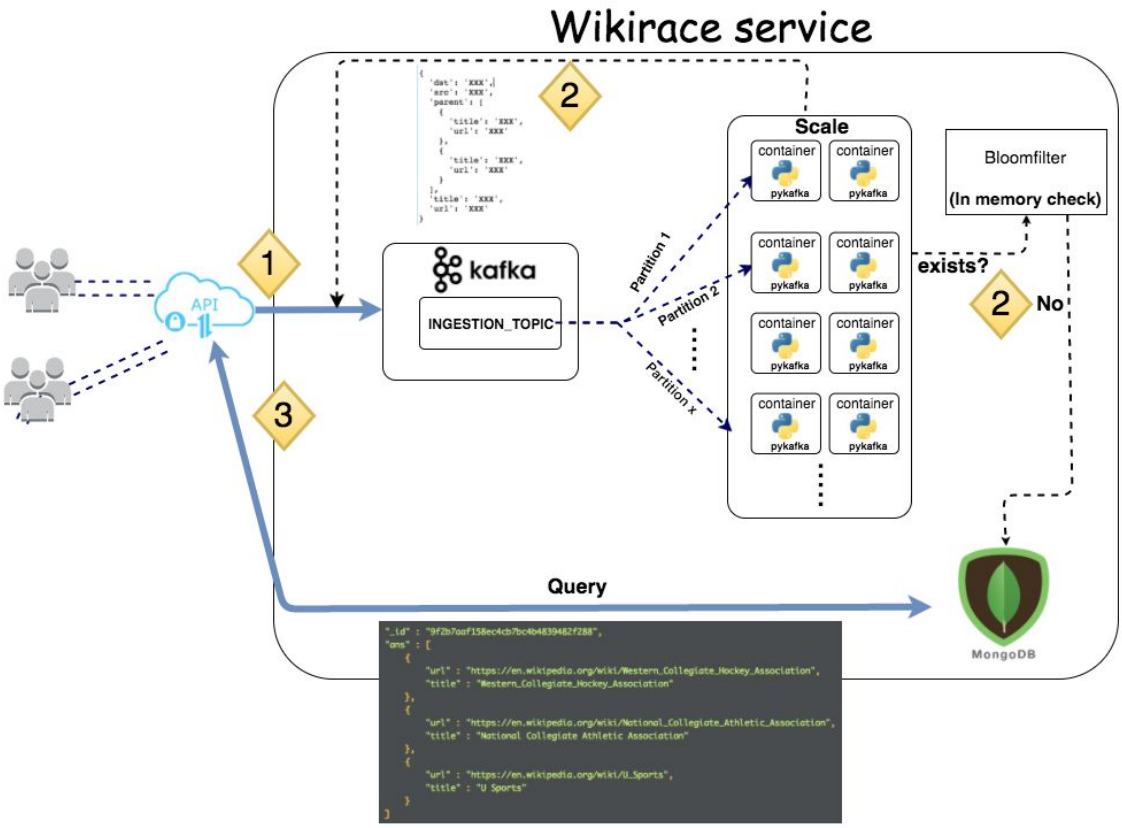
Architecture:

Following is an overview of my design solution for the gender service.



Technologies:

Apache Kafka:

kafka used two times in the above architecture:

1. To ingest input metrics so that each model can read the data separately. Moreover, any new model can read from the first offset(depending on retention policy) from kafka using a new consumer group.
2. To collect all the output results from each model.

Advantages:

- Kafka is massively scalable. There is no limitation on the volume of input data.
- Using Kafka We can push different stream of data by defining different topics and partition them logically for different solutions.
- Using different consumer groups each new model can start consuming json metrics from kafka
- By defining multiple partitions for kaka topics, number of pykafka wrapper can be started and respond. This increase the level of parallelism.

pyKafka wrapper: I have written a wrapper around pykafka to simplify push(produce) and pull(consume) from kafka. It serializes/deserializes the message(json) in python. This wrapper can also be replaced by spark streaming-kafka connector.

Mongodb: A database is used to persist the result. Also it is used as cache so once we find the path between two links we store it there. Therefore for the same subsequent request webservice replies using that result.

Flask: Flask web service is used to handle the API calls from the end user of the service by querying the outputs from mongo and send back the reply.

Python: Project code is written in python which is my comfortable language.

Installation and Run:

Please follow the instruction to install and run the service.

Requirements:

- Fill database connection info in configuration.py
- Fill Kafka broker IP in configuration.py

- **Installation:**
  - clone repository
  - go to cloned folder
  - install: `sudo pip/pip3 install .`

- **Run it:**

1) Run consumer:
   a) Python run.py

   Note: you can run multiple instances of the consumer on different processes.
2) Start the web service:
   a) Python webservice.py

Deployment:

To deploy, the repo will be containerize and pushed to docker hub. On deployed node those containers will be pulled and run independently. To increase parallelism run multiple copy of the container. You can use docker swarm or marathon to start many containers.