Exercise set: 5
Tommaso Carraro
student number: 015147592

Introduction to IoT: fifth set of exercises

First exercise

1. This is an event-based query that is triggered when the fire is detected by the sensor. This query returns the time, the event location and the maximum temperature over the different measurements performed by the sensors (the query takes 6 measurements in total, each measurement requires 600s to be performed) when the location of the event is critical.
2. This is a temporal aggregate query. This query returns the average pressure and the maximum pressure over last 10 seconds once every 2 seconds when the sensor is sampled every 2 seconds and the altitude of the measurement is over 1000 meters.

Second exercise

The choice is between TinyDB and Antelope data management systems, in fact both allow to perform data aggregation by grouping the tuples, implementing indexes and performing multiple queries simultaneously. I've decided to choose TinyDB over Antelope simply because the application requires the access to every sensor's features and TinyDB has support for query types that help to achieve this goal, in fact it allows to perform event-based query (this approach is really useful for the best integration between the data management system and the sensors' features), temporal aggregates and lifetime queries. Antelope is less powerful than TinyDB because it doesn't have support for these types of queries and it can't offer the best integration of the data management system with the sensors.

Third exercise

1. I would send the execution of the query to a big data framework that works on the server cloud. This consists in sending to the server cloud all the measurements performed by the sensors at the end of every hour. It is possible to use Apache hadoop to perform this job because it is made for batch processing, like in this case. In the big data framework it is possible to use the MapReduce algorithm to parallelize the computation for this query. In this case the job is to get the GPS coordinates of the location with the highest average temperature in Helsinki in one specific hour of the day. This job can be subdivided in many tasks and each task consists in computing the average of the temperature in a specific location in one specific hour of the day (the location of one of the 3000 temperature sensors distributed around Helsinki area). Using the MapReduce algorithm the client can send this job to the framework that will compute it in parallel. The tasks are distributed by the framework among the data nodes that compose the distributed network and at the end of the execution the result will be sent back to the client. In particular, there is a mapper for each of the 3000 sensors and its job is to compute the average of the temperature for that specific sensor, then there is only one reducer that aggregates all these averages together and returns the maximum one. Since the network of the nodes it is distributed it is possible to use a routing technique such as the semantic routing tree like in TinyDB to distribute the query along these different nodes. The node that

sends the query inside the network would be the gateway of the hadoop distributed netowork of nodes.

2. SELECT COUNT(*)
   FROM sensors
   WHERE location = (0, -78.467834)
   FRESHNESS 10 minutes
   EPOCH 1 hour
   RUNCOUNT 1 week
   In this query the COUNT operator is used to count the number of times that the sensor has collected data. With the WHERE condition we can retrieve all the measurements collected by a specific sensor that is located at the coordinates provided. The FRESHNESS condition is used to match the constraint that the records collected have to correspond to the last 10 minutes. The EPOCH condition specifies that the query has to be performed every one hour. Finally, the RUNCOUNT condition specifies that the query has to be performed for one week.

Fourth exercise

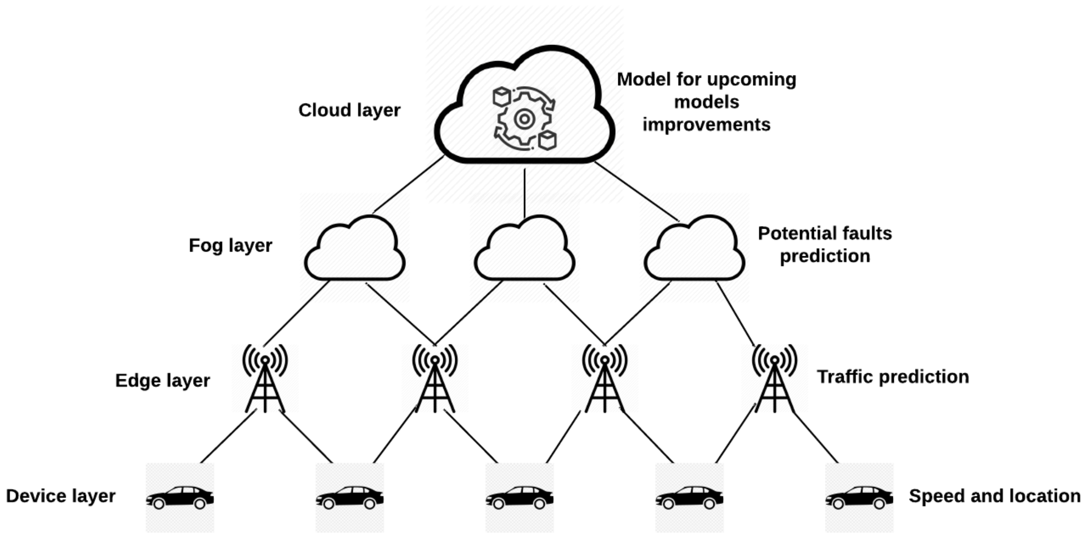
1. First of all we have to identify the differents operations that have to be performed by this application:
   a. when the car is asked to provide the current driving conditions such as speed, location and traffic, near real-time operations have to be performed because it is necessary to take the data that are continuosly streamed by the sensors to provide these informations in real-time. The speed and the location of the car can be computed directly on the device itself by reading the sensors data, while the informations about the traffic need to be computed on the gateway where a model to predict traffic conditions could be placed. This model should use the current location of different vehicles to predict correctly traffic conditions. The gateway should have enough computational power to perform the inference of this model. I assume that this model is deployed on each different base station and each different base station is a gateway where edge computing is performed. So, when the car needs to know about traffic conditions the request is sent to the nearest edge that takes the location of all the cars near its location and perform the inference of the model based on these data. This should be enough to predict traffic conditions in the area where the request has been generated.
   b. informing the driver of potential faults in the vehicle is an operation that needs to be computed in real-time, so it requires near real-time operations. To perform this operation current conditions of the car have to be used together to historical information. A model built on the base of historical data has to be used to predict potential faults given the current conditions of the car. This model could be placed in a fog node, so fog computing can be used to perform this job. The current information about the car conditions is periodicaly sent by the car to the nearest edge that sends this information to the fog node where the computation takes place. I decided to use the fog because the model requires a computational power that is higher than the previous model and because the fog allows to deploy the model in less places

instead of deploying it in every edge node of the network. Finally, every time we send data to the fog, the fog sends their to the server cloud where the model is improved based on the new data that arrive periodicaly. These data are really huge in size because they arrive from all the different vehicles of the manufacturer and this is the motivation to perform the training of the model on the server cloud. This is a batch processing task because it requires a lot of data that arrives periodicaly at ther server.

   c. using historical sensors data to evaluate the performances of the components in the vehicle to asses what could be improved in upcoming models is an operation that has to be performed periodaly over a huge amount of data that arrive from all the different cars of the manufacter, so it is a batch processing operation that must be performed on the cloud. The models to evaluate the performances of the different parts of the vehicles can be trained and also deployed on the server cloud, in fact in batch operations there aren't all the latency issues that have to be taken into account on the real-time operations.

2. Part of this question (where to execute jobs and tasks) has already been answered on the previous answer. In this answer I will explain only the types of jobs and tasks that I have identified in this application. The three main jobs of this application are:
   a. display speed, location and traffic conditions reading the informations provided by the sensors. The tasks of this job could be:
      i. display speed and location in real-time using sensors data (where: on the device itself);
      ii. display traffic conditions using the infomations about the location of the nearest cars (where: inference in the edge and training in the cloud).
   b. predict potential faults in the car using historical information and current conditions of the car. The tasks of this job could be:
      i. predict potential faults on the wheels of the car using historical information and current conditions of the wheels (where: inference in the fog and training in the cloud);
      ii. predict potential fault on the brakes of the car using historical information and current conditions of the brakes;
      iii. the other tasks are related to all the different parts of the vehicle and a specific model for each of these specific parts should be used.
   c. evaluate the performances of the different parts of the vehicles to improve the upcoming models. The tasks of this job could be:
      i. the tasks are really similar to the tasks explained for the previous job but the outputs of the models are slightly different because the models should suggest what it needs to be improved on the car and how and not to predict potential faults (where: inference and training in the server cloud).

This is the free form diagram that has been requested.



3. I explain each of the three operations using nearby vehicles instead of the remote infrastructure. I assume that the models are already trained in a remote location and then deployed on the vehicles because the training of a batch processing model could take forever to execute in a distributed way using cars (latency, connection loss and data replication problems, but also low computational power):
    a. <u>display speed, location and traffic conditions reading the informations provided by the sensors</u>. The speed and the location are displayed using the computational power of the device itself, so there is no need to use nearby vehicles. For the prediction of the traffic conditions the car could connect to the nearest car and this second car could connect to its nearest car like in a chain. Then, we can use the informations about the location of the cars in this chain to try to predict traffic conditions. The model has to be deployed on each car and should be executed in parallel in a distributed way in such a way that it is possible to use an horizontal scaling to reach the computational power that is required for this operation.
    b. <u>predict potential faults in the car using historical information and current conditions of the car</u>. Since we have assumed that the model is deployed on each vehicle we can execute it in a distributed way like in the previous answer giving in input the current car conditions detected by the sensors. If we want to try to train or improve the model using only the nearby vehicles we should extend their computational power and memory to allow them to store all the historical data that are required to compute this operation. This means that we should use vertical scaling on each car and horizontal scaling using the distributed network of cars.
    c. <u>evaluate the performances of the different parts of the vehicles to improve the upcoming models</u>. The answer is the same of the previous point.

First bonus exercise

It is possible to find the requested code in the file “bonus1.txt” that I have attached to the e-mail.

Second bonus exercise

1.
    a. The first application I have selected is sleep monitoring using a set of different sensors. The application is able to check irregularities on the sleeping and suggest to doctors which problems there could be and their solutions. The patient has to stay one night at the hospital to be monitored properly. This is a warehouse application because it uses a lot of different sensors that work in a limited area and send their data to the gateway periodically. At the end of the night it is possible to use big data frameworks such as Apache hadoop to perform the batch processing related to this application. In this application the energy overhead of the warehouse approach it isn't a problem because all the sensors are directly connected to the main power supply of the hospital, so there aren't power constraints.
    b. The features of this application are:
        i. check breathing irregularities during the night using a breathing sensor;
        ii. check hearbeat irregularities while the patient is sleeping using an heartbeat sensor;
        iii. check brain pulses during the night using a brain sensor: this should help to understand if the problems on the sleep are related to the anxiety of the patient;
        iv. there could be other sensors to provide other features but I'm not really informed on this application.
    c. These are the 5 V's in this application:
        i. Value: more sensors we add to this application and more features we are able to provide more value will be generated by the application because it can become more accurate. In this application there could be the monetary value of data because the data generated by the patient could be sell to medical industries to create new types of medicines to help the sleep (I assume there aren't privacy issues). Then there is the business value because with this application we are able to identify patterns that we weren't able to identify before and with this patterns we can help the patients to sleep better.
        ii. Variety: in this application the variety is related to the fact that there are a lot of sensors that send data to the gateway. These sensors use different types of data formats so we need to deal with all these data formats and try to integrate them.
        iii. Volume: in this application the volume is related to the fact that these sensors work all the night and could generate a huge amount of data that then have to be managed by a big data framework.
        iv. Velocity: in this application we are talking about only one room where the patient is monitored so we can't say that the data arrive so fast. The velocity could be high if different patients around the world are monitored at the same time generating a huge amout of data.

v. Veracity: in this application the veracity is related to the fact that is really difficult to combine data of different formats that arrive from different sensors. This could turn into errors and uncertainty in the results of the model that should suggest how to help the patient to sleep better.

2.

a. The second application I have selected is smart city air quality checking using sensors to check the status of the air and to suggest to people the places they should avoid cause of the pollution and in which hours they can't go in these places. The application is also able to check the air quality in a specific location in real-time when a user asks for this information. This is an application that uses a distributed approach because the power of these sensor devices is low (the warehouse approach is subject to potentially high energy overhead) and there could be the necessity to query a specific sensor in the distributed network if we need to know about the air condition in a specific location (in the warehouse approach all the sensors send periodicaly the data to the gateway instead to directly answer to a specific query). Another motivation to choose the distributed approach for this application is because in the warehouse approach there are difficulties to guarantee freshness of measurements because there is a limited control over sampling intervals, but in air quality application it is necessary to know the perfect time when a measurement takes place because we need to provide to the user accurate hours and places where they can go to avoid the pollution.

b. The features of this application are:
   i. real-time air quality checking in a specific location when a user asks for this information;
   ii. send useful notification to the users in a specific location when pollution has been detected by the sensors in that area;
   iii. air quality prediction for the upcoming days based on historical data and current sensors records.

c. These are the 5 V's in this application:
   i. Value: more sensors we add to this application and more accurate will be the air quality predictions and more value will be generated by the application. In this application there could be the monetary value because users are interested in buying this service if it performs well. Then there is the business value because with this application we are able to know information about the air quality that can help the people to live better and avoid getting sick.
   ii. Variety: in this application the variety is related to the fact that the same data are used to provide different services and not only one. There isn't the variety related to the different formats of the data because all the sensors generate the same format of data.
   iii. Volume: in this application the volume is related to the fact there are many sensors (more than 1000) placed around the city and since they are constantly working they generate a huge amount of data.

iv. Velocity: in this application the velocity is related to the fact that since we have a huge amount of sensors around the city and they are constantly working, the data arrive very fast.

v. Veracity: in this application the veracity is related to the fact that there could be errors in the sensors calibration or measurements. This could turn into uncorrect predictions or information about the air quality.