INDIAN INSTITUTE OF TECHNOLOGY, ROORKEE

Department of Computer Science

(2020-2021)

CSN – 505 | PROJECT LAB REPORT

POPULARITY PREDICTION ALGORITHM FOR APTSTORE

Submitted To

Dr. Durga Toshniwal,

Professor,

IIT Roorkee

Submitted By

Dayanand Raut,

M. Tech. CSAE,

Enrollment No.: 20535010

TABLE OF CONTENTS

1. PROBLEM STATEMENT

In AptStore[1], a decision engine (DE) uses an algorithm to predict the expected popularity of files for the next reference time (RT) in the Hadoop cluster. This algorithm is called a popularity prediction algorithm (PPA). Based on the predicted popularity score of the files, DE can suggest USS (Unified Storage System) on which files are to be kept in the primary storage and which ones to be kept in the secondary storage. It also gives insight to decide which files are to be replicated more and for which files replication is to be avoided. This decision helps in improving the i/o throughput as well as reducing the storage cost. ***Implement PPA which will predict the expected popularity of files using their initial popularity, access intervals, number of blocks and load in the cluster.***

2. INTRODUCTION

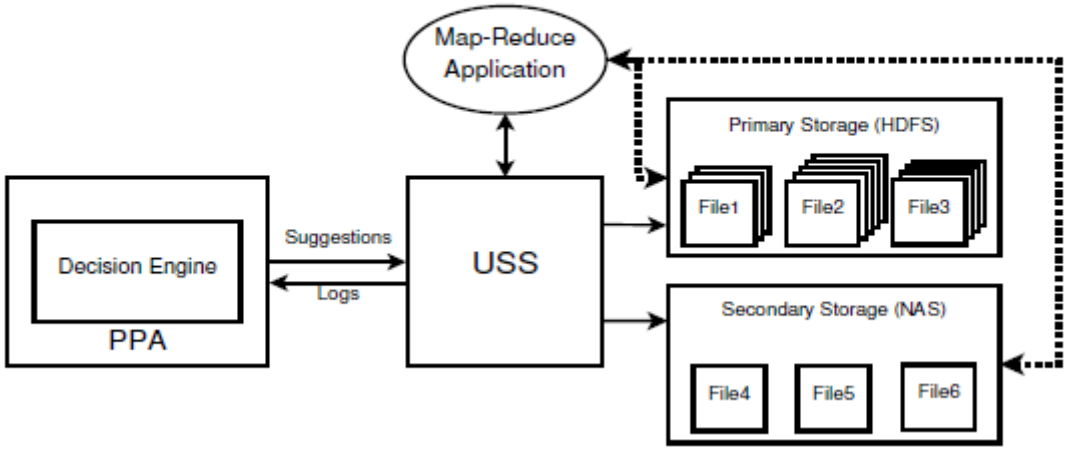


Fig. Architecture of Aptstore

Typical Hadoop setups employ Direct Attached Storage (DAS) with compute nodes and uniform replication of data to sustain high I/O throughput and fault tolerance. However, not all data is accessed at the same time or rate. Thus, if a **large replication factor** is used to support higher throughput for popular data, it **wastes storage** by unnecessarily replicating unpopular data as well. Conversely, if **less replication** is used to conserve storage for the unpopular data, it means fewer replicas for even popular data and thus **lower I/O throughput**.

AptStore[1]:

AptStore is a dynamic data management system for Hadoop, which aims to improve overall I/O throughput while reducing storage cost. It is a tiered storage system that uses the standard Direct Attached Storage (DAS) (**primary storage**) for **popular data** to sustain high I/O throughput, and network attached storage (NAS) (***secondary storage***) for cost-effective, fault-tolerant, but lower-throughput storage for ***unpopular data***. Its architecture is shown in the figure above.

To determine how popular a file is, an algorithm is needed that can analyze the file system audit logs and predict the popularity of file. This is done by using a file **popularity prediction algorithm (PPA).** At every RT (Reference time), the PPA analyzes the access pattern for each file and predicts a expected popularity value for it for the next RT.

3. ALGORITHM

Input : USS file System Audit Logs
Output: Predicted popularity $P_{predicted}$.
F is the set of files in the file system;
foreach *access $i+1$ to the file $f \in F$ in RT* **do**
  if i==*0* **then**
    $P_{i+1}(f) \leftarrow AVG(P)$;
  end
  else
    $P_{i+1}(f) \leftarrow P_i(f) + \frac{c}{a(f)*l*b(f)*P_i(f)}$;
  end
  if $P_i(f) < P_{Min}P$ **then** $P_i(f) \leftarrow P_{Min}$;
  else if $P_i(f) > P_{Max}$ **then** $P_i(f) \leftarrow P_{Max}$;
  $IP = IP + P_{i+1}(f) - P_i(f)$;
end
foreach *deletion of the file f in F* **do**
  $IP = IP + AVG(P) - P(f)$;
end
$MIP \leftarrow \frac{IP}{size(F)}$;
foreach *file f in F* **do**
  $P_i(f) \leftarrow P_i(f) - \frac{MIP}{s}$;
  where i is the most recent access to the file f.
  $P_{predicted}(f) \leftarrow P_i(f) + (P(f) - P_i(f))$;
  $P(f) \leftarrow P_i(f)$;
end

Algorithm 1: Popularity Prediction Algorithm.

The popularity value $P_{i+1}(f)$ of a file f varies with each access $i + 1$ to the file. $P_{i+1}(f)$ is defined as:

$$P_{i+1}(f) = P_i(f) + \frac{c}{a(f) * l * b(f) * P_i(f)}$$

In the above formula, c is the popularity constant, a(f) is a function of the access interval of file f, l is the load in the cluster and b(f) is a function of number of blocks in the file f.

During the creation of a file, the popularity of the file $P_1(f)$ is initialized to average file popularity observed in the system, *AVG(P)*. Whenever a file is deleted, it will result in popularity of other files being modified when the values are updated at the end of RT. When a popular file is deleted, the popularity of other files in the system increases. Conversely, when an unpopular file is deleted, the popularity of other files decreases. We do fix the minimum, P_{Min}, and maximum, P_{Max}, threshold on the popularity of a file to make sure that there are bounds on the number of file replicas in the system. The minimum threshold ensures data reliability and compliance with system SLAs, while maximum threshold captures space constraints in primary storage.

After the accesses of all files in the reference time RT are processed the popularity value $P_i(f)$ of a file *f* for the most recent access *i* is modified as follows:

$$P_i(f) = P_i(f) - \frac{MIP}{s}$$

MIP is the mean increase in the popularity of the file f during reference time RT, *AVG(P)* is the average popularity of all the files in the cluster, *s* is the scalability constant. The mean increase in popularity is a fraction of increase in popularity, *IP* during RT over F, the set of all files in the system. The scalability constant s is used to contract or expand the amount of data stored in primary storage. For a value of *s* greater than one, more data is pushed to primary, while a positive value of *s*, less than one, creates more space in the primary storage.

4. IMPLEMENTATION DETAILS

The PPA algorithm is implemented in Python (python3.8) using Dynamic Programming (Bottom-Up Approach)

```
def predict_popularity(accessed_files, deleted_files, popularity,block_numbers, l, c, s, Pmin, Pmax):
```

- accessed_files : list of accessed files
- deleted_files : list of deleted files
- block_numbers : dictionary containing the number of blocks in files

- popularity: dictionary containing the initial popularity of file
- l: load in the cluster
- c: popularity constant
- s: scalability constant
- P {}: dictionary of list to store the expected popularity of each access of each file
- P[i][f] = expected popularity of file *f* at i^{th} access
- i = len(P[f])-1
- For i = 1, P[f][i] = average popularity in the cluster
- For i>1, P[f].append(**P[f][i] + c / access_interval(f) * l * block_number(f) * P[f][i]**)
- access_interval(f): the interval between current and last access of *f*
- block_numbers(f): number of blocks in file *f*
- Pred: dictionary to store expected predicted popularity of each file

5. DATA SIMULATION

As the actual USS audit logs were not found, input data has been simulated by randomly generating the values for accessed_files, deleted_files, popularity, block_numbers.

- accessed_files[] : list of randomly generated files from 1 to 9 for a random number 14 to 18.
- deleted_files[] : list of randomly chosen files from the accessed_files
- block_numbers{}: a dictionary that maps random number (1,50) to each unique file of accessed_files
- popularity{}: a dictionary that maps random number in the range of Pmin =0.1 to Pmax=0.9) each unique file of accessed_files
- c = 10000
- l = 9
- s = 5

6. RESULT

```
Accessing files:  [8, 3, 8, 5, 6, 8, 5, 4, 9, 7, 5, 4, 4, 1, 8, 7, 2]
Deleting files:  [2, 6]
```

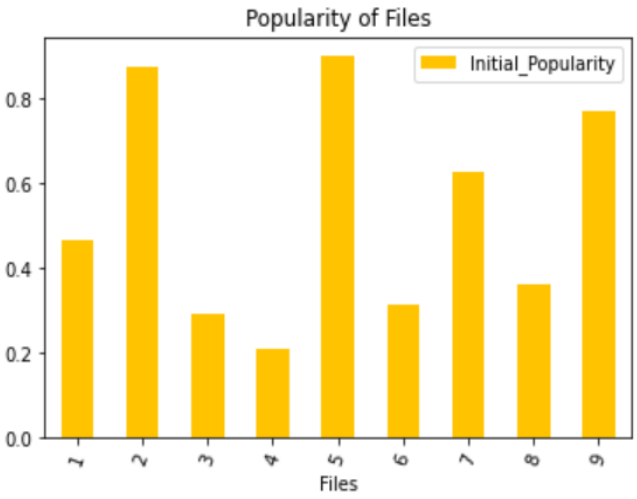
The random files accessed and files deleted has been shown above. Thereafter, initial popularity, number of block numbers have been generated randomly for each unique files in the accessed_files. Frequency is the number of times the file has been accessed.

Files	Initial_Popularity	Block_numbers	Frequency
1	0.465	47	1
2	0.874	45	1
3	0.294	22	1
4	0.210	4	3
5	0.900	30	3
6	0.314	49	1
7	0.627	10	2
8	0.363	13	4
9	0.773	5	1

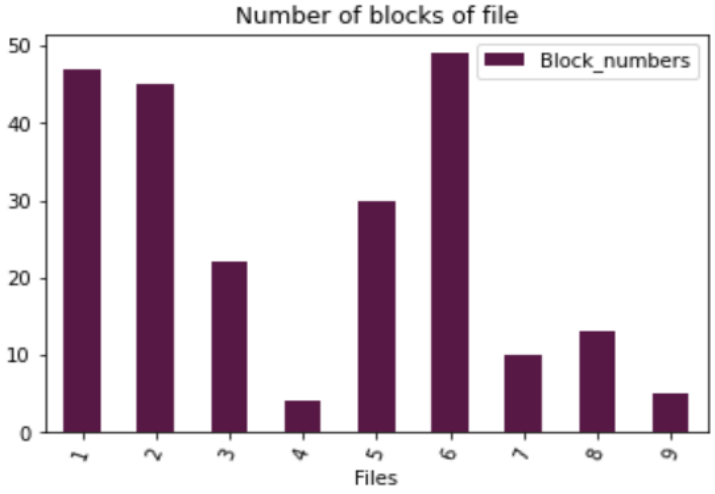
After running the algorithm, the expected popularity of files are shown below. NaN in the Predicted_Popularity column represents that file has been deleted.

Files	Initial_Popularity	Block_numbers	Frequency	Predicted_Popularity
1	0.465	47	1	-0.699146
2	0.874	45	1	NaN
3	0.294	22	1	-0.870283
4	0.210	4	3	14.066784
5	0.900	30	3	2.365740
6	0.314	49	1	NaN
7	0.627	10	2	2.919645
8	0.363	13	4	7.618028
9	0.773	5	1	-0.391488

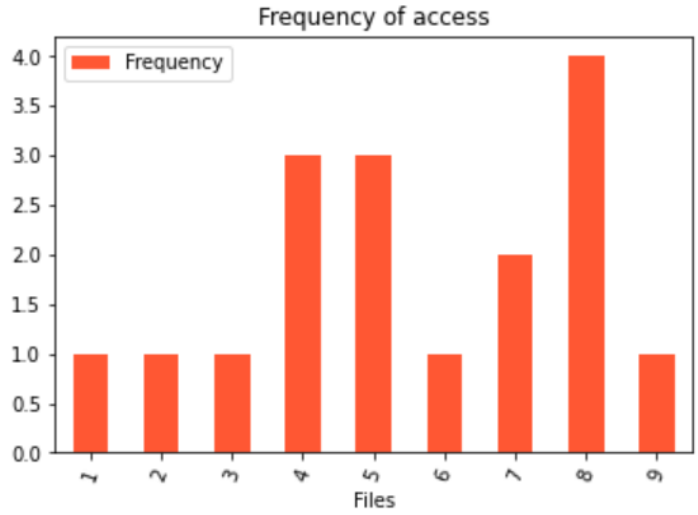
Result Analysis: File 4 has the highest value for the predicted_popularity as it has comparatively the least number of blocks and it has been accessed comparatively a greater number of times. Conversely, File 3 has the lowest value for the predicted_popularity as it has comparatively a greater number of blocks and it has been accessed only once. Also, its initial popularity is less.



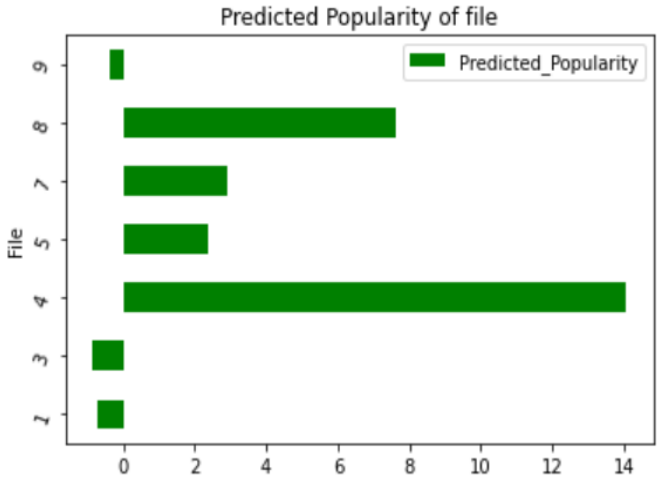
Initial Popularity of files



Number of blocks in files



Number of times files are accessed



Output: Predicted popularity of files

7. CONCLUSION

Hence, the file popularity prediction algorithm (PPA) was implemented in python which was able to predict the expected popularity of the files for next reference time based on the initial popularity, access interval, number of blocks, load in the cluster and popularity constant. The data was simulated by generating random values and fed to the algorithm to get the result. The results were explained and represented in the bar graph for better visualization.

Repository

- **Source code : https://github.com/dayanandraut/aptstore-popularity-prediction-algorithm**
- **Jupyter Notebook : https://github.com/dayanandraut/aptstore-popularity-prediction-algorithm/blob/main/aptstore_popularity_prediction_algorithm.ipynb**

References

1. ***K. R. Krish, A. Khasymski, A. R. Butt, S. Tiwari and M. Bhandarkar, "AptStore: Dynamic storage management for hadoop," 2013 IEEE International Conference on Cluster Computing (CLUSTER), 2013, pp. 1-5, doi: 10.1109/CLUSTER.2013.6702657.***
2. ***https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.plot.bar.html***