Assignment 7

Due	Points	Questions	Time Limit
Apr 12 at 11:55pm	100	4	None

Instructions

CS-554 Assignment 7

Choosing Technical Stacks

For this lab, you will not be programming, but rather analyzing several scenarios and determining what technologies would work well together to solve these problems. For each answer, back up your reasoning with *why* you chose each technology, strategy, etc.

Attempt History

	Attempt	Time	Score
LATEST	Attempt 1	1,587 minutes	0 out of 100 *

* Some questions not yet graded

Score for this quiz: **0** out of 100 *
Submitted Apr 9 at 8:19pm
This attempt took 1,587 minutes.

Question 1 — Not yet graded / 25 pts

In this scenario, you are tasked with creating a logging server for any number of other arbitrary pieces of technologies.

Your logs should have some common fields, but support any number of customizeable fields for an individual log entry. You should be able to effectively query them based on any of these fields.

How would you store your log entries? How would you allow users to submit log entries? How would you allow them to query log entries? How would you allow them to see their log entries? What would be your web server?

Your Answer:

How would you store your log entries?

I would use Apache Hadoop to store log entries. The most obvious aspect why we don't choose Elasticsearch is that it doesn't support multi-platforms and distributed computer clsuters. However, the scenario requires some common fields from multi-platform users. So, distributed file system is a must. The benefits of using hadoop is that it provides a software framework for distributed storage and processing of big data using the MapReduce programming model, and it is very effecitive in solving problems involving massive amounts of data and computation. In our scenario, using hadoop is the best choice.

How would you allow users to submit log entries?

Using standard input streaming

How would you allow them to query log entries?

Use Map-reduce algorithm. Use Map function takes the data stored in Hadoop and pulls it into a format needed for the operation we need to perform. So in this case, we query the log entries from HDFS distributed file system

How would you allow them to see their log entries?

Use Map-reduce algorithm. Use Reduce accumulates a result based on that data for the operation we need to perform. So, in this case, worker nodes will process each group of output data, per key, in parallel. The final processed data are log entries the users want to see.

What would be your web server?

(1) As hadoop is affiliated to Apache, we had better to use Apache server for the sake of avoiding compatibility conflict by using other server.

(2) Apache server is very easy to configure especially in Linux with command "sudo apt-get install apache2"

Question 2

Not yet graded / 25 pts

In this scenario, you are tasked with making an expense reporting web application.

Users should be able to submit expenses, which are always of the same data structure:

`id`, `user`, `isReimbursed`, `reimbursedBy`, `submittedOn`, `paidOn`, and `amount`.

When an expense is reimbursed you will generate a PDF and email it to the user who submitted the expense.

How would you store your expenses? What web server would you choose, and why? How would you handle the emails? How would you handle the PDF generation? How are you going to handle all the templating for the web application?

Your Answer:

How would you store your expenses?
Using redis. Since redis is an open-source in-memory database project implementing a distributed, in-memory key-value store with optional durability. It supports many abstract data structures, such as strings, lists, maps, sets, sorted sets, hyperloglogs, bitmaps and spatial indexes. So, in this case, we will use lists to save the expenses. Specific method is like lpushAsync() to add expenses to redis

What web server would you choose, and why?
choosing Express server, because the expense data structure is in JSON format while Express simply returns JSON

How would you handle the emails?
Communicating over IPC(Inter-process communication). In this case, the best way is to use messaging queues to handle emails.
From a design perspective a message queue is essentially a FIFO queue, which is a rather fundamental data type: What makes a message queue special is that while the application is responsible for en-queueing, a different process would be responsible for de-queueing. In queueing lingo, the application is the sender of the message(s), and the de-queueing process is the receiver. The obvious advantage is that the whole process is asynchronous, the receiver works independently of the sender, as long as there are messages to process. However, we need an extra component, the sender, for the whole thing to work.

How would you handle the PDF generation?
(1)use LaTeX. LaTeX is a document preparation system for creating printable documents. LaTeX is a content first approach, which can then be combined with a set of styles. LaTeX is effectively the language we can write PDFs in.
(2)use wkhtmltopdf. It's growing to compete with LaTeX for PDF

generation. We would compose an HTML document complete with CSS and images and send it through wkhtmltopdf to be converted into a PDF.

How are you going to handle all the templating for the web application?

Using Streams and Shell Commands.

First, we need to understand what is template. Hopefully most of us are aware of the MVC (Model, View, Controller) design model, where models process data, views show the results and finally, controllers handle user requests. For views, many dynamic languages generate data by writing code in static HTML files. For instance, JSP is implemented by inserting <%=....=%>, PHP by inserting <?php.....?>, etc. So, the template mechanism is like this: (1)Hello, my name is {Name}, (2) Name = "Mary" and we can get (3) Hello, my name is Mary.

Therefore, how these components in HTML files communicate with each other. The best way is to use Streams and Shell Commands. Couple of ways for processes to talk to each other:

* Through messaging queues such as RabbitMQ
* Through pub/subs like Redis
* Through pipes, such as stdin/stdout
* Through memory mapped files
* Through sockets

Question 3

Not yet graded / 25 pts

In this scenario, you are tasked with creating a service for your local Police Department that keeps track of Tweets within your area and scans for keywords to trigger an investigation.

This application comes with several parts:

- An online website to CRUD combinations of keywords to add to your trigger. For example, it would alert when a tweet contains the words (`fight` **or** `drugs`) AND (`SmallTown USA HS` or `SMUHS`).
- An email alerting system to alert different officers depending on the contents of the Tweet, who tweeted it, etc.
- A text alert system to inform officers for critical triggers (triggers that meet a combination that is marked as extremely important to note).
- A historical database to view possible incidents (tweets that triggered an alert) and to mark its investigation status.
- A historical log of *all* tweets to retroactively search through.

- A streaming, online incident report. This would allow you to see tweets as they are parsed and see their threat level. This updates in real time.
- A long term storage of all the media used by any tweets in your area (pictures, snapshots of the URL, etc).

Which Twitter API do you use? How would you build this so its expandable to beyond your local precinct? What would you do to make sure that this system is constantly stable? What would be your web server technology? What databases would you use for triggers? For the historical log of tweets? How would you handle the real time, streaming incident report? How would you handle storing all the media that you have to store as well? What web server technology would you use?

Your Answer:

Which Twitter API do you use?

Use REST API. The Reason is that use REST API in our products is convenient to have different pieces of software communicate, even if they are running on entirely different platforms and if some of your products are internal only. In this case, there are 6 components running on different platforms: An online website, An email alerting system, A historical database, A historical log of all tweets, A streaming, online incident report, A long term storage of all the media. So, these components will work very well by using REST API

How would you build this so its expandable to beyond your local precinct?

Use Apache Hive with HDFS file systems. Apache Hive is a data warehouse software project built on top of Apache Hadoop for providing data summarization, query and analysis. Hive gives an SQL-like interface to query data stored in various databases and file systems that integrate with Hadoop. It Especially supports analysis of large datasets stored in Hadoop's HDFS and compatible file systems such as Amazon S3 filesystem.

In this case, the local precinct databases and other places databases will share their data in Apache Hive.

What would you do to make sure that this system is constantly stable?

(1)Accelerate and Secure Applications with a Reverse Proxy Server

Load balancing (Minimum number of concurrent requests. Maximum time to serve a request. Minimum number of requests per hour.)

Caching static files

Securing the site

(2)Add a Load Balancer

(3)Cache Static and Dynamic Content

Moving content closer to users
Moving content to faster machines
Moving content off of overused machines
(4)Compress Data
(5)Optimize SSL/TLS
Session caching
Session tickets or IDs
OCSP stapling
(6)Implement HTTP/2 or SPDY
(7)Update Software Versions
(8)Tune Linux for Performance
Backlog queue
File descriptors
Ephemeral ports
(9)Tune Your Web Server for Performance
Access logging
Buffering
Client keepalives
Upstream keepalives
Limits
Worker processes
Socket sharding
Thread pools
(10)Monitor Live Activity to Resolve Issues and Bottlenecks

What would be your web server technology?

Exactly Express server is the best choice because REST API will easily interact with Express. An architectural style called REST (Representational State Transfer) advocates that web applications should use HTTP as it was originally envisioned. Lookups should use GET requests. PUT, POST, and DELETE requests should be used for mutation, creation, and deletion respectively.

As Express server has GET, PUT, POST, DELETE requests, it fits well. Like Apache server only has GET and POST, so it won't work.

What databases would you use for triggers? For the historical log of tweets?

Use Oracle PL/SQL to deal with Triggers. The format is like this (CREATE [OR REPLACE] TRIGGER trigger_name). Triggers are stored programs, which are automatically executed or fired when some events occur. In this

case triggers are fight, drugs, SmallTown USA HS, and SMUHS.

Use MongoDB for historical log of tweets. Because mostly the logs are stored in JSON format. MongoDB is document-oriented database which uses JSON-like documents with schemas. So in this case, we will save historical log of tweets in documents and store it in MongoDB.

How would you handle the real time, streaming incident report?

Use PagerDuty API. PagerDuty is a cloud-based platform that provides incident management and response across a huge range of third-party operational services. With PagerDuty, it is very easy to manage incident response and escalation across a collection of services and team members. PagerDuty is the hub that collects and organizes all of the necessary information and functionality for operational response in one place. The PagerDuty API features a number of endpoints for escalation policies, incidents, log entries, maintenance windows, schedules and notifications

Specific way to use PagerDuty API:

* dividing up the code using BLOCK
* dividing up the code using User Interface

How would you handle storing all the media that you have to store as well?

Use ImageMagick Since it is a very powerful image manipulation utility. It is a command line utility that essentially allows you to perform any sort of conversion imaginable on an image. We can run images through ImageMagick to have them compressed, resized, cropped,etc. Also, it's very convenient to upload photos

In this case, we have resources like pictures and snapshots of the URL. So ImageMagick is the most suitable way to handle media data.

What web server technology would you use?

Same question

Question 4

Not yet graded / 25 pts

In this scenario, you are tasked with creating the web server side for a mobile application

where people take pictures of mildly interesting things and upload them. The mobile application allows users to see mildly interesting pictures in their geographical location.

Users must have an account to use this service. Your backend will effectively amount to an API and a storage solution for CRUD users, CRUD 'interesting events', as well as an administrative dashboard for managing content.

How would you handle the geospatial nature of your data? How would you store images, both for long term, cheap storage and for short term, fast retrieval? What would you write your API in? What would be your database?

Your Answer:

How would you handle the geospatial nature of your data?

Utilize Cloud Computing to address big geospatial data. Big Data has emerged with new opportunities for research, development, innovation and business. It is characterized by the so-called four Vs: volume, velocity, veracity and variety and may bring significant value through the processing of Big Data. Cloud Computing has emerged as a new paradigm to provide computing as a utility service for addressing different processing needs with a) on demand services, b) pooled resources, c) elasticity, d) broad band access and e) measured services. Cloud Computing can be utilized to address Big Data challenges to enable transformation in many different cases, like climate studies, geospatial knowledge mining, land cover simulation, and dust storm modelling.

Our case is to handle pictures in different geographical locations. So we will use cloud computing to analyze and calculate the data. And Use Apache Hive and HDFS to store shared distributed data.

How would you store images, both for long term, cheap storage and for short term, fast retrieval?

For long term, cheap storage:

(1) use IndexDB. The IndexedDB API provides the browser with a complete database system for storing complex data. This can be used for things from complete sets of customer records to even complex data types like audio or video files. This IndexDB database is also not expensive for long term storage

(2) use Cache API. Some modern browsers support the new Cache API. This API is designed for storing HTTP responses to specific requests, and is very useful for doing things like storing website assets offline so the site can subsequently be used without a network connection.

For short term and fast retrieval:

(1) use cookies. Since the early days of the web, sites have used cookies

to store information to personalize user experience on websites. They're the earliest form of client-side storage commonly used on the web

(2) use ImageMagick. it is a very powerful image manipulation utility. It is a command line utility that essentially allows you to perform any sort of conversion imaginable on an image. We can run images through ImageMagick to have them compressed, resized, cropped,etc. Also, it's very convenient to upload photos. Also, ImageMagick is a very high powered, battle tested software for hardcore graphic manipulation. Graphically manipulation is very common in web development. We can use ImageMagick to resize, optimize, crop and combine images.

(3) use Web Storage. The Web Storage API provides a very simple syntax for storing and retrieving smaller, data items consisting of a name and a corresponding value. This is useful when you just need to store some simple data, like the user's name, whether they are logged in, what color to use for the background of the screen, etc.

What would you write your API in?

Use REST API. The most remarkable benefit to use REST API is that it can have the different pieces of software in our products to communicate, even if they are running on entirely different platforms and if some of your products are internal only.

In our case, there are two platforms: web server side and mobile application. So we should use REST API

What would be your database?

For Mobile application. Here we use Android as an example. If the data(pictures) are in small-scale, using embedded database SQLite. If the data(pictures) are in large-scale, we should use URLConnection or HTTPClient in doInBackground() method connect to outside database Like Oracle which can handle large-scale data.

For web server side, If the data(pictures) are in small-scale, using mysql database other wise use Oracle.

Quiz Score: **0** out of 100