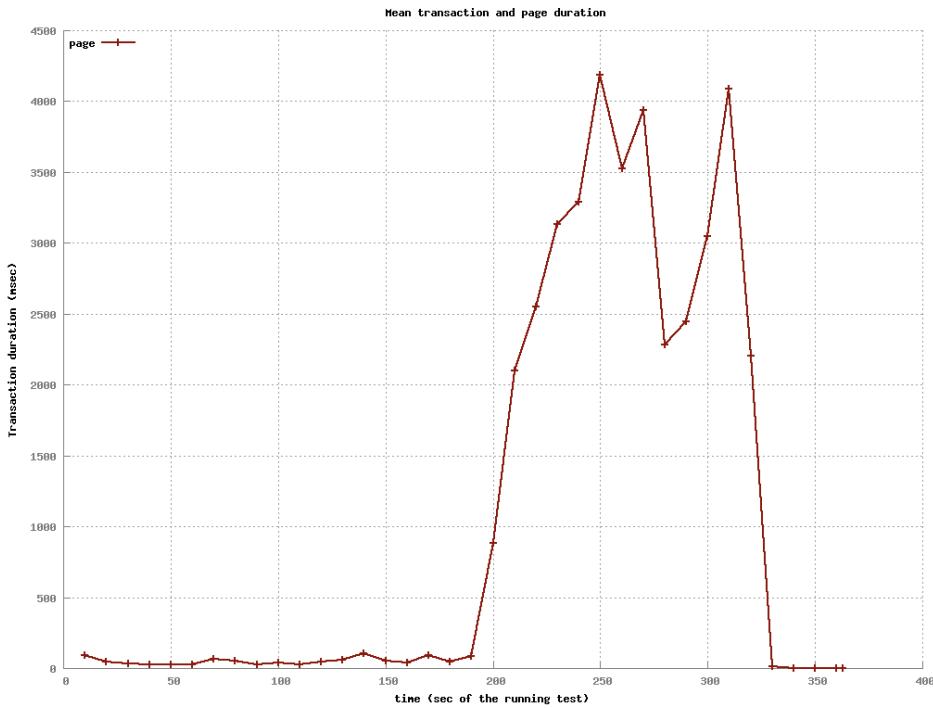


From the start, the application can't handle a single user arriving each second



Rails Caching

Performance on server_side_caching:



With the server-side caching implemented, the server can handle up to two new users a second easily.



AWS Instructions

Now that you've all got a blank Rails app (or more) pushed to Github, it's time to learn how to deploy to AWS.

We will be using CloudFormation, and here are step-by-step instructions for deploying.

But first, some warnings and rules.



AWS Instructions

These are scarce resources

- Amazon has been very generous to give you all free time on their infrastructure.
- Unless you have a specific reason to do otherwise, always use micro instances.
 - Example of a good reason: testing vertical scaling.



AWS Instructions

These are scarce resources

- Our AWS budget has a fixed limit.
- Whenever you are done with an instance, shut it down.
- Never keep important data on the instance, because it can go down at any time.
 - SCP important data back to your laptop.
- I will periodically run a script that terminates all instances that have been on longer than X hours.



AWS Instructions

Go to <https://scalable-internet-services.signin.aws.amazon.com/console>



Account:

User Name:

Password:

I have an MFA Token (more info)

Sign In

[Sign-in using root account credentials](#)

English 

[Terms of Use](#) [Privacy Policy](#) © 1996-2015, Amazon Web Services, Inc. or its affiliates.

Login with Username and Password provided in email.



AWS Instructions

AWS Services Edit Andrew Mutz Oregon Support

Amazon Web Services

Compute

- EC2 Virtual Servers in the Cloud
- Lambda Run Code in Response to Events
- EC2 Container Service Run and Manage Docker Containers

Storage & Content Delivery

- S3 Scalable Storage in the Cloud
- Storage Gateway Integrates On-Premises IT Environments with Cloud Storage
- Glacier Archive Storage in the Cloud
- CloudFront Global Content Delivery Network

Database

- RDS MySQL, Postgres, Oracle, SQL Server, and Amazon Aurora
- DynamoDB Predictable and Scalable NoSQL Data Store
- ElastiCache In-Memory Cache
- Redshift Managed Petabyte-Scale Data Warehouse Service

Networking

- VPC Isolated Cloud Resources
- Direct Connect Dedicated Network Connection to AWS
- Route 53 Scalable DNS and Domain Name Registration

Administration & Security

- Directory Service Managed Directories in the Cloud
- Identity & Access Management Access Control and Key Management
- Trusted Advisor AWS Cloud Optimization Expert
- CloudTrail User Activity and Change Tracking
- Config Resource Configurations and Inventory
- CloudWatch Resource and Application Monitoring

Deployment & Management

- Elastic Beanstalk AWS Application Container
- OpsWorks DevOps Application Management Service
- CloudFormation Templated AWS Resource Creation
- CodeDeploy Automated Deployments

Analytics

- EMR Managed Hadoop Framework
- Kinesis Real-time Processing of Streaming Big Data
- Data Pipeline Orchestration for Data-Driven Workflows
- Machine Learning Build Smart Applications Quickly and Easily

Application Services

- SQS Message Queue Service
- SWF Workflow Service for Coordinating Application Components
- AppStream Low Latency Application Streaming
- Elastic Transcoder Easy-to-use Scalable Media Transcoding
- SES Email Sending Service
- CloudSearch Managed Search Service

Mobile Services

- Cognito User Identity and App Data Synchronization
- Mobile Analytics Underly Analytics App Usage Data at Scale
- SNS Push Notification Service

Enterprise Applications

- WorkSpaces Desktops in the Cloud
- WorkDocs Secure Enterprise Storage and Sharing Service
- WorkMail PREVIEW Secure Email and Calendaring Service

Service Health

- All services operating normally.

Updated: Apr 19 2015 20:56:00 GMT-0700

[Service Health Dashboard](#)

Make sure your region is set to
“US-West Oregon”



AWS Instructions

AWS Services

Andrew Mutz | Oregon | Support

Amazon Web Services

Compute

- EC2 Virtual Servers in the Cloud
- Lambda Run Code in Response to Events
- EC2 Container Service Run and Manage Docker Containers

Storage & Content Delivery

- S3 Scalable Storage in the Cloud
- Storage Gateway Integrates On-Premises IT Environments with Cloud Storage
- Glacier Archive Storage in the Cloud
- CloudFront Global Content Delivery Network

Database

- RDS MySQL, Postgres, Oracle, SQL Server, and Amazon Aurora
- DynamoDB Predictable and Scalable NoSQL Data Store
- ElastiCache In-Memory Cache
- Redshift Managed Petabyte-Scale Data Warehouse Service

Networking

- VPC Isolated Cloud Resources
- Direct Connect Dedicated Network Connection to AWS
- Route 53 Scalable DNS and Domain Name Registration

Administration & Security

- Directory Service Managed Directories in the Cloud
- Identity & Access Management Access Control and Key Management

Trusted Advisor

- AWS Cloud Optimization Expert
- CloudTrail User Activity and Change Tracking
- Config Resource Configurations and Inventory
- CloudWatch Resource and Application Monitoring

Deployment & Management

- Elastic Beanstalk AWS Application Container
- OpsWorks Dev/Ops Application Management Service
- CloudFormation Templated AWS Resource Creation
- CodeDeploy Automated Deployments

Analytics

- EMR Managed Hadoop Framework
- Kinesis Real-time Processing of Streaming Big Data
- Data Pipeline Orchestration for Data-Driven Workflows

Machine Learning

- Build Smart Applications Quickly and Easily

Application Services

- SQS Message Queue Service
- SWF Workflow Service for Coordinating Application Components

AppStream

- Low Latency Application Streaming

Elastic Transcoder

- Easy-to-use Scalable Media Transcoding

SES

- Email Sending Service

CloudSearch

- Managed Search Service

Resource Groups

A resource group is a collection of resources that share one or more tags. Create a group for each project, application, or environment in your account.

Create a Group Tag Editor

Additional Resources

Getting Started

See our documentation to get started and learn more about how to use our services.

Mobile Services

AWS Console Mobile App

View your resources on the go with our AWS Console mobile app, available from Amazon Appstore, Google Play, or iTunes.

AWS Marketplace

Find and buy software, launch with 1-Click and pay by the hour.

AWS Summit - San Francisco

Learn about the exciting new services and features announced at the AWS Summit in San Francisco

Service Health

All services operating normally.

Updated: Apr 19 2015 20:56:00 GMT-0700

Service Health Dashboard

We will be deploying with clouformation, so click on that



AWS Instructions

AWS Services Edit Andrew Mutz Oregon Support

Create Stack Update Stack Delete Stack C G

Filter: Active By Name: Showing 3 stacks

Stack Name	Created Time	Status	Description
<input type="checkbox"/> demo2	2015-04-19 19:24:50 UTC-0700	DELETE_IN_PROGRESS	
<input checked="" type="checkbox"/> admin	2015-03-23 17:59:22 UTC-0700	CREATE_COMPLETE	
<input type="checkbox"/> tsung	2015-03-16 13:25:07 UTC-0700	CREATE_COMPLETE	

Overview Outputs Resources Events Template Parameters Tags Stack Policy

Stack name: admin

Stack ID: arn:aws:cloudformation:us-west-2:122276281439:stack/admin/026d5350-d1c1-11e4-914c-5088487b0895

Status: CREATE_COMPLETE

Status reason:

Description:

Click on “Create Stack”



AWS Instructions

Screenshot of the AWS CloudFormation "Select Template" step.

The page shows the navigation bar: AWS Services Edit Andrew Mutz Oregon Support.

Select Template

Specify a stack name and then select the template that describes the stack that you want to create.

Stack

An AWS CloudFormation stack is a collection of related resources that you provision and update as a single unit.

Name

Template

A template is a JSON-formatted text file that describes your stack's resources and their properties. AWS CloudFormation stores the stack's template in an Amazon S3 bucket. [Learn more](#).

Source

Select a sample template

Upload a template to Amazon S3 No file chosen

Specify an Amazon S3 template URL

[Cancel](#) [Next](#)

The stack name must begin with your team name.

For example: “demo-test”



AWS Instructions

The screenshot shows a GitHub repository page for 'scalableinternetservices / utils'. The repository has 120 commits, 2 branches, 0 releases, and 2 contributors. The master branch is selected. The repository contains scripts for managing GitHub and AWS accounts. A section titled 'Scalable Internet Services Templates' lists various single instance templates, including NGINX + Passenger, NGINX + Passenger + memcached, Puma, and WEBrick. The WEBrick template is highlighted with a red border.

This repository Search Explore Gist Blog Help amutz Unwatch Star Fork

scalableinternetservices / utils

Scripts to help manage github and AWS accounts for the various teams. — Edit

120 commits 2 branches 0 releases 2 contributors

branch: master / +

Merge branch 'master' of https://github.com/scalableinternetservices/... amutz authored 7 days ago latest commit 5b15ae099b

.gitignore Update .gitignore. 6 months ago

Bootstrap.json auto check out the utils in the bootstrap template 27 days ago

README.md Update README.md 16 days ago

admin.py Merge branch 'master' of https://github.com/scalableinternetservices/... 7 days ago

README.md

Scalable Internet Services Templates

Single Instance Templates

Both the app server, and database are located on a single EC2 instance.

- **NGINX + Passenger** (Recommended for regular testing):
NGINX handles requests to port 80 and passes connections to instances of the app through Passenger. Multiple concurrent connections are supported.
<https://scalableinternetservices.s3.amazonaws.com/SinglePassenger.json>
- **NGINX + Passenger + memcached**:
Same as above, with the addition of using memcached through the dalli gem.
<https://scalableinternetservices.s3.amazonaws.com/SinglePassengerMemcached.json>
- **Puma**:
Puma allows both thread-based and process-based concurrency.
<https://scalableinternetservices.s3.amazonaws.com/SinglePuma.json>
- **WEBrick** (Use only for slow-performance testing):
WEBrick handles requests to port 80 directly, permitting only a single connection at a time.
<https://scalableinternetservices.s3.amazonaws.com/SingleWEBrick.json>

Go to

<https://github.com/scalableinternetservices/utils>
and choose your template.

Today just choose the Webrick template.

Copy the link.



AWS Instructions

Screenshot of the AWS CloudFormation "Select Template" step.

The page shows the AWS navigation bar with "Services" selected. The user is at the "Select Template" step, which is the first in a multi-step process.

Select Template

Specify a stack name and then select the template that describes the stack that you want to create.

Stack

An AWS CloudFormation stack is a collection of related resources that you provision and update as a single unit.

Name

Template

A template is a JSON-formatted text file that describes your stack's resources and their properties. AWS CloudFormation stores the stack's template in an Amazon S3 bucket. [Learn more.](#)

Source

Select a sample template

Upload a template to Amazon S3 No file chosen

Specify an Amazon S3 template URL

Cancel

Paste that link into the field that says “Specify an Amazon S3 template URL”

Click Next.



AWS Instructions

Screenshot of the AWS CloudFormation 'Specify Parameters' step.

The screenshot shows the 'Specify Parameters' step of a CloudFormation template. On the left, a sidebar menu includes 'Select Template', 'Specify Parameters' (which is highlighted in orange), 'Options', and 'Review'. The main area is titled 'Specify Parameters' and contains instructions: 'Specify values or use the default values for the parameters that are associated with your AWS CloudFormation template.' Below this is a 'Parameters' section with three fields:

- AppInstanceType**: A dropdown menu set to 't1.micro'. A tooltip says: 'The AppServer instance type.'
- Branch**: An input field containing 'master'. A tooltip says: 'The git branch to deploy.'
- TeamName**: A dropdown menu set to 'demo'. A tooltip says: 'Your team name.'

At the bottom of the screen are 'Cancel', 'Previous', and 'Next' buttons.

Choose your fields.

Select your TeamName from the dropdown.

The teamname is how the template knows where to get your code.



AWS Instructions

Options

Tags

You can specify tags (key-value pairs) for resources in your stack. You can add up to 10 unique key-value pairs for each stack. [Learn more.](#)

Key (127 characters maximum)	Value (255 characters maximum)
1	

Advanced

You can set additional options for your stack, like notification options and a stack policy. [Learn more.](#)

Notification options

No notification

New Amazon SNS topic

Topic

Email

Existing Amazon SNS topic

Timeout Minutes

Rollback on failure Yes

No

Stack policy Enter policy

Upload policy file

Choose File No file chosen

[Learn more](#)

On the next screen just choose “Next”

If you are having problems deploying, you can disable rollback.



AWS Instructions

Screenshot of the AWS CloudFormation console showing the list of stacks and the outputs tab.

The top navigation bar includes: AWS, Services, Edit, Andrew Mutz, Oregon, Support.

Stacks List:

Stack Name	Created Time	Status	Description
demo-test	2015-04-19 21:09:13 UTC-0700	CREATE_COMPLETE	
admin	2015-03-23 17:59:22 UTC-0700	CREATE_COMPLETE	
tsung	2015-03-16 13:25:07 UTC-0700	CREATE_COMPLETE	

Outputs Tab:

Key	Value	Description
SSH	ssh -i demo.pem ec2-user@ec2-52-11-199-3.us-west-2.compute.amazonaws.com	AppServer SSH connect string
URL	http://ec2-52-11-199-3.us-west-2.compute.amazonaws.com	The URL to the rails application.

After creation, the outputs tab in the bottom pane will tell you how to reach your server via HTTP and SSH.

The PEM file mentioned in the SSH command was emailed to you.

SCP accepts the same
-i FILE.pem argument



AWS Instructions

Screenshot of the AWS CloudFormation console showing the list of stacks and a detailed view of the "demo-test" stack.

The top navigation bar includes: AWS, Services, Edit, Andrew Mutz, Oregon, Support.

Buttons: Create Stack, Update Stack, Delete Stack (highlighted with a red box).

Filter: Active, By Name: (input field).

Table header: Stack Name, Created Time, Status, Description.

Table rows:

- demo-test (selected, checked), 2015-04-19 21:09:13 UTC-0700, CREATE_COMPLETE
- admin, 2015-03-23 17:59:22 UTC-0700, CREATE_COMPLETE
- tsung, 2015-03-16 13:25:07 UTC-0700, CREATE_COMPLETE

Showing 3 stacks.

Below the table, tabs: Overview, Outputs, Resources, Events, Template, Parameters, Tags, Stack Policy.

Key Value Description

Key	Value	Description
SSH	ssh -i demo.pem ec2-user@ec2-52-11-199-3.us-west-2.compute.amazonaws.com	AppServer SSH connect string
URL	http://ec2-52-11-199-3.us-west-2.compute.amazonaws.com	The URL to the rails application.

When you are done with your stack, remember to delete it!

If you have any questions, please post to Piazza!



Motivation

After today you should understand

- Why server side caching exists
- What options you have when using server side caching
- How to use this in your projects
- How to deploy on AWS using CloudFormation



For Next Time...

Try and get your app deployed on AWS. If you run into problems, post on Piazza.

Guest speaker, please be on time:

- Darren Mutz, Founder, Red Aspen Software
- “Technical Challenges in Content Delivery Networks”

