CS 5412 Cloud Computing: Project Plan

Yuhuan Qiu, yq56
Soobin Han, sh767
Helen Sun, hs667

It would be helpful if there's a project description up front so readers get an idea on what you are going to propose

Data

- Static Data
    - User Account Data: (username, password, preferences)
    - User preferences: (deliveryTime, regions, sentiment)
    - Credentials to the news source interface
- Real-Time Feed
    - News API: (source, author, title, description, url, urlToImage, publishedAt)

  We will have both static and real-time data in our system. Static data includes both user account data including their preferences, and credentials to any news sources we draw from. And we will have a real-time feed of news articles including meta information such as author, title, description, publication date.

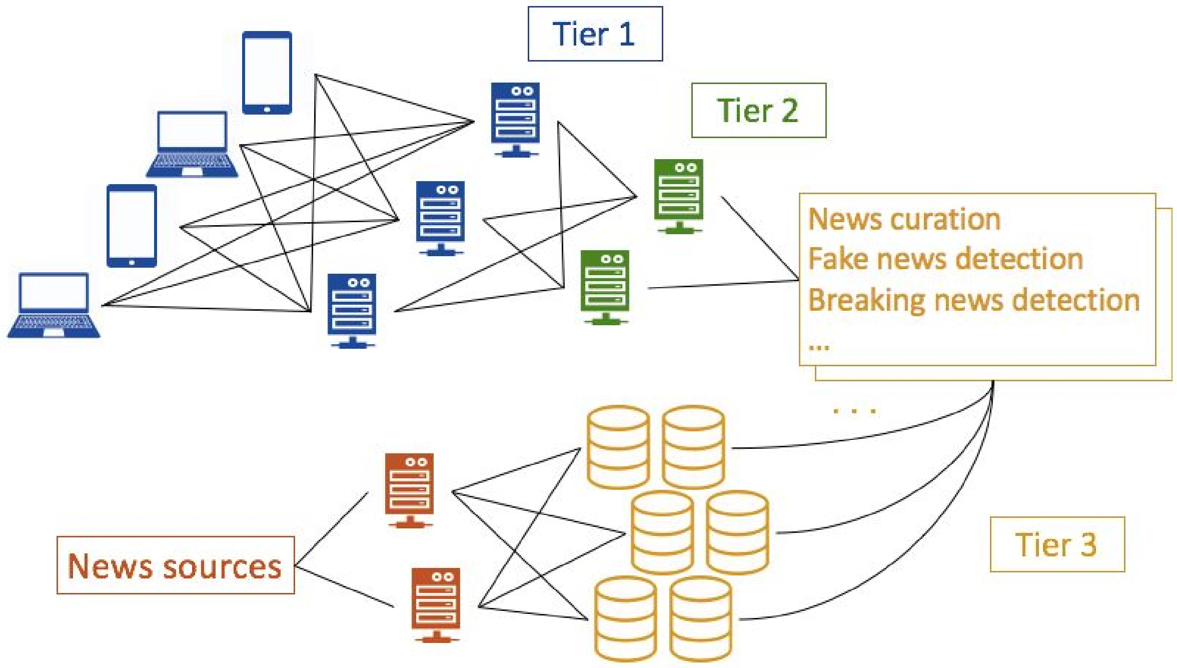
Interface

- Users makes request for news articles
    - We go to the second tier and backend to process the request. (username, regions, sentiment, categories) -> 2nd tier/backend -> [(source, description, url) , …]
- Users will send feedback on which articles they enjoyed and would like see more of, or articles they would like to see less of. Our system will collect this feedback and update our backend Models.

Is the 2nd tier equivalent to the edge server or the cloud? Do the backend models live on the cloud or the edge server? If the backend is on the cloud, then interaction with it will can be slow, and it would not be real time response to users' requests. If it's on the edge server, it is generally not recommended to put complicated ML models on the edge server because it increases computation time to train the model and training might require a huge amount of data to be stored.

When you say update backend models do you mean updating tier 3 storage or tier 2 cache?

Implementation/Architecture



Tier 1

- Interact with terminal devices for news requests and other API calls that the system provides
- Cache contents

Tier 2

- Provide additional caching mechanism

Tier 3

- Compute news recommendations, detect fake or breaking news, etc.
- Request news articles to news sources
- Have storage system for data

Evaluation

- Where are we going to evaluate the project?

    - We plan to evaluate our project by running instances in Amazon EC2 free tier.
- Experiments
    - Latency between making a query and getting an answer
    - Throughput: Number of queries we can support per unit of time or the capacity of our system.
    - Scalability: Throughput and latency improvements when we add additional nodes or instances. Does it make use of additional resources well?
    - Fault tolerance: Can our system continue functioning through node or instance failures in the first tier, second tier, or backend?
    - Partition tolerance: Can our system continue functioning through network errors causing partition

What kind of failures are you guys tolerating? Byzantine? Crash failures? Fail-stop?

How many failures are you planning to tolerate? If there are n nodes, tolerate n - 1 failures? n / 2 failures?

How would you simulate a partition in your system to test it out?

- How these experiments show that we have achieved our objectives?
    - Quality of Service is shown by latency (accuracy is also part of it but it has more to do with the ML aspect).
    - Capacity is shown by our throughput or load testing.
    - The scalability test demonstrates whether our system makes use of additional resources well. Does capacity increase? Does Latency remain i.e. below <100ms?
    - Fault tolerance and partition tolerance experiments show that our system can tolerate failures and partitioning

Milestones

We will be collectively working on all components of the project:

- Milestone 1 (3/6) Be able to interact in real time with News API i.e. make calls, store news article data.
- Milestone 2 (3/13) Have basic ML component for at least curation, potentially also fake news detection.
- Milestone 3 (3/20) Finish and streamline ML component, test effectiveness and accuracy. Be able to pass large amounts data from News API through ML component with relatively high accuracy and efficiency.
- Milestone 4 (3/27) Have a functioning backend, add database for users, and begin to add a caching layer for user requests for popular keywords or data sources. Complete intermediate report.
- Milestone 5 (4/17) Have a basic frontend to interface with the backend, be able to set preferences, make news requests, and update users and user information.
- Milestone 6 (4/24) Develop front end to be able to display article thumbnails and

description and user profiles. Be able to feed data from News API to ML to front end successfully.
- Milestone 7 (5/1) Evaluate our project and improve upon latency, throughput, scalability, fault tolerance, partition tolerance, if necessary and feasible.
- Milestone 8 (5/8) Make final improvements, evaluate our project, and begin final write up and presentation preparation.
- Milestone 9 (5/15) Finish write up and presentation, submit final project.