# Reach-U Assignment Design

In this Project java 1.8 and spring framework are used. According to the main task, a general solution has been developed for writing multiple parquet files into file system.

## Prerequisites

1. Install Maven
2. Install Java JDK or JRE with 1.8 version
3. Install RabbitMQ (for the quick install, use docker-compose.yml in Project)
4. Define environment variable as “REACHU\_PROJECT\_BASE”. This variable is used to locate Project related configs and outputs

For a basic explanation; Maven is needed to build codes and generates executable. Java 1.8 version is needed to compile and run code properly ,and RabbitMQ message queue is used to provied high tolerance queue for multiple parquet file writing into system at the same time.

## Design

* DTO (Schema)

Data transfer objects (DTO) are used to create some sample parquet structures. Using the Java Reflection API, a DTO object converted into parquet structure on runtime. If DTO object is changed, no need to define schema etc. The new DTO content immediately effect written parquet files.

There are two predefined DTO objects created;

1. SampleParquetDTO1
2. SampleParquetDTO2

Both classes are child of BaseParquetDTO. This class has a timestamp field. Timestamp is the common field of all parquet file structures.

* Partitioning

In config there are two parameters about partitioning.

1. partition.path : This indicates where the partitioning happen in file system.
2. partition.time: This parameter indicates that each day splitted into hours value. For example; value 6 means; the day has four time periods. Etc 00:00 | 06:00 | 12:00 | 18:00. Each parquet file will be located in related time range when file creation.

I implemented the root level of portioning as day folder “2018-12-20”. Each day folder has portioning folders. For example if you specify “partition.time” to 6 then you have four folders as “201812200000” | “201812200600” | “201812201200” | “201812201800” in day folder. Then each parquet file located in this folders according to the timestamp value. For example if a parquet file created at time 02:24, means this parquet file will be located in “201812200000” folder.

* RabbitMQ

RabbitMQ is most popular and trusted open source message broker. Basic features are lightweight, easy to use and deploy on cloud. Also RabbitMQ can be deployed in distributed and federated configurations to meet high-scale, high-availability requirements.it is thread safe and support concurrent messaging.

In properties there are some parametes;

1. spring.rabbitmq.host: host address of rabbitmq
2. spring.rabbitmq.port: port number of rabbitmq
3. spring.rabbitmq.username: username of rabbitmq
4. spring.rabbitmq.password: password of rabbitmq
5. spring.rabbitmq.listener.simple.concurrency: Min concurrency of message processing
6. spring.rabbitmq.listener.simple.max-concurrency: Max concurrenct of message processing
7. spring.rabbitmq.listener.simple.retry.initial-interval: Retry interval
8. rabbitmq.queue.maxlengthinbytes: Max length in bytes of queue size. It is set default to 1GB

I tested the queue using postman runner with 50K request iteration and seems it is very fast and low memory used☺

RabbitMQ works with Publish/Subscriber pattern. Therefore there are three fundamental class called; “RabbitConfig”, “ParquetMessageListener”, “ParquetMessageSender” implemented. RabbitConfig is used when the application started, it provides to build queue on rabbitmq. Message listener listens the queue, if a new message arrived then it process the message and remove the message from queue when its job finished. On the contrary, Message sender, publish new messages to the queue.

* Writer

In this project, for reading and writing parquet file I used the “parquet-avro” module is used. This module has some classes for easy read and write parquet files.

1. AvroParquetWriter: Provides to Write parquet file into file system
2. AvroParquetReader: Provides read a parquet file from file system

In addition, Abstraction pattern is used and “AbstractPartitionWriter” abstract class is created. Using this class, concrete subchilds “SampleParquetDTO1Writer”, “SampleParquetDTO2Writer” classes are also implemented.

* Controller

A controller “Controller.java” class implemented. This class have two methods.

1. Read Parquet: read a parquet file from file system with given timestamp
2. Add to Queue: create a new message with newly creted SampleParquet1 and sent to queue

## Conclusion

i believe that the result of Project can be useful for writing multiple parquet files in paralel. The Project could be much more improved, and it provides to be extensible. A new DTO can be defined in a short time in a same manner, a new writer also can be implemented in a very short time of implementation. I have tested my code using postman runner with 100K request. İt work cool and queue is the all magic ☺. For the measurement of code optimization, clearly i’m honest person, i have never made a microbenchmark process on java application. But recenly i have searched and some opinions about how to make microbenchmarking in java. After sending the Project to you, i will make it and later i will send the microbenchmark results ☺.

Thats it…