Table of Contents

1. Introduction

1.1 Background

The airborne PM2.5 has returned to become a serious issue since the start of 2020. The thickening smog, exceeding the standard safe level, prompted The Royal Thai government to approve measures to prevent and address the on January 21, 2020.

PM 2.5 comes primarily from combustion - Fireplaces, car engines, and coal- or natural gas–fired power plants are all major PM 2.5 sources. PM stands for particulate matter (also called particle pollution): the term for a mixture of solid particles and liquid droplets found in the air. Some particles, such as dust, dirt, soot, or smoke, are large or dark enough to be seen with the naked eye.

Fine particulate matter (PM2.5) is an air pollutant that is a concern for people's health when levels in air are high. PM2.5 are tiny particles in the air that reduce visibility and cause the air to appear hazy when levels are elevated.

To measure this level in air, sensor PM2.5 (PM2.5 refers to particles that are 2.5 microns or smaller in diameter) can be placed at different places where the chances of such issues. This sensor uses laser scattering to radiate suspending particles in the air, then collects scattering light to obtain the curve of scattering light change with time.

1.2 Purpose of the system

In this contemporary environment air pollution or air quality have become a big concern as the quality is degrading exponentially. One of which is the PM2.5 crisis where the size of particles is directly linked to their potential for causing health problems. Fine particles (PM2.5) pose the greatest health risk. These fine particles can get deep into lungs and some may even get into the bloodstream. Exposure to these particles can affect a person's lungs and heart.

On a very clear and non-hazy day, the PM2.5 concentration can be as low as 5 µg/m3 or below. But the PM2.5 is considered unhealthy when it rises above 35.4 µg/m3.

Our motive for this project is to analyze the air quality data from the sensor and give a clear vision in a way of dashboard. Therefore, we are going to develop the web app which will show all the required details related to air quality and accordingly people can take preventive measures.

For this we will receive sensor data from nodes or stations with different types of data such as text value and number and process using Big Data Tool.

1.3 Design goals

Air Quality is a big question in these times and to reduce the air pollution, we are looking forward to making an air quality monitoring system. This system will be taking the data from sensors such as PM2.5. In case of AIT, this application can be used to monitor the air quality on campus. Once the application can give accurate and real time information on air quality from the data imported from sensors then other investors or different organizations can use it.

This web application will be using Hadoop and Apache Kylin as a backend to hold a huge amount of data from sensors, and they will be processed with BI tools and different important insights are visualized using an interactive dashboard. As of now, we will be starting with Java Spring Boot framework for developing the system.

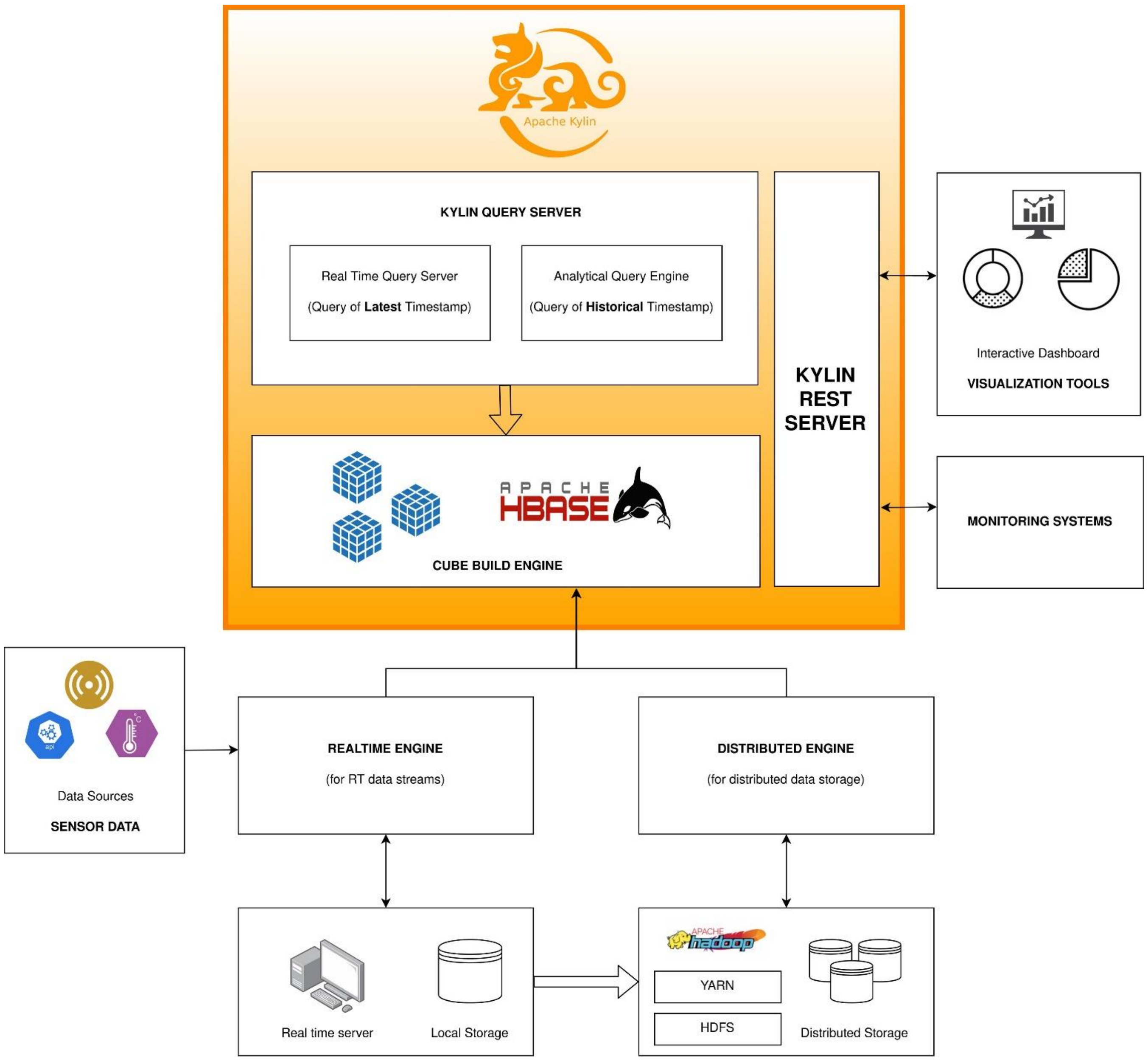


Figure 1. System Architecture

2. Functional Requirement

2.1 Visualization Module

- The end users should be able to see an interactive dashboard of air quality monitoring with different forecasts and insights.
- The system should be able to stream real-time data from different nodes and stations.

2.2 System Admin Module

- The admin should be able to login, logout to the system and modify the system parameters and toggle dashboard controls.
- The admin should be able to register new sensors and manage the sensors in system.
- The admin should be able to generate reports of specific time periods and export those in varies formats.

2.3 Data Collection Module

- Data will be extracted from sensors and stored in Hadoop which is acting as data warehouse using Hive.
- Kylin does aggregation functions on cube (HBase) and provide the required parameters to the system.

Note:

❖ Air quality and location details provided to the system from the sensor shall be stored in the database (Hadoop).

❖ The information shall be accessible via distributed system and JPA (Java Persistence API).

❖ The data stored should be able to be manipulated through interface.

3. Non-functional Requirements

3.1 Performance

The system shall be designed with high performance level to handle concurrent or multiple information from sensors which are placed at different locations in real time. Large numbers of data collected from the sensor should be displayed on the responsive web application and we need to focus on concurrency, response time and block time.

✓ The application should respond to a user within 2 seconds.

✓ The application should be able to handle 100 transactions per second in the peak load time.

✓ The application will be available with the uptime of 95% between 6:00 am to 1:00 am.

3.2 Reliability

Reliability is one of the key attributes of the system. Back-ups will be made regularly so that restoration with minimal data loss is possible in the event of unforeseen events. The system will also be thoroughly tested by all team members to ensure reliability.

- ✓ The system should be able to restore backward data of 24 hours (maximum 3 Days) within 2 hours as a recovery function.

3.3 Portability

The system shall be designed in a way that allows it to be run on multiple computers with different browsers. As it is a web application, mobile phone web browsers can also access the application.

- ✓ The web app must support latest Web browsers for any OS.
- ✓ The web app should be responsive.

3.4 Usability

Once the system is deployed, the maintenance of the system including tasks such as monitoring the system, repairing problems that arise, updating or upgrading software components, should be easy to be sufficiently performed by any person with a basic understanding of the dashboard system.

- ✓ The web app should be easy to operate by users with a certain navigation menu or option. No need for a user manual.

3.5 Interoperability

For the application, we just import and integrate various information with different values into the system from the sensors. Therefore, we need to know to segregate the data and proceed forward.

3.6 Scalability

With data, the storage size will increase but can be managed with time. This app can be made horizontally scalable when there are issues of memory storage.

3.7 Reusability

The system should be designed in a way that allows the database to be re-used regularly for the various similar sensors that the organization shall hold.

4. System models

This section presents a list of the fundamental sequence diagrams and use cases that satisfy the system's requirements. The purpose is to provide an alternative, "structural" view of the requirements stated above and how they might be satisfied in the system.

4.1 Use Case Model

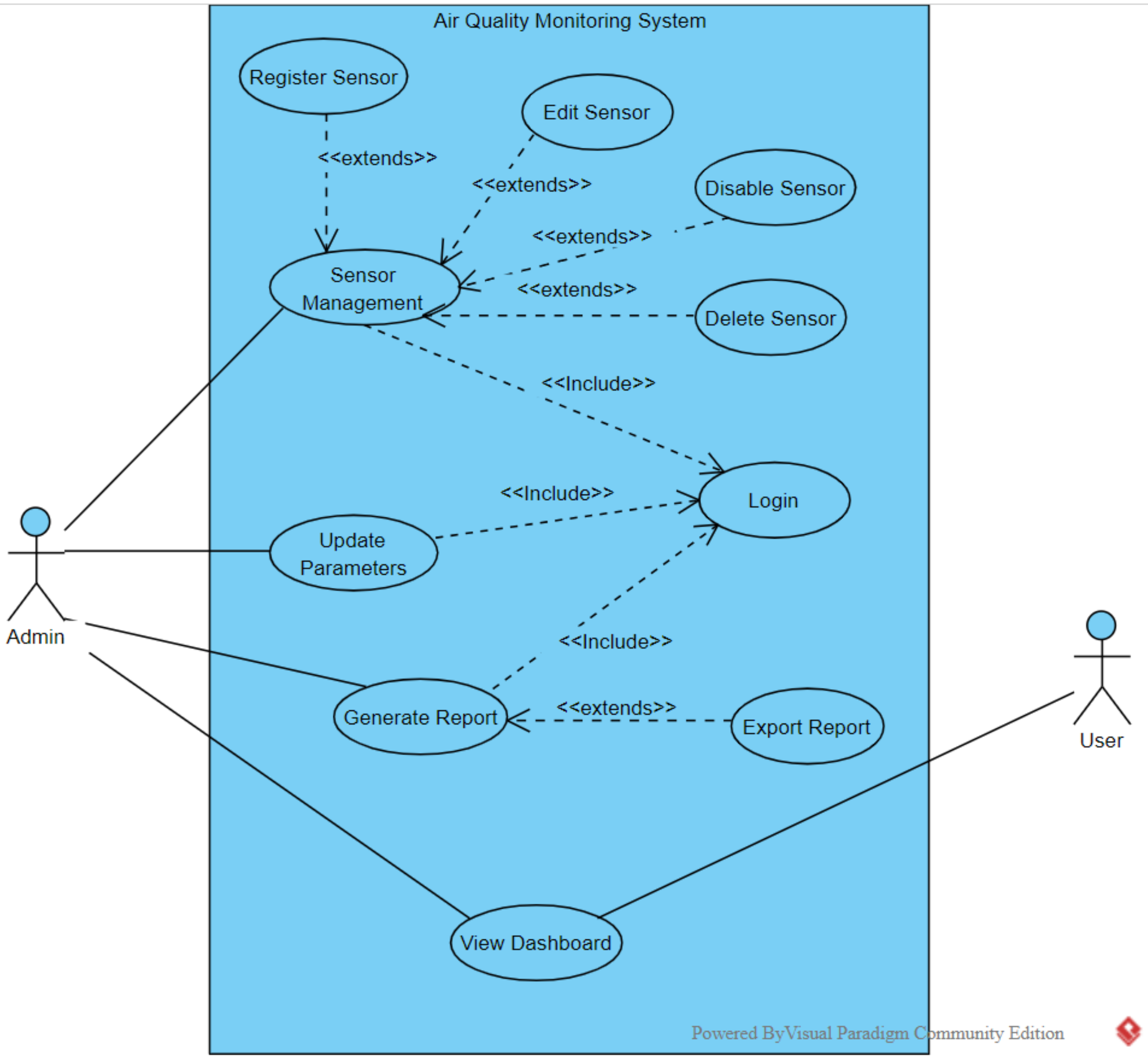


Figure 2. Use case diagram.

4.2 Dynamic models

4.2.1 Sequence Diagram

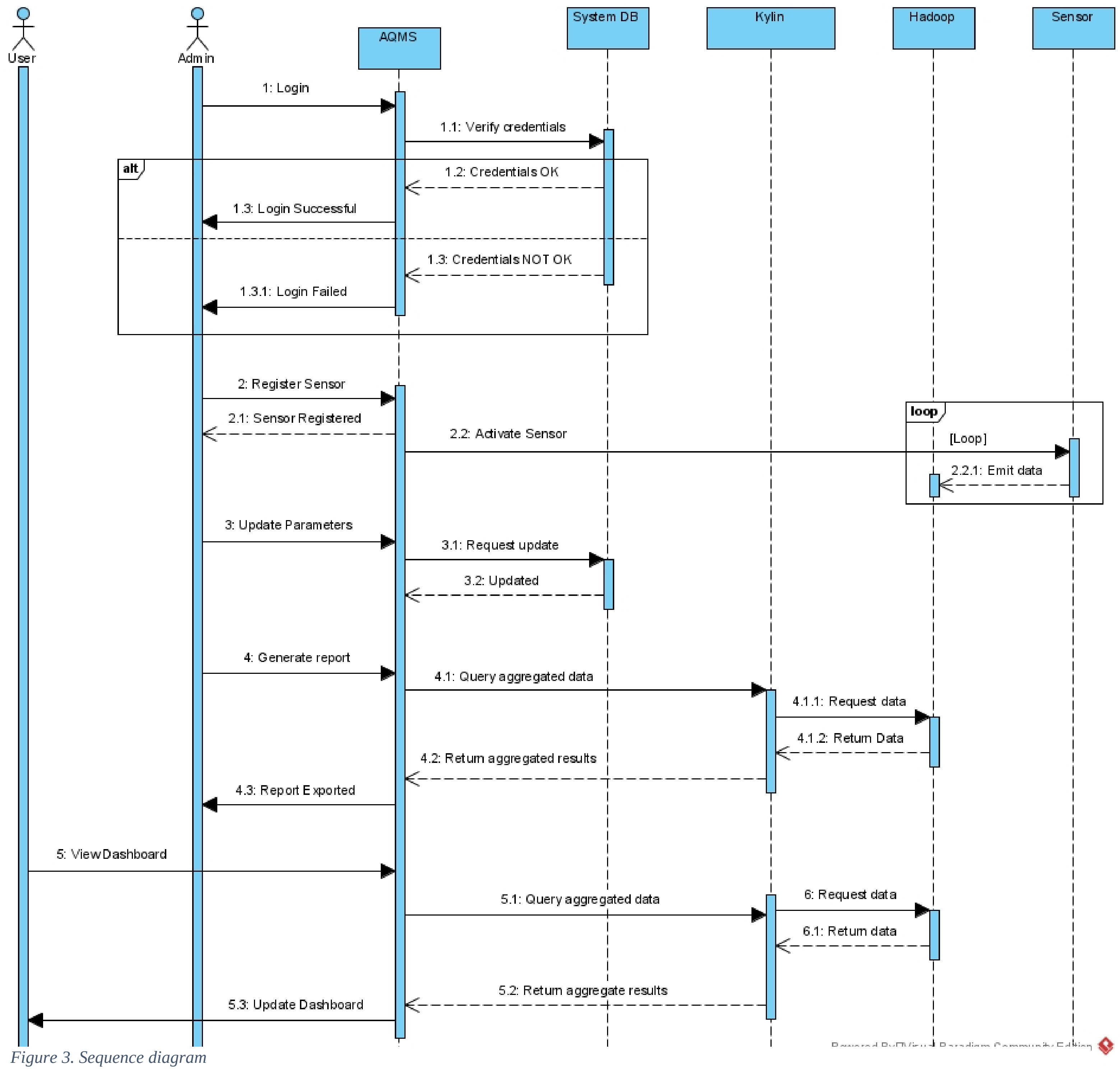


Figure 3. Sequence diagram

4.2.2 State diagram

a) State diagram for user with internal process.

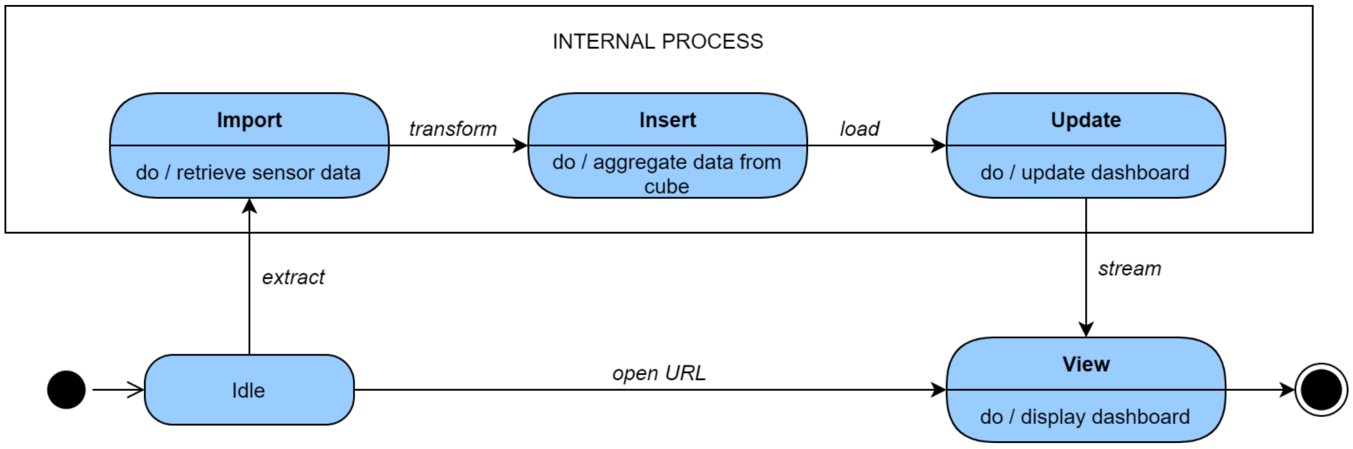


Figure 4. User State diagram

b) State diagram for admin with different state.

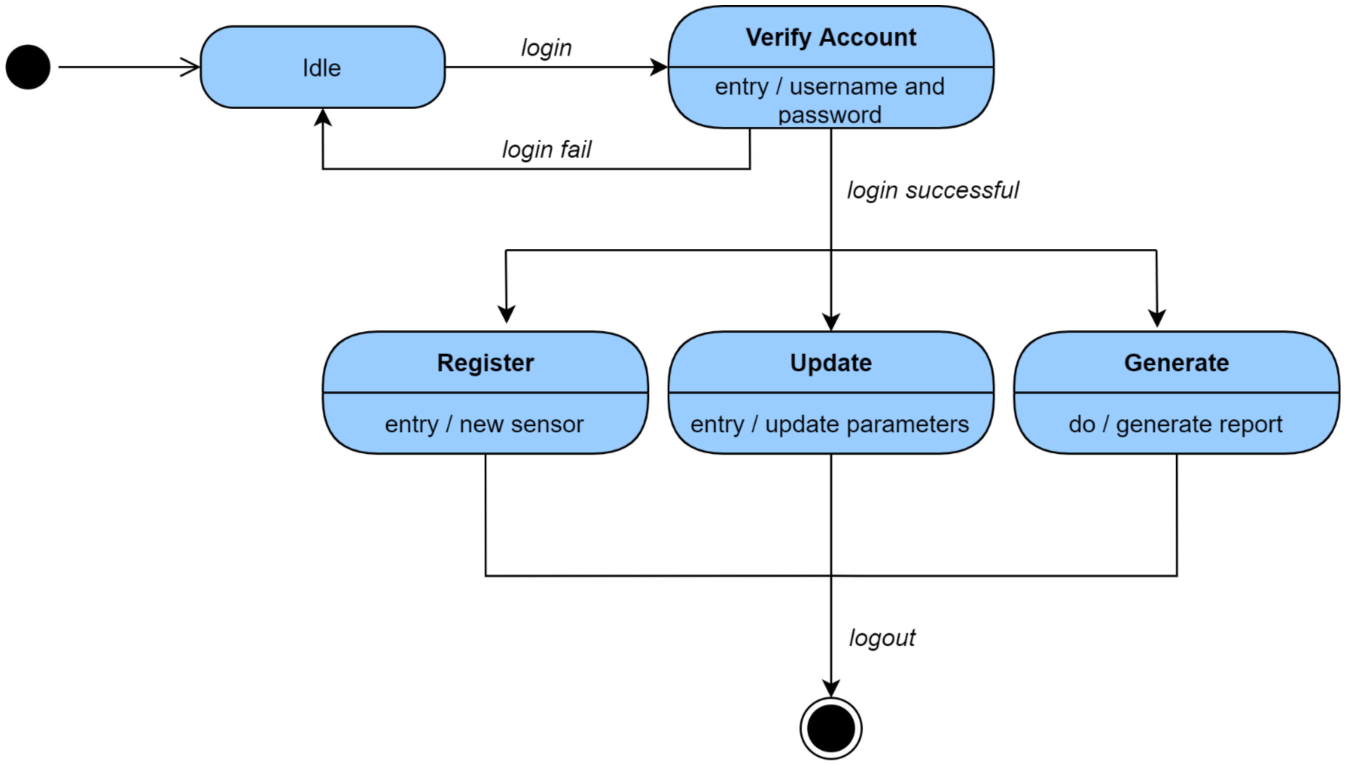


Figure 5. Admin State diagram

4.2.3. Activity diagram

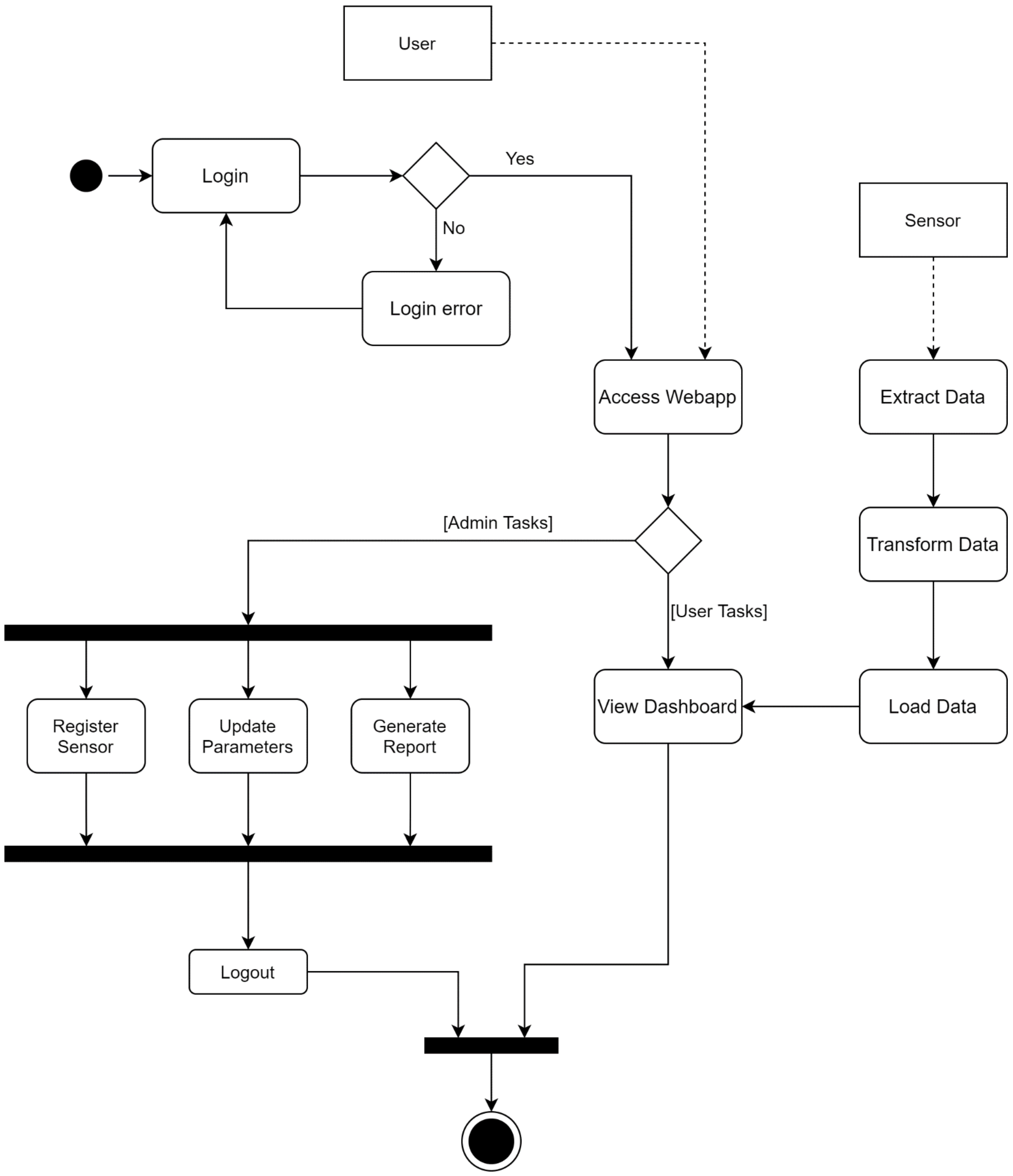


Figure 6. Activity diagram

4.3 Object and class model

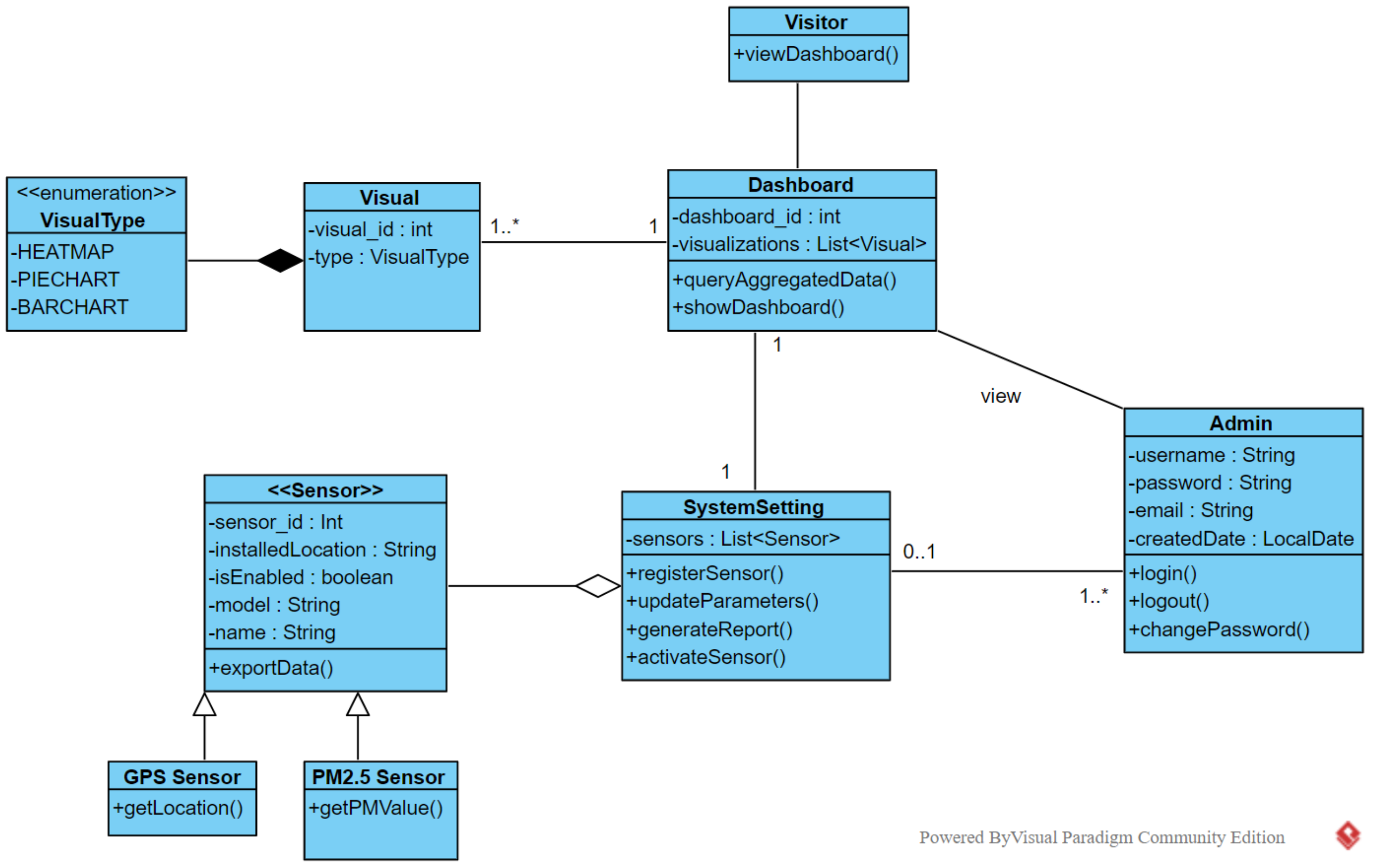


Figure 7. Initial class diagram

5. Software architecture

5.1 Subsystem decomposition

After analysis, we reduce system complexity while allowing high coherence dependency among classes where classes in a subsystem perform similar tasks and are related to each other via many associations. Coupling measures dependency among subsystems, so we made low coupling since a change in one subsystem does not affect any other subsystem. If high coupling is there, then changes to one subsystem will have high impact on other subsystems.

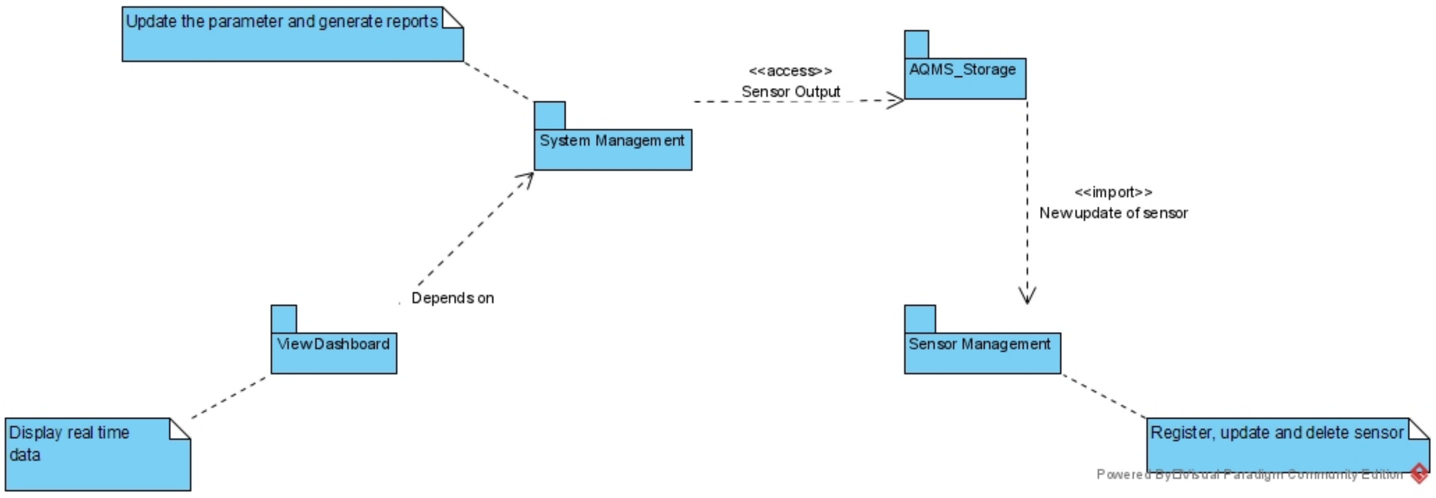


Figure 8. Subsystem decomposition

5.2 Hardware/software mapping

5.2.1 Deployment Diagram

Deployment diagram comes under structural diagrams that is used in modeling the physical aspects of an object-oriented system. Based on our project, we will use various software elements like login system, admin panel, dashboard, manage system as nodes and show the communication association between nodes.

5.2.2 Component Diagram

Component diagrams are used to visualize the organization of system components and the dependency relationships between them. They provide a high-level view of the components within a system. For our system, we will have software components like database, user interface, admin, user(public) as component diagram with interfaces between components to show how components are wired together and interact with eachother.

5.3 Persistent data management

Some objects in the system model need to be persistent and values for their attributes have a lifetime longer than a single execution.

A persistent object can be realized with one of the following mechanisms:

1. Filesystem
2. Database

For our project we will be following Filesystem since data are used by multiple readers but a single writer (admin).

For Database, we might use it when data are used by concurrent writers and readers. UML object models can be mapped to relational databases by:

- Each class is mapped to its own table
- Each class attribute is mapped to a column in the table
- An instance of a class represents a row in the table
- One-to-many associations are implemented with foreign key
- Many-to-many associations are mapped to their own tables

5.4 Access control and security

- ✓ Discusses access control

  The only admin login is required for the setup of the parameters and other dashboard page is available to everyone who visits the website.
- ✓ Describes access rights for different classes of actors

  The dashboard will be accessed by all the users who visits our website. Although the system settings will only be accessed by admin. Admin can set the parameters for dashboard.
- ✓ Describes how object guard against unauthorized access.

  If the normal user tries to access through login they will not be until they have admin rights (have used the spring security).
- ✓ Does the system need authentication?

  Admin needs the authentication with username and password to access and control the system.

Access Rights

Classes

Table 1. Access Matrix

Actors

	Dashboard	**System Setting**
Admin	viewDashboard () generateReport ()	registerSensor () updateParameter () activateSensor ()
Users (Public)	viewDashboard ()	

5.5 Boundary conditions

5.5.1 Initialization

- What data need to be accessed at startup time?

API will be called to fetch the sensor data to display real time at the dashboard. The data could be real time data or the visualization from aggregated queries.

- What services have to registered?

Sensors has to be activated and streaming APIs needs to be initialized to receive real-time updates. For data storage Hadoop server has to run which receives and sends data periodically to Kylin servers for cube building. Apache Kylin REST API will be used to perform aggregated queries as required.

- What does the user interface do at start up time?

It displays the latest updated values of database for PM2.5 sensor in the dashboard. If sensor streaming is active, it shows the real-time stream of the data values otherwise dashboard with aggregated query results.

5.5.2 Termination

- How are updates communicated to the database?

System is going to get the updates at specified time intervals.

5.5.3 Failure

- How does the system behave when a node or communication link fails?

It will retry to connect the call and fetch the data.

- How does the system recover from failure?

We will not have such failure, or the system recover is automatic since we are using Hadoop and it is having fault tolerance feature.

6. Subsystem services (In progress)

The middleware subsystem services data requests between the user interface and the database subsystem.

7. Low level design (In progress)

7.1 Final object design

7.2 Packages

We can use the UML package mechanism to organize classes into subsystems.

7.3 Class Interfaces

8. Glossary & references

8.1 Glossary

Here are some domain terms we are using in document:

- **Data Warehouse**: a data warehouse (DW or DWH), also known as an enterprise data warehouse (EDW), is a system used for reporting and data analysis.
- **Business Intelligence**: Business intelligence (BI) is the set of techniques and tools for the transformation of raw data into meaningful and useful information for business analysis purposes.
- **OLAP**: OLAP is an acronym for online analytical processing
- OLAP **Cube**: an OLAP cube is an array of data understood in terms of its 0 or more dimensions.
- **Lookup Table**: a lookup table is an array that replaces runtime computation with a simpler array indexing operation.
- **Dimension**: A dimension is a structure that categorizes facts and measures in order to enable users to answer business questions. Commonly used dimensions are people, products, place and time.

8.2 References

Apache Kylin. (2015). Bring OLAP back to big data! Retrieved from Apache Kylin | Analytical Data Warehouse for Big Data

Fann,N.,& Risley,D. (2011,January 5). The public health context for PM2.5 and ozone air quality trends. Air Qual Atmos Health 6, 1–11 (2013). https://doi.org/10.1007/s11869-010-0125-0

Geetha,S.M.N. (2021, March 19). Hadoop for Analyst-Apache Druid, Apache Kylin and Interactive query tools. Retrieved from https://www.saigeetha.in/post/hadoop-for-analysts-apache-druid-apache-kylin-and-interactive-query-tools?fbclid=IwAR0RRXXxKmv8onswnS-g5mV5Hh_L5R9zOSWly6YO8d4kb6oYYW4rrjF5wlo

Gupta,A.k., & Johari,R. (2019). IOT based electrical device surveillance and control system. International Conference on Internet of Things: Smart Innovation and Usages (IoT-SIU), https://doi.org/10.1109/IoT-SIU.2019.8777342

Nethu, M.V. (2018, September 25). OLAP on Hadoop-Apache Kylin. Retrieved from https://medium.com/@mvneethu90/olap-in-hadoop-apache-kylin-bf0377d8b44f

Sinha, S. (2016, October 28). Hadoop ecosystem- Get to know the Hadoop tools for crunching Big Data. edureka. Retrieved from https://medium.com/edureka/hadoop-ecosystem-2a5fb6740177

Sinha,S. (2014, October 9). Hadoop tutorial- A comprehensive guide to Hadoop. edureka. Retrieved from https://medium.com/edureka/hadoop-tutorial-24c48fbf62f6

9. Appendix – Hadoop and Apache Kylin

Apache Kylin is an open-source distributed analytics engine designed to provide a SQL interface and multi-dimensional analysis on Hadoop supporting extremely large datasets. It was originally developed by eBay and is now a project of the Apache.

In addition, it easily integrates with BI tools via ODBC driver, JDBC driver, and REST API. It was created by eBay in 2014, graduated to Top Level Project of Apache Software Foundation just one year later, in 2015 and won the Best Open-Source Big Data Tool in 2015 as well as in 2016.

Currently, it is being used by thousands of companies worldwide as their critical analytics application for Big Data. While other OLAP engines struggle with the data volume, Kylin enables query responses in the milliseconds. It provides sub-second level query latency over datasets scaling to petabytes. It gets its amazing speed by precomputing the various dimensional combinations and the measure aggregates via Hive queries and populating HBase with the results.

Apache Kylin™ lets you query billions of rows at sub-second latency in 3 steps.

1. Identify a Star/Snowflake Schema on Hadoop.
2. Build Cube from the identified tables.
3. Query using ANSI-SQL and get results in sub-second, via ODBC, JDBC or RESTful API.

9.1 Big Data Tool with Apache Kylin

Now, before moving on to Apache Kylin, let us start the discussion with Big Data, that led to the development of Hadoop and then to Apache Kylin.

IoT connects our physical device to the internet and makes it smarter. Nowadays, we have smart air conditioners, televisions etc. Our smart air conditioner constantly monitors our room temperature along with the outside temperature and accordingly decides what should be the temperature of the room. Now imagine how much data would be generated in a year by smart air conditioner installed in tens & thousands of houses. By this we can understand how IoT is contributing a major share to Big Data.

❖ **Why Big Data is a problem statement and how Hadoop solves it.**

There were three major challenges with Big Data:

1. **The first problem is storing the huge amount of data**. Storing huge data in a traditional system is not possible. The reason is obvious, the storage will be limited to one system and the data is increasing at a tremendous rate.
2. **The second problem is storing heterogeneous data**. The data is not only huge, but it is also present in various formats i.e. unstructured, semi-structured and structured. So, we need to make sure that we have a system to store different types of data that is generated from various sources.
3. **Finally, the third problem, which is the processing speed**. Now the time taken to process this huge amount of data is quite high as the data to be processed is too large.

To solve the storage issue and processing issue:

- Two core components were created in Hadoop — **HDFS(**Hadoop Distributed File System**) and YARN(**Yet Another Resource Negotiator**)**. HDFS solves the storage issue as it stores the data in a distributed fashion and is easily scalable. And YARN solves the processing issue by reducing the processing time drastically.

While setting up a Hadoop cluster, we have an option of choosing a lot of services as part of your Hadoop platform, but there are two services which are always mandatory for setting up Hadoop. One is HDFS (storage) and the other is YARN (processing). HDFS stands for Hadoop Distributed File System, which is a scalable storage unit of Hadoop whereas YARN is used to process the data i.e. stored in the HDFS in **a distributed and parallel fashion.**

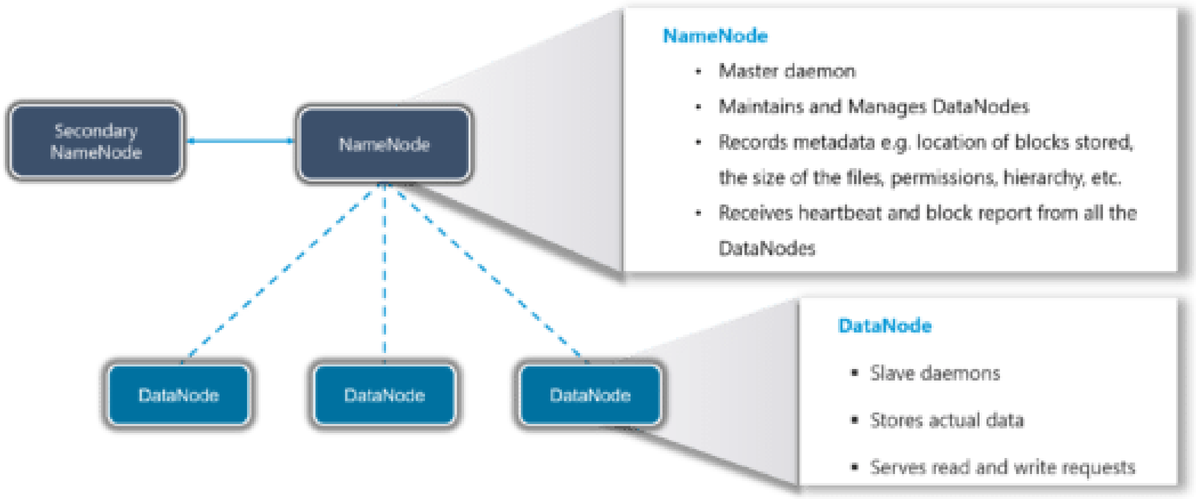

Secondary NameNode
NameNode
DataNode
DataNode
DataNode
NameNode
• Master daemon
• Maintains and Manages DataNodes
• Records metadata e.g. location of blocks stored, the size of the files, permissions, hierarchy, etc.
• Receives heartbeat and block report from all the DataNodes
DataNode
• Slave daemons
• Stores actual data
• Serves read and write requests

Figure 9. HDFS

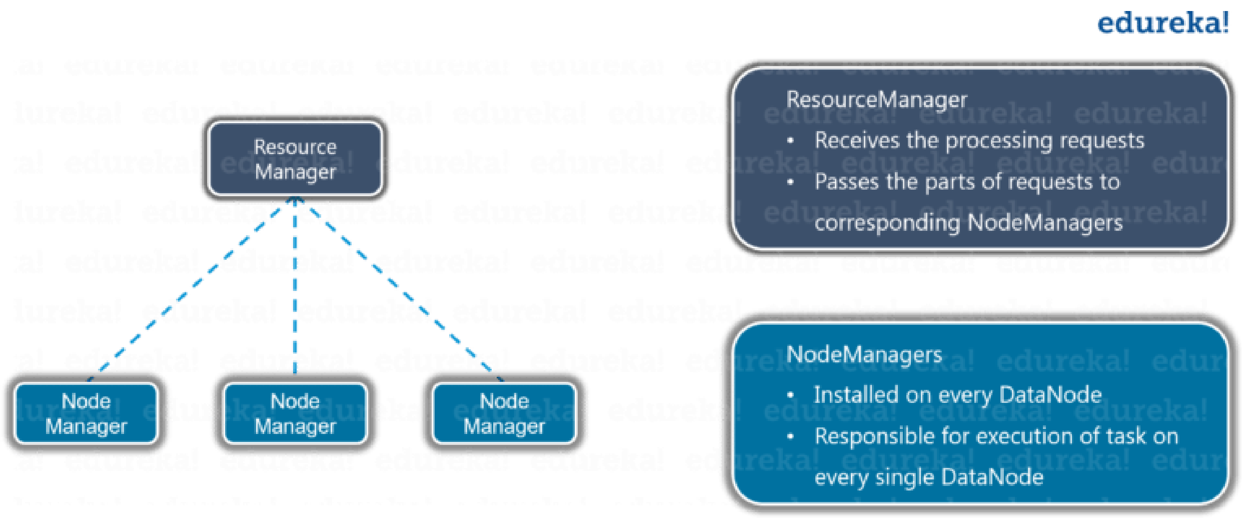

edureka!
Resource Manager
Node Manager
Node Manager
Node Manager
ResourceManager
• Receives the processing requests
• Passes the parts of requests to corresponding NodeManagers
NodeManagers
• Installed on every DataNode
• Responsible for execution of task on every single DataNode

Figure 10. YARN

Data Warehouses have served very good purposes of Data Storage and Data Analytics. But with the exponential growth of data and with the potent intelligence lying in them, enterprises are pushing toward adopting Big Data Technologies.

While Hadoop provides a reliable, scalable and distributed computing power, its initial design was towards enabling batch processing of large data. But if you need to move our analysts to use Hadoop, to unleash the power of insights into large data, we need to enable:

1. Interactive querying capability

2. Reporting and dashboard development through BI Tools

9.2 Hadoop Eco-system:

The ecosystem around Hadoop is rapidly developing and adding on tools for various data needs. No single tool can provide both and hence we would have to go with a combination of tools.
The solution could be a combination of one Interactive Querying tool and one OLAP (online Analytical Processing) tool on Hadoop. **OLAP on Hadoop – Apache Kylin**

- ❖ **What is Kylin?**

Kylin is an open source Distributed Analytical Engine that provides SQL interface and multidimensional analysis (OLAP) on Hadoop supporting extremely large datasets. Apache kylin pre-calculates OLAP cubes and store the cubes into a reliable and scalable datastore (HBase).

- ❖ **Why Kylin?**

In most of the use cases in bigdata, we see the challenge is to get the result of the query within a sec. It takes lot of time to scan the database and to return the results. This is where the concept of OLAP in Hadoop emerged to combine the strength of OLAP and Hadoop and hence gives a significant improvement in query latency.

- ❖ **How it works?**

Below are the steps on how kylin fetches the data and saves the results.

- First, sync the input source table. In most of the cases, it reads data from Hive
- Next it runs map reduce / spark jobs (based on the engine you select) to pre-calculate and generate each level of cuboids with all possible combinations of dimensions and calculate all the metrics at different levels.
- Finally, it stores cube data(aggregated result) in HBase where the dimensions are rowkeys and measures are column family.

After aggregated data is ready, we can integrate with popular BI tools, such Tableau or Superset or connect with JDBC from our code. One more option is Kylin also provides the simple UI for quickly visit data.

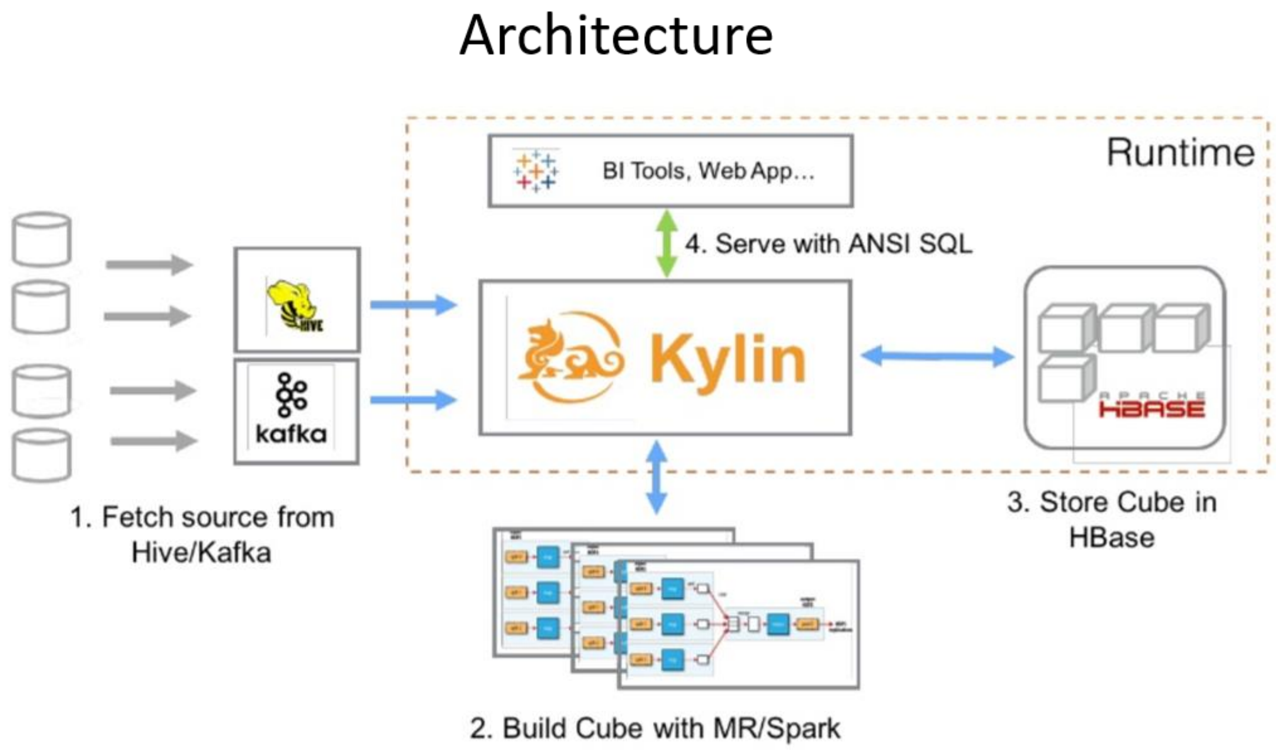


Figure 11. Apache Kylin – Architecture

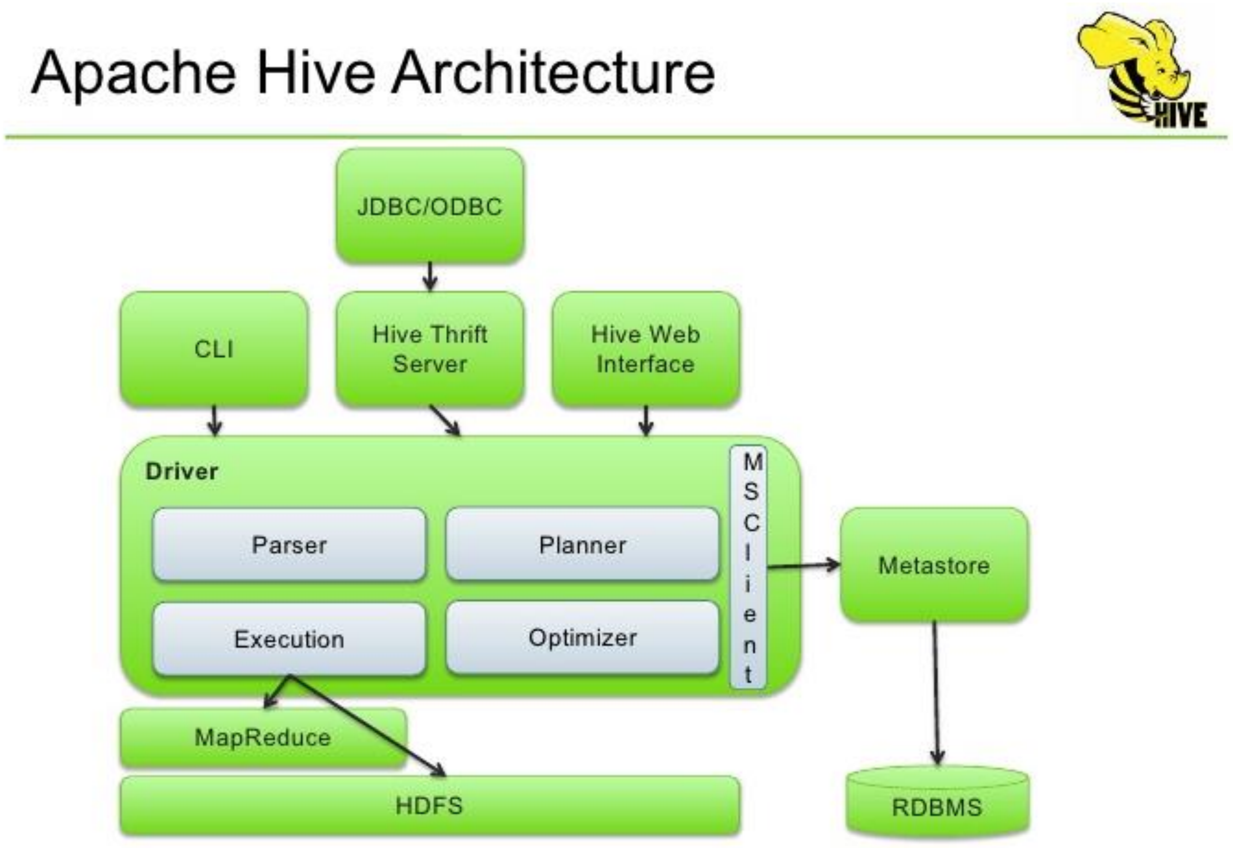
9.2.1 Hive

Apache Hive is a data warehouse system built on top of Hadoop or in Hadoop Ecosystem and is used for analyzing structured and semi-structured data. Basically, it provides a mechanism to project structure onto the data and perform queries written in HQL (Hive Query Language) that are similar to SQL statements. Internally, these queries or HQL gets converted to map reduce jobs by the Hive compiler.

Hive is not only a savior for people from a non-programming background, but it also reduces the work of programmers who spend long hours writing MapReduce programs.

Apache Hive supports Data Definition Language (DDL) and Data Manipulation Language (DML)

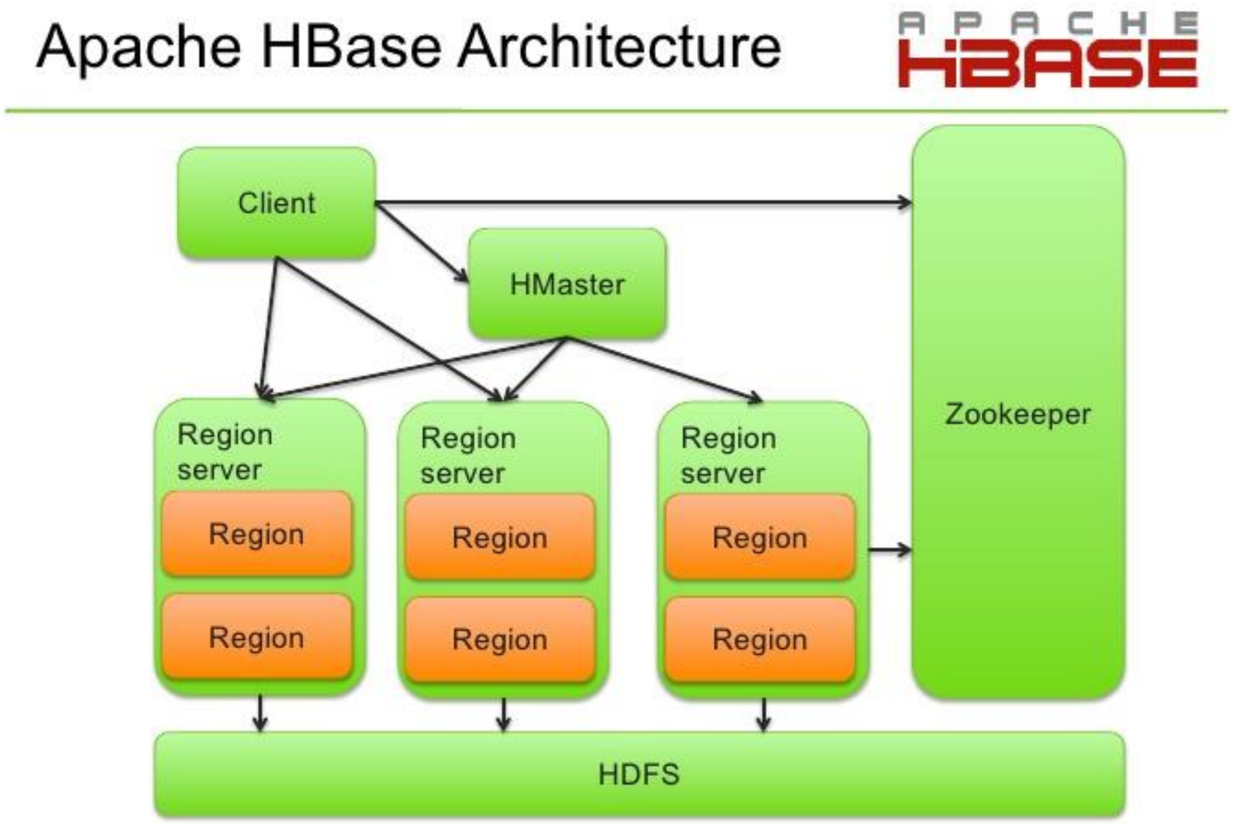
SQL + Hadoop MapReduce = HiveQL



9.2.2 HBase

HBase is an open-source, multidimensional, distributed, scalable, and a NoSQL database written in Java. HBase runs on top of HDFS (Hadoop Distributed File System) and provides Bigtable like capabilities to Hadoop.

HBase supports random read and writes while HDFS supports WORM (Write once Read Many or Multiple times).



Where we can use HBase?

We should use HBase where we have large data sets (millions or billions of rows and columns) and we require fast, random, and real-time, read, and write access over the data.

OLAP cube definition

First the user must identify the tables and the relationship between them over the data model stored in the source system chosen. Secondly, to reduce the effects of the dimensionality curse, it is very important to determine the optimal type for each dimension column. Finally, for those dimensions defined as “Normal”, we must create one or more aggregation groups and apply to them the possible optimizations allowed.