DesignDoc
Capstone Project: Tap-News

Author: Anqi Liu
V1.0: JUN 28, 2019
V2.0: JUL 16, 2019

Overview ：

Tap-News project is a full-stack system that allows registered users to browse personalized recommended-news fetched from multiple news-websites. The document covers details of the implementation of: web_server, backend_server, and several services based on SOA design from an engineering perspective.

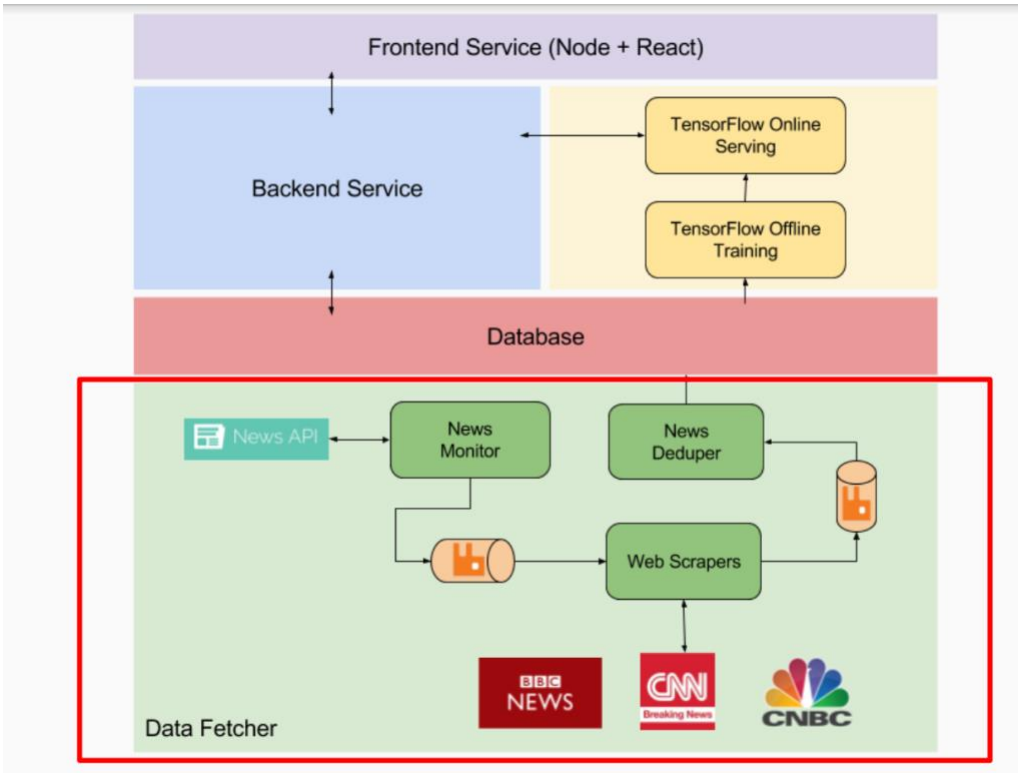
Major User Cases:

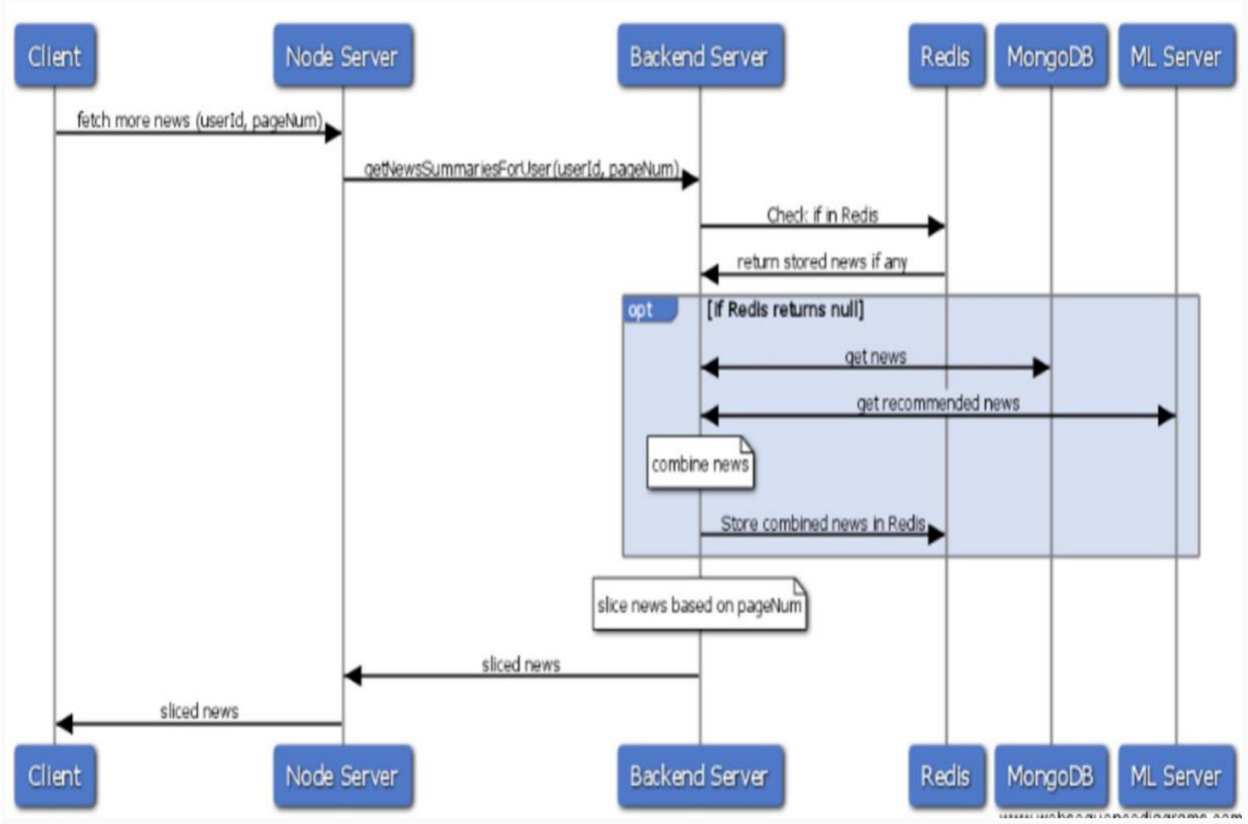
1. User can signUp and Login to the system in order to view news.
2. User can click on the news they are interested, and he would be re-directed to the news source for more details.
3. User can roll down their scroll-bar for more news on the web-page.
4. Different user see different recommended news based on the history of their click behavior.
5. User can search news by keywords.
6. User can click on tag-name and the web-page would display only the news with the same tag in order of date/time.

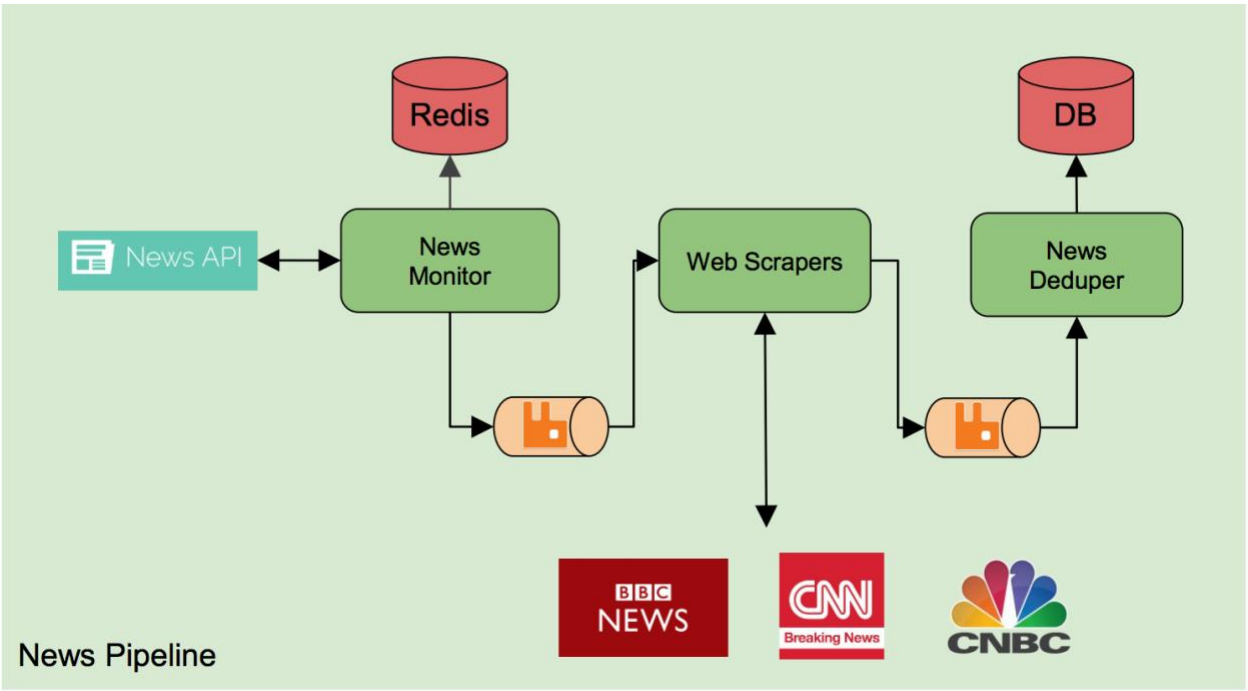
High-level stack diagram:

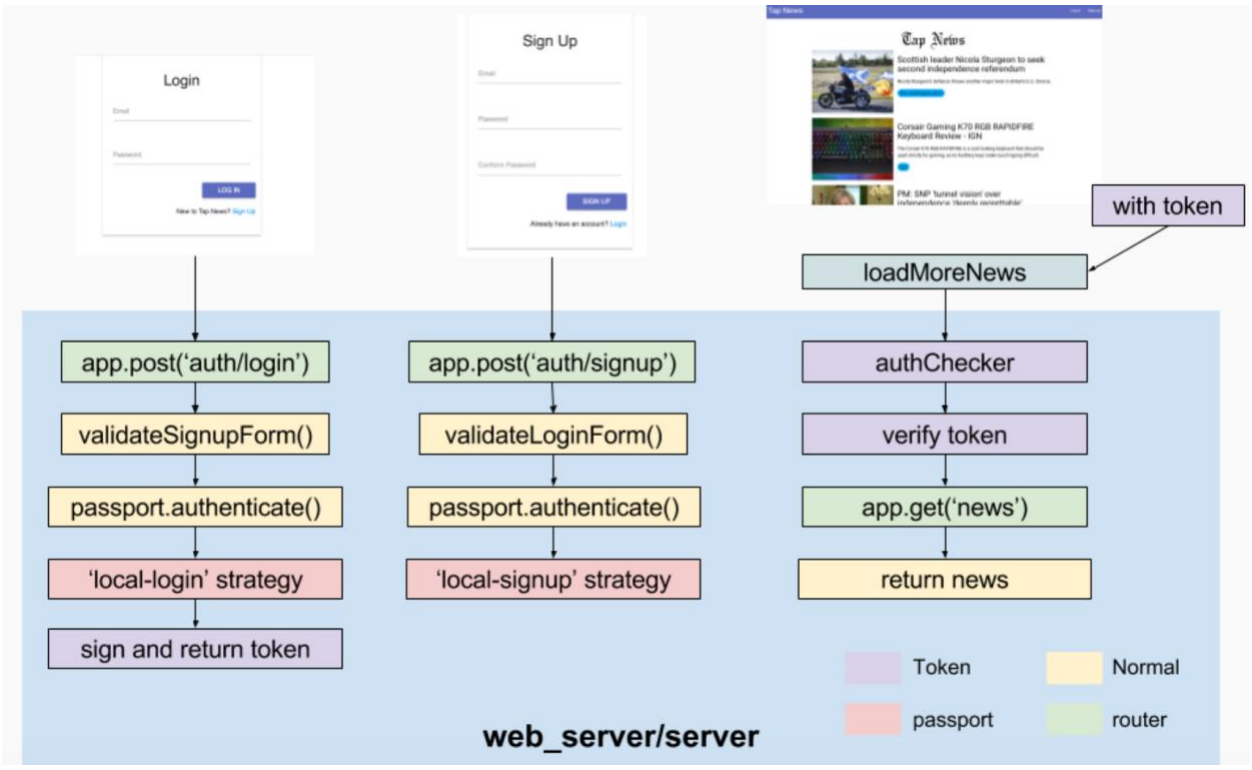
Stack	Technologies
Frontend - client	React, Materialize-css, Lodash
Frontend - web_server	Node.js(Express), MongoDB, JSON-RPC(Jayson), Bcrypt, Cors
Backend_server	Redis, CloudAMQP, RPC-service,
News_pipeline	Requests, Redis, CloudAMQP, Machine Learning(sklearn, TF-IDF)
Other_services (including: news_recommendation_service,	Tensorflow, Pandas, watchdog, xpath

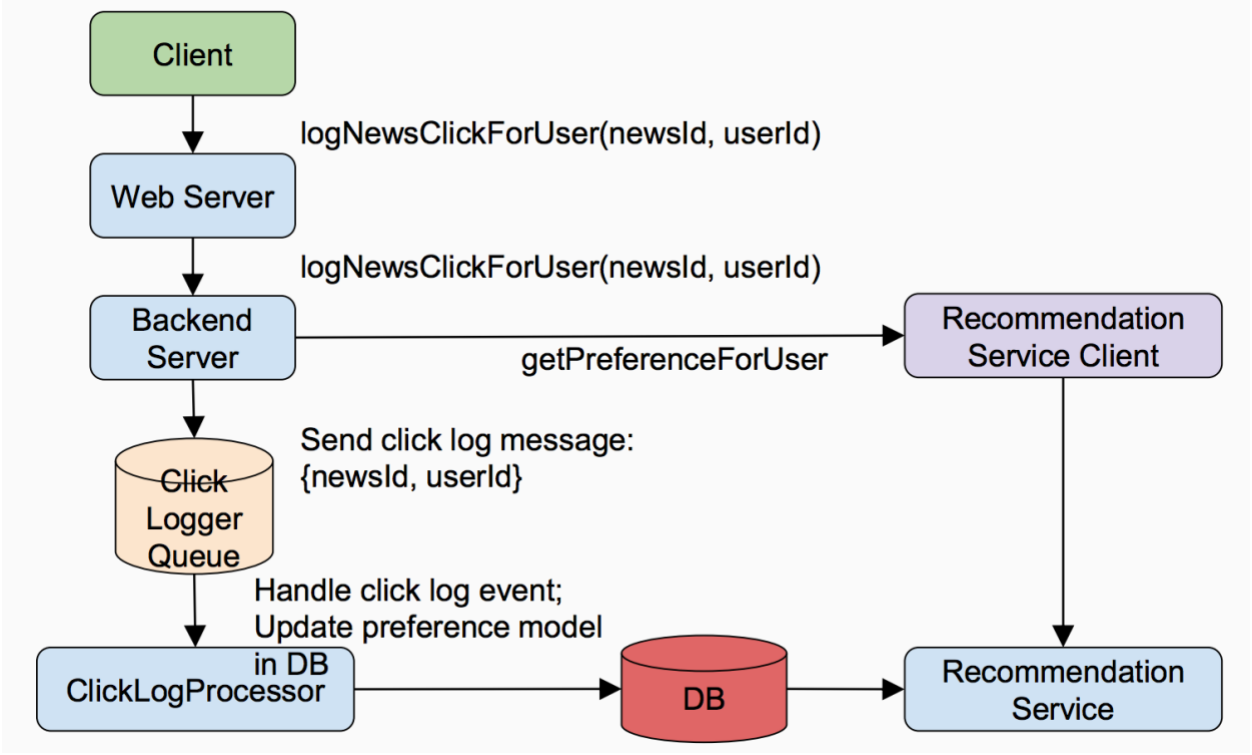
news_topic_modeling_service, config_service)	




Client
Node Server
Backend Server
Redis
MongoDB
ML Server
fetch more news (userId, pageNum)
getNewsSummariesForUser(userId, pageNum)
Check if in Redis
return stored news if any
opt
[if Redis returns null]
get news
get recommended news
combine news
Store combined news in Redis
slice news based on pageNum
sliced news
sliced news
Client
Node Server
Backend Server
Redis
MongoDB
ML Server


Redis
DB
News API
News Monitor
Web Scrapers
News Deduper
BBC NEWS
CNN Breaking News
CNBC
News Pipeline





Detailed design:

1. Front-end

Front-end is proposed to add three more feathers based on the existing Tap-news project:

- Today's New:

  Provide a "today's new" sectionsession displaying today's newly added news.

- Display news based on topics:

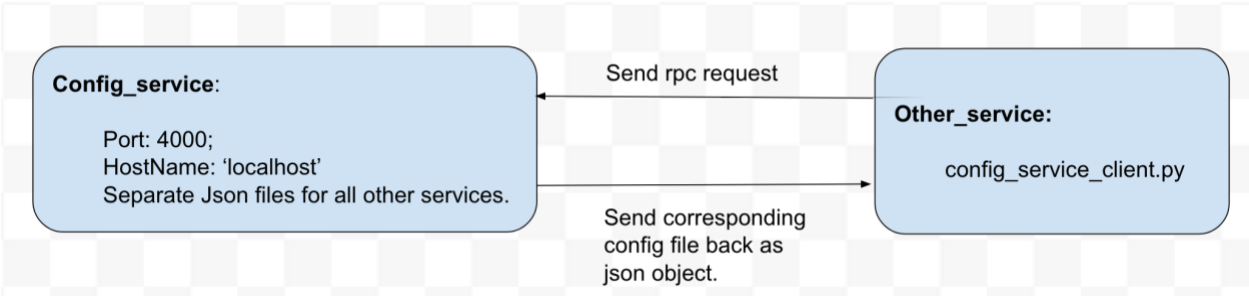
  When the user clicks a tag, we display all the news with this tag on top of the page. New user(registered and use the news-webpage for the first time) can click on the topic they like as a cold start. The top two news in each topic would be: (1) newest based on date/time. If multiple, show random one. (2) the news get clicked the most in the past 3 days. If none, show one of the newest news based on date/time. If multiple, show the newer one.

- Search bar:

  Provide a "search bar" for the users. Lucene or Elastic search is proposed to use here for searching news by keywords.

2. Create a new config_service to config all the modules and services.

In our project, the default package.json file could be replaced with a config_service. A new folder named config could be created, and each module/service should have a corresponding .json file in the config folder. In specific module, we create a new config_client.py file, and request config parameters from config_service via sending rpc request.



For web_server we would use a local config file instead of using the config_service. So in summary, config_service would be the config hub for all the

services except for the web_server. The web_server use a local config file for its configuration.

3. Apply machine learning to display personalized news list.

- Mark tags for news based on topics:

  Machine learning is used to label the topics for all the fetched news. Machine learning model would be accomplished with Tensorflow, DNN, NLP. The training results are to be visualized on with tensorBoard.

  One way is to utilize the click behavior of the user. By setting a threshold, we can compare the click frequency and/or time spent on a specific news, and then save the news above threshold as user's preference list. Google analytics could be used here to get the time user spent on a news. If the time is larger than the threshold, the preference adds weight.

  One thing that needs to be considered is the news topic distribution of our provided news. If we have a bias of the news topics in the first place, we may wrongly predict the user's prefered topic. Also, the news are going to be re-ranked every 24 hours.

- Store Search bar keywords for user preferences:

  A mongodb is used to store user information:{userid, keywords_searched[ ], clickedNews_category_rates[ ]}

  keywords_Searched is a queue to store the keywords the user searched. News which match the keywords better would be displayed for this user.

  Clickednews_category_rates would be used to store user preference. Time_decay_model is used to store the possibilities for each topic based on user's clicks.

- Recommended for you features personalization:

  Assume the user logged in his account. If we have either keywords_searched[ ], or clickedNews_category_rates[ ] not empty, then we display the top 1 news for him based on his search-word, or the most clicked news in his category/categories. Otherwise, this session is hided.

4. Add a complete Logging, and use a suitable Monitoring system to build a visual monitoring system to monitor key parameters and indicators

Logging data includes two parts: the logging written in local file, and the metrics for the tap-news running system. Logging for local file includes token, cpu usage, memory usage; while the metrics includes user information and server responses, for example, IP, login speed, requests/min, loading time, exceptions and debugging. Each step should include a timestamp.

Logging data would be collected through a plugin app (e.g. statsD and Collectd, *not decided yet*). Then the data is fed into carbon, which later written in whisper for long-term storage. User works with graphite web UI and the requested graph is constructed and shown with data from carbon and whisper. Graphite would display things like rabbitMQ status, operation system loading status.

Another thinking is to use TrakERR (https://trakerr.io/#/; https://github.com/trakerr-com/trakerr-python) to track and visualize logging and system key parameters.

5. Continuous deployment pipeline and load balancing.

Jenkins and gitlab would be used to realize continuous deployment. Nginx would be used for load balancing. Unit tests and end-to-end tests would be conducted.

For Nginx, after installing, two folders are created in `local/etc/nginx/: sites-enabled, and sites-available. In sites-enabled, we create a file named: load_balancer:`

```
upstream executor {
        server 127.0.0.1:5001;
        server 127.0.0.1:5002;
        server 127.0.0.1:5003;

}

server {
        listen 8080;
        server_name executor;
        location / {
                proxy_pass http://executor;
        }

}
~
```

And by including load_balancer into /nginx/nginx.conf, we can use the listed three servers to load balance http://executor. In our project, all the servers should have a Nginx load-balancer.

Scalability:

Docker is in consideration for scalability, since we can let multiple docker images running for our project.

References:

1. https://github.com/Houfeng/confman
2. http://www.jianshu.com/p/6427ce018e07
3. https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/35599.pdf information filter method. googleNews as example.
4. https://indico.io/blog/make-content-recommendation-better-with-machine-learning/
5. https://zhuanlan.zhihu.com/p/25328686 recommend by topic.
6. https://www.ibm.com/developerworks/cn/web/1103_zhaoct_recommstudy1/index.html today's recommendation; new for you.
7. http://www.wildml.com/2015/12/implementing-a-cnn-for-text-classification-in-tensorflow/
8. https://yq.aliyun.com/articles/39629
9. http://www.apache.wiki/pages/viewpage.action?pageId=10028733
10. https://graphiteapp.org/
11. https://www.howtoing.com/how-to-keep-effective-historical-logs-with-graphite-carbon-and-collectd-on-centos-7/