UNIVERSITAT POLITÈCNICA DE CATALUNYA

BDM PROJECT 2 REPORT

Formatted and Exploitation Zone

Big Data Management

Broniewski, Adam
Kylynnyk, Vlada

Supervisor

Nadal, Sergi

Universitat Politècnica de Catalunya
June 2022

Table of Contents

A Project Overview

The scope of this project is focused on storing and distributing tenant data for analysis. The business idea is for a tenant with a data analysis task to provide a dataset and select some basic options like which analysis type they want, and which attributes should be used. The analysis would then be complete on their behalf using pre-made machine learning code flexible for any data analysis task.

The scope this project with respect to the proof of concept is:

- Initialize a distributed cluster based on dynamic input
- Partition a dataset based on dynamic input
- Complete generic data analytics with a dynamic dataset and model inputs
- Store machine learning model, and display analytics and predicted model accuracy

This report will provide an overview and justification of the architecture used to achieve the purpose. All supporting code and resources can be found in the github repository[1] and in the appendix of this report.

B Integration and Reconciliation

B.1 Data Formatters

B.1.1 Scheduling

Data formatting begins by reading a schedule file that includes the IP addresses of cluster slaves (Hosts) and the workload ratio that each Host is intended to process. The correct dataset is retrieved based on the item “status” in the schedule. This validates the schedule file against the data structure and storage format in the landing zone.

B.1.2 Partitioning

The dataset is partitioned by reading the dataset file as an RDD using PySpark and retrieving the count of rows that exist. Several map/reduce functions are then used to create an index and map each row of the dataset to a particular key determined by the workload ratio in the schedule. As an example, let’s say a dataset of 100 rows will be processed by 3 Hosts, where the hosts have unique ID 1, 2, and 3. 1 will process 20% of the dataset, 2 will process 30%, and 3 will process 50%. The map/reduce algorithm will map 20 rows to 1, and 50 rows to 2, and 50 rows to 3. The functions called are shown below.

```
dataset_rdd = spark.read.parquet(file_location).rdd
dataset_rdd2 = dataset_rdd.zipWithIndex()
dataset_rdd3 = dataset_rdd2.map(lambda x: (x[1], x[0]))
dataset_rdd4 = dataset_rdd3.map(
        lambda x: ((1 if x[0] < second_lower else
                    (2 if x[0] < third_lower else 3)), x[1]))
```

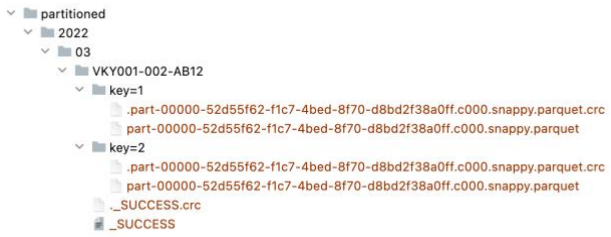
[1] Github repository: abroniewski/IdleCompute-Data-Management-Architecture

The partitioned dataset would then be written to the formatted zone with a partition by key.

B.2 Formatted Zone

The partitioned data set would then be written to the HDFS cluster by writing to the cluster master. The partitioning will control the amount of data that would be processed by each slave, as Hadoop takes advantage of data locality. Any scripts for data exploitation must also be run from the cluster master.

In the previous example, the resulting partitioned dataset is stored with the partitioning column values encoded in the path of each partition directory as seen below.



```
partitioned
  2022
    03
      VKY001-002-AB12
        key=1
          .part-00000-52d55f62-f1c7-4bed-8f70-d8bd2f38a0ff.c000.snappy.parquet.crc
          part-00000-52d55f62-f1c7-4bed-8f70-d8bd2f38a0ff.c000.snappy.parquet
        key=2
          .part-00000-52d55f62-f1c7-4bed-8f70-d8bd2f38a0ff.c000.snappy.parquet.crc
          part-00000-52d55f62-f1c7-4bed-8f70-d8bd2f38a0ff.c000.snappy.parquet
        ._SUCCESS.crc
        _SUCCESS
```

Partitioned data would be pushed to the master node in an HDFS cluster created with temporary slave nodes based on the schedule provided.

C Exploitation Zone

The exploitation zone made use of the following tools and technologies:

- PySpark – Python interface for Apache Spark
- MLlib – Spark machine learning
- HandySpark – column statistics and visualizations

C.1 Data Integration

Due to the unknown nature of incoming user data, the transformations must be executed in a manner that is generic and agnostic to the dataset structure.
When a dataset is originally uploaded for analysis, it includes 2 additional metadata files:

- Header names (features and target)
- Analytics model parameters

These files are combined with the dataset from the formatted zone to transform the data into a target schema that is different from the data model and more suitable for machine learning algorithms. The data transformations are done using PySparks DataFrames, which maintain a distributed collection of data and provide named columns for abstraction. This is the correct tool for exploratory analysis as the DataFrame API is faster for analysis that requires aggregations by avoiding the need for Java serialization as the schema is known. Additionally, DataFrames can directly access the MLlib API.

The transformations shown in the table below were completed.

Transformation	Function used
Data description by feature (count, mean standard deviation, min, max)	`df.describe().toPandas()`

Histogram generation	`hdf.cols[columnName].hist()`

Histograms were generated with the help of the HandySpark library[2], which provides access to matplot lib descriptive analytics and visualization functionality while still taking advantage of performing distributed computing.

All saved data from the exploitation zone is written to a folder following the same structure used by the partitioned and processed datasets in a parent directory named “analyzed”.

```
IdleCompute-Date-Store
    └─ data
        ├─ processed
        │   └─ 2022
        │       ├─ 03
        │       │   └─ ABR001-002-AB12.parquet
        │       └─ …
        ├─ partitioned
        │   └─ 2022
        │       └─ 03
        │           └─ ABR001-002-AB12
        │               ├─ key=1
        │               └─ key=2
        └─ analyzed
            └─ 2022
                └─ 03
                    └─ ABR001-002-AB12
                        ├─ describe.csv
                        ├─ feature1.pickle
                        ├─ feature1_histogram.png
                        └─ …
```

Figure: Formatted and Exploitation zone data structure

The data description is written as a csv called “describe.csv”. The histogram plots are all exported as “.png” files and as “.pickle” files.

A file containing user selected parameters to be used with a selected model will also be integrated to train the model. Model parameter selection is, again, made to be generic regardless of dataset. The file with model parameters is generated during user dataset upload and analytics request.

C.2 Data Quality Process

There is no data quality checks in this stage of the proof of concept development, however controls are in place for some aspects of data creation. The model parameter file is system generated; all inputs are controlled by IdleCompute. The data pipeline would benefit most from a consistency check of the number of columns in the analysis dataset against the number of features provided in the feature names file.

[2] HandySpark repository - https://github.com/dvgodoy/handyspark

D Data Analysis

D.1 Analytical Sandboxes

Generic analytics do not include any subsets of data as incoming datasets will vary. KPIs are calculated by aggregating data using describe() function and saving results to share with user and display in visualization tool. No subsets of data are need for the proof of concept.

It would be possible to include this feature with user provided inputs in a separate metadata file. The sandboxes can be generated using a map/reduce with RDDs, or the stratify function from Handy Spark library, applied to Spark DataFrames.

D.2 Feature Generation

A test train split is created based on the user input for model building and testing metadata. The datasets from the test/train split are not cached to disk in this implementation, as there is a single call of the datasets for use. If, however, there were different models being trained, the data subsets should be written to disk. Caching is unlikely to be an option in the long-term given the scale of future datasets; there will not be enough memory available in the slave nodes.

Transformation	Function used
Feature standardization	`standardScaler.fit(df).transform(df)`
Train/test split	`scaled_df.randomSplit([.8, .2])`

D.3 Model Training

The model is chosen by the user and stored in the dataset name as metadata. The proof of concept defines the data preparation and building of a single model (linear regression), with the idea being that all machine learning models are built in a similar generic manner and chosen through conditions on the metadata. Parameters for the model are pulled from the parameters file stored in the landing zone, as no data formatting or cleaning is needed for the parameters file.

D.4 Model Validation

Basic model validation is completed with precision, recall and f1 score being calculated from the predicted values against the labelled dataset. As all data is being returned to the client, the validation results should be stored and included in a package for the client.

D.5 Model Deployment

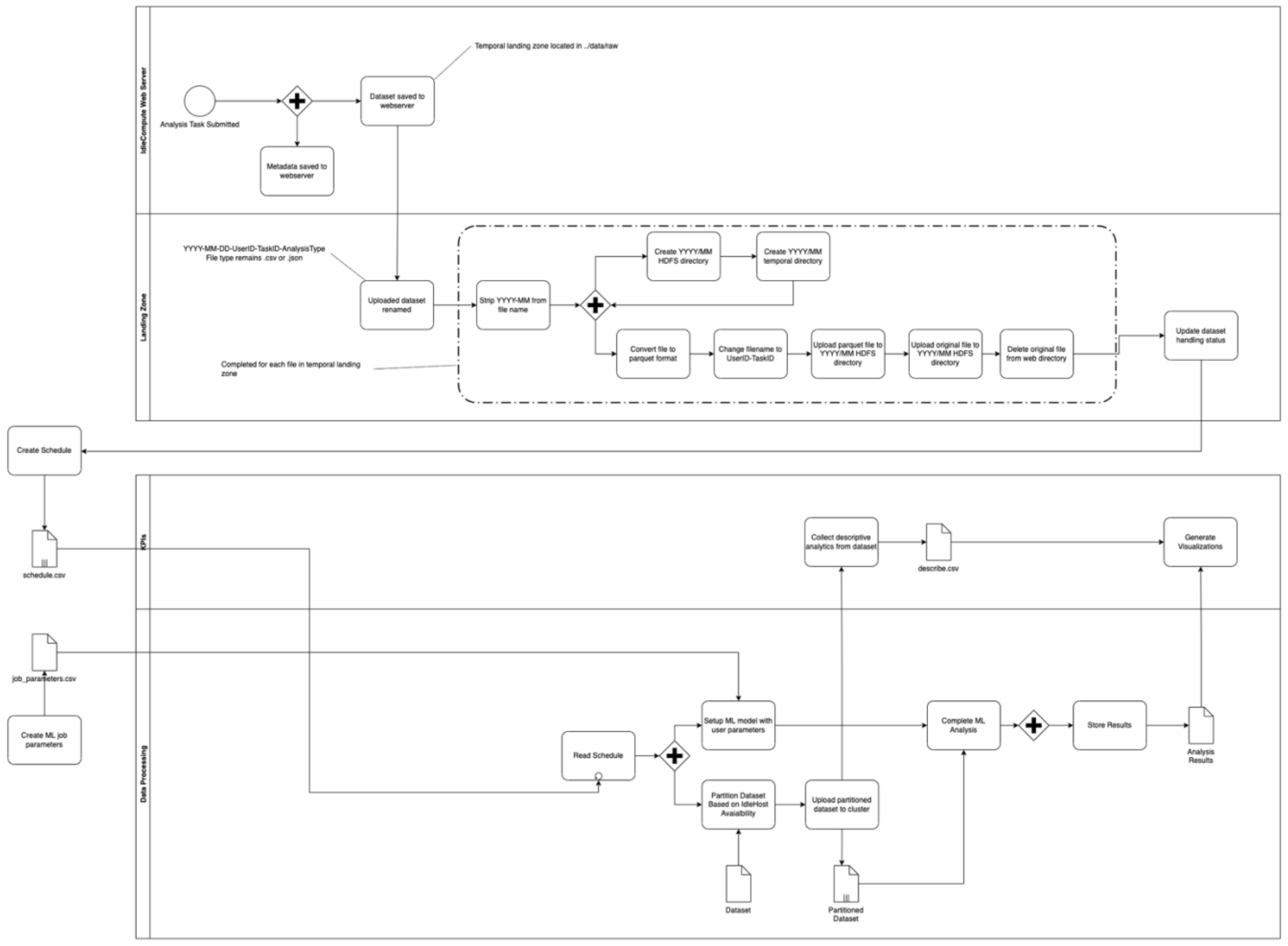
The parameters of the model trained from the data are written to the exploitation zone (analyzed directory) for deployment by the client.

E Visualization

A dashboard system like Tableau or Grafana was considered but was not used. The strength of those tools is to slice through subsets of data and give users the capability to choose alternative features to display. All the aggregations of data generated would be static in nature, and without knowing the schema of a dataset ahead of time, it is not possible to create relevant subsets for interactive exploration.

As mentioned in the exploitation section, visualizations are exported as pickle files. These are serialized plots that can be loaded with other scripts and provide users with access to plot properties like the axis and title. This allows for flexibility for the user to use the analytical visualizations as they see fit. Providing these file formats is most in line with the business needs of the user.

A Landing Zone BPMN Diagram


IdleCompute Web Server
Analysis Task Submitted
Dataset saved to webserver
Temporal landing zone located in ../data/raw
Metadata saved to webserver
Landing Zone
YYYY-MM-DD-UserID-TaskID-AnalysisType
File type remains .csv or .json
Uploaded dataset renamed
Strip YYYY-MM from file name
Create YYYY/MM HDFS directory
Create YYYY/MM temporal directory
Convert file to parquet format
Change filename to UserID-TaskID
Upload parquet file to YYYY/MM HDFS directory
Upload original file to YYYY/MM HDFS directory
Delete original file from web directory
Completed for each file in temporal landing zone
Update dataset handling status
Create Schedule
schedule.csv
KPIs
Collect descriptive analytics from dataset
describe.csv
Generate Visualizations
job_parameters.csv
Create ML job parameters
Data Processing
Read Schedule
Setup ML model with user parameters
Partition Dataset Based on IdleHost Avaialbility
Upload partitioned dataset to cluster
Dataset
Partitioned Dataset
Complete ML Analysis
Store Results
Analysis Results