News Buff:

Dataset, API

Backend CI/CD

Backend data etch architecture: new york times api, vaderSentiment-sentiment tool; node.js scriptingh,

Python for web server, flask web framework, swagger openAPI, github actions;

Message broker: rabbitMQ, docker, kubernetes, gcp

Frontend architecture: React, leaflet, MUI; NPM, webpack, netify.

Project management tool: pivotal tracker, slack, google docs , github, intellj,

Scale: 3 million

6 hoursfetaching, 8 hours bulk geo, 4 hours database queries

Challenges: bulk geocoding, restribtion on API

Future work: accessibility,

Headliners:

Thank you

Theorem Prover:

Compiler and interpreters.

Pixelbar:

25 cocktails from twitter

Architecture:

General:

Front end: EJS, CHART.JS, Bootstrap, css, highchart

Back end: Node.js, GCP Deploy, Docker CICD

API and dataset setup: Mongo, twitter

Challenges: connecting, Filter, software, library,

Collatz Travel Recommendations:

Intro:

Architecture: Priceline API, Google place API,

Intro

Demo

Archtiecture

2个report: stat + machine learning

1个project: big data

Indexing: Making one or more documents searchable by analyzing the documents' contents.

Stemming: Swimming -> Swim

Hadoop: is not real time system

Storm allows the creation and submission of topologies using any programming language

Spark can persist datasets in memory, instead of having to write to and read from disk

D3 gives you more control over your visualization and interactive behavior

POST /user

Elasticsearch and Kibana : To explore new problem spaces with data

Spark: useful for distributing dataset and computation, store data in memory, iterative with mapReduce, interactive queries, and streaming queries. Spark needs a cluster, interactive, Resilient distributed datasets, an immutable collection of partitioned data, a superset of mapreduce, Spark Transformation define data manipulation then action, persistence because in memory and no need to write to disk like mapreduce

Search: flexibility, cost. Indexing, user input, ranking, result display. Query Processing

**Streaming:** real-time in the order of milliseconds, message processing is clearly the fundamental paradigm for real-time computation. A type of data processing engine that is designed with infinite datasets in mind

**Cardinality**: bounded/Unbounded data, the infinite datasets imposes additional burdens on data processing frameworks that consume them

**Constitution**: Table or Steam. A holistic view of a dataset at a specific point in time. SQL systems have traditionally dealt in tables. An element-by-element view of the evolution of a dataset over time. The MapReduce lineage of data processing systems have traditionally dealt in streams.

**Event time**: The time at which events actually occurred. **Processing time:** This is the time at which events are observed in the system. Batch fixed windows for unbounded data. Streaming are built for unbounded data. Time agnostic, Approximation, windowing. Buffering, Completeness.

Data to information: Analytics. Hype curve. Elasticsearch + Kibana to explore the data. Pipeline in kubernetes. Validate early and often and dogfood your pipeline.

REST: representational state transfer, Client-Server, improve portability of the user interface, improve scalabiliy. Stateless: induce visibility, reliability, scalability. Visibility is the ability of a component to monitor or mediate the interaction between two other components. Reliability is improved because it eases the task of recovering from partial failures. Scalability is improved by allowing the addition of more servers that do not have to know client application state. Concerns may decrease network performance by increasing the repetitive data (per-interaction overhead) sent in a series of requests. Cache used to improve network performance, can be cached on client side or server side; data labeled as cacheable or non-cacheable. Caching can decrease reliability if stale data within the cache differs significantly from server data.

REST is defined by four interface constraints: identification of resource, hypermedia as the engine of application state, implementation are decoupled from the services they provide, which encourages independent evolvability. Manipulation of resource through representations.

Layered System, allow an architecture to be composed of hierarachical layers by constraining component behavior such that each component cannot see beyond the immediate layer with which thye are interacting. Code on demand, allwos client functionallity to be exatendede by downloading and executing code in the form of applets or scripts. Reduce visibility and thus is only an optional constraints within REST.

API should not dictate your architecture, it may likely inform it. If you are working with an existing system, establish your API as a façade. Think about the capabilites that need to be exposed to your client. The core processing unit accesses all resources through an address bus. Unix operating system: everything is a file. REST Design Tips model your process not the implementation design resources and represetations that reflect your business logic. Model your process not the implementation.

Critical process should be isolated to the Server, errors should reflect business logic problems

POST create new resource, run queries with large inputs

Get for safe ande idempotent information retrieval of resources.

Patch update resource in an idempotent, non-constrained way, do not use underly operation is not atomioc/non-transactional, if the input is constrined to a well-defined, unchanging representation.

POST /user {'firstName':'ddd', 'adress': {''''''}}

Get /user

Get /user/:id

/address

POST /user

GET /user

App server - > DB

-> kafka -> hadoop writer -> hadoop distributed file system (tweeter id, account id, )

import {Fire} from '@styled-icons/heroicons-solid'

const MarkerContainer = styled.div`

${Fire} {

color: orange;

}

<Fire>

{children}

</Fire>

"@styled-icons/heroicons-solid": "^10.36.0",

Icebox:

Load data from api into kafka

Get kafka runing. Setting

As a user of the website, I want to … code star

Fix the APU bug fixx

Write in a customer voice, feature,

Story points = complexity of story

Creating the webhook on GitHub, github

Iteration/Backlog

Add story: small Task

Validation loss stop decreasing then stop because overfitting

3/21 to 3/25 Spring break

5/4 last day

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Monday | Tuesday | Wednesday | Thursday | Friday |
| 12:20 - 1:10 | Stat2  Fleming Building 156 |  | Stat2 |  | Stat2 |
| 2-3:15 |  | Machine learning  Engineering Center ECCS 1B28 |  | Machine learning |  |
| 4:40 - 5:55 | Big Data  Duane Physics G130 |  | Big Data |  |  |
|  |  |  |  |  |  |

<https://drive.google.com/drive/folders/1yN3IjupEaiod7slbpykODkOmxfs2GQiH?usp=sharing>

Good evening, everyone. We are the Team firewatch working on wildfire. I'm Xingyu, This is RYAN, Sophie, chinmay, and philip. Our web application is a historical map of U.S and global wildfires, it also provide prediction service based on date and location. Users can filter the wildfires in certain conditions and check wildfire info in chart view. And each wildfire has weather, cause and duration.

Here is general information for our database, the u.s. one having 2 million rows and 37 features for the past 25 years wildifire in the u.s. The tow global DATASET, one from nasa with 1 million another for wilder date range with monthly wildfire

Then the key atrributes for prediction service, your have date, location, and weather related data.

And the data model but some of them are overfitting so it will be future work for improving the data model.

We also use recharts library for data visualization, we can check total fire per year, total acres burned per year, it seems it increaing yearly, and the wildifre by size class, we have most wildfire around 1 to 10 acres. Then the wildfire by general cause, the top three are natural, firearm and explosive use, and power geneartion transimission and distribution. The final graph is wildfire by day of year, it make sense when most wildfire incident occur during summer time

Clustering:

DBSCAN

Hierarchical clustering

GMM

Spatio-temporal approaches

Categories:

#Dimensionality reduction algorithms

Ensmeble methods

Semi-supervise learning

Gradient boosting algorithm - xgbooster

Neural network (CNN. RNN)

#Logistric regression

Naive Bayes