Plagia: Cloud Implementation of a Plagiarism Detection System

Omair Jaswal
10 June 2015

Agenda

- Introduction
- Service Oriented Architecture
- Deployment Model
- Technologies used
- Scalability & Cost
- The Application
- The System & Data Flow
- Improvements
- Q&A

Introduction

Plagia is a cloud based web application for plagiarism detection system using contextual n-grams with simplified implementation of CoReMo's Plagiarism Detector.

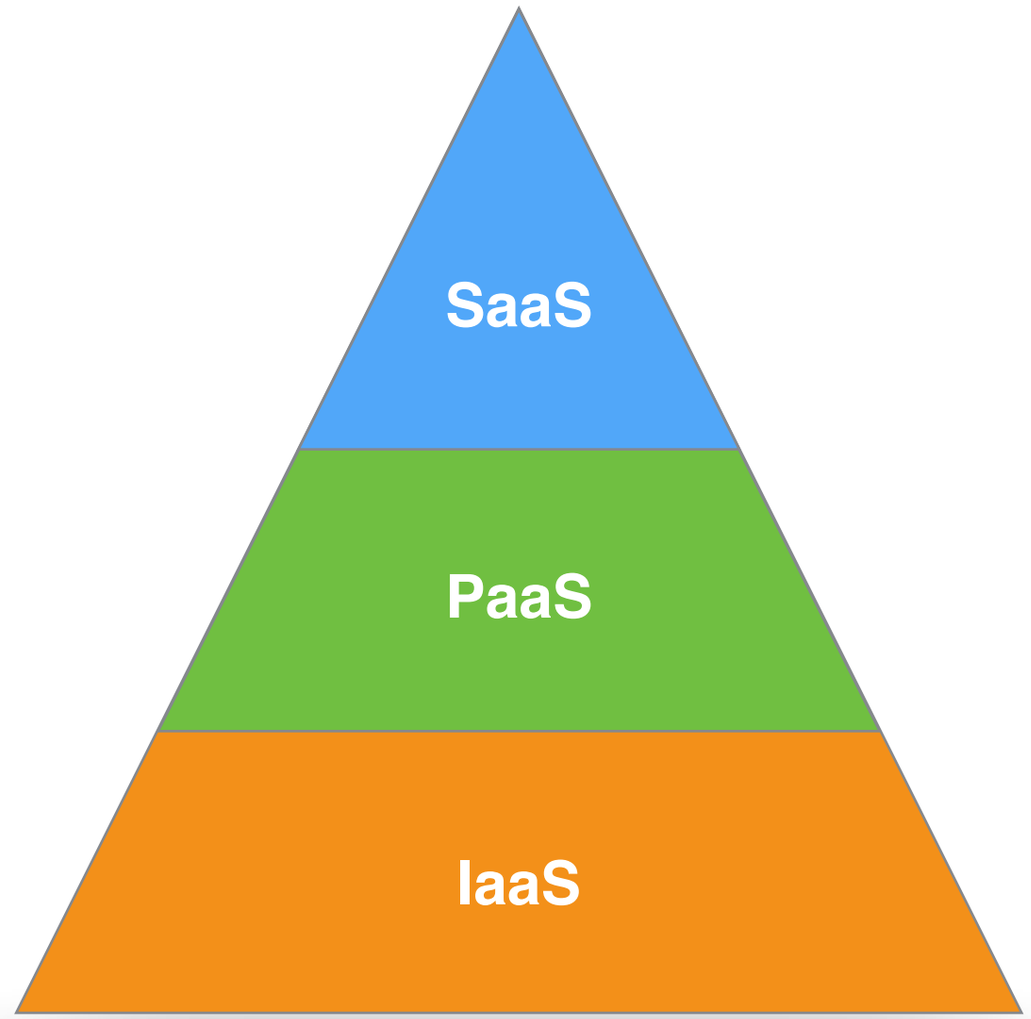
The system processes plagiarism in *"PAN External Plagiarism Corpus 2011"* via a cost effective and high performing web application utilising only the cloud capabilities.

Service Oriented Architecture

SaaS
Plagia *WebApp (possible)*

PaaS
Google *AppEngine*

IaaS
Google *Storage, Cloud SQL*
Amazon *EC2, EMR, S3*
OpenStack *Compute, Sahara, Swift*



* Red is either an improvement or a feature not implemented.

Technologies Used

- Google AppEngine, Cloud SQL, Cloud Storage
- Amazon EC2, EMR, S3,
- OpenStack Compute, Sahara, Swift

- PHP, AJAX, jQuery, Bootstrap framework
- Python, Boto, MySQLdb
- MySQL
- Hadoop MapReduce
- Shell Scripting

* Red is either an improvement or a feature not implemented.

The WebApp

Step 1 Go to: *http://plagia-2015.appspot.com/*

Step 2 Find the blue *"Start a job"* panel.

Step 3 Choose any *"Suspicious Document"* from the list.

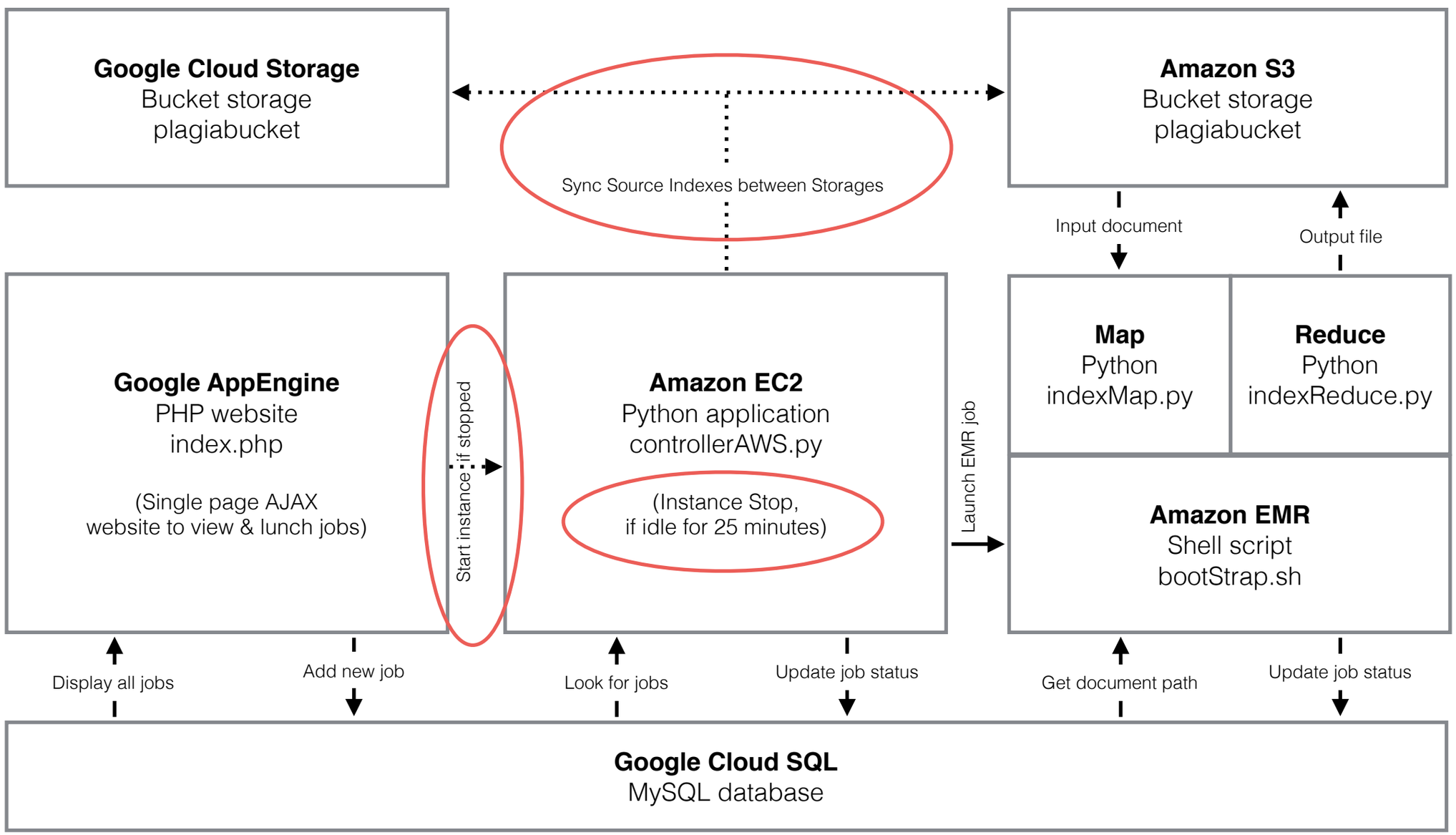
Step 4 Don't worry about the options.
(The *"Source Documents"* will require re-indexing if of Pattern Size option is changed.)

Step 5 Hit the big blue *"Launch!"* button.

Step 6 See live status updates in the grey *"Jobs in progress"* panel.

Step 7 Once job status shows *"Completed"*, click on the document to observe any plagiarism found.

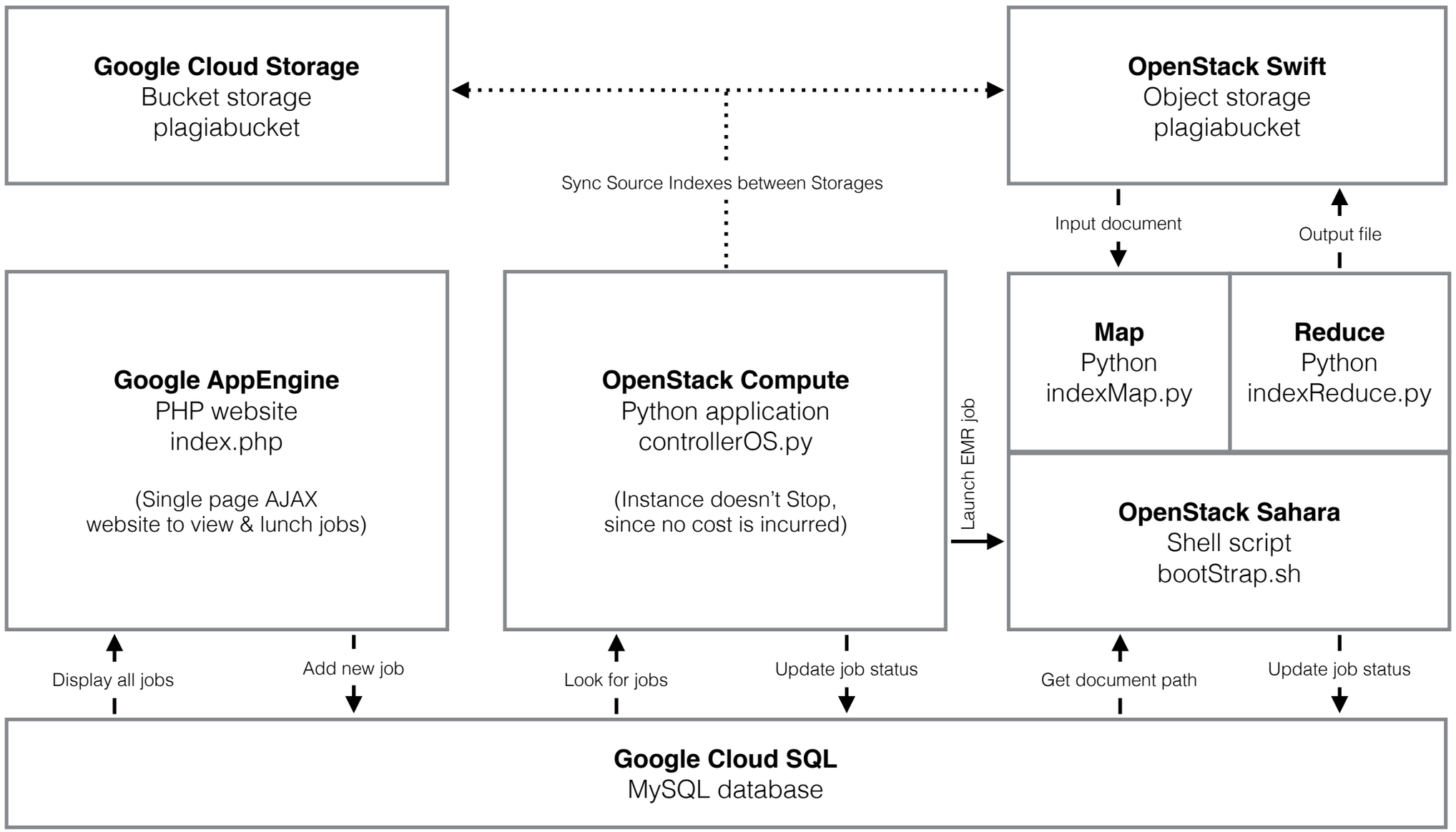
The System & Data Flow



System and Data flow diagram for Amazon Web Services

* Red is either an improvement or a feature not implemented.

The System & Data Flow



System and Data flow diagram for OpenStack

* Red is either an improvement or a feature not implemented.

Conclusion

- Fun and deep learning experience gained about the inner workings of the cloud.
- Could have done a lot more. (ongoing project!)
- Should try implementing OpenStack.
- Try to change SQL querying method to RESTfull API to reduce the overhead on requests made to the server and save money.
- Try to improve speed and performance of the overall system and reduce cost.
- Learn task Queuing, Load Balancing and Auto Scaling.
- Build a more intelligent application.

Q&A