Mining on Big Data Using Hadoop MapReduce Model

IOP Conf. Series: Materials Science and Engineering **263** (2017) 042007 doi:10.1088/1757-899X/263/4/042007

Mining on Big Data Using Hadoop MapReduce Model

Salman Ahmed G and Sweta Bhattacharya

VIT University, Vellore 632014, Tamil Nadu, India
E mail: sweta.b@vit.ac.in

Abstract. Customary parallel calculations for mining nonstop item create opportunity to adjust stack of similar data among hubs. The paper aims to review this process by analyzing the critical execution downside of the common parallel recurrent item-set mining calculations. Given a larger than average dataset, data apportioning strategies inside the current arrangements endure high correspondence and mining overhead evoked by repetitive exchanges transmitted among registering hubs. We tend to address this downside by building up a learning apportioning approach referred as Hadoop abuse using the map-reduce programming model. All objectives of Hadoop are to zest up the execution of parallel recurrent item-set mining on Hadoop bunches. Fusing the comparability metric and furthermore the locality-sensitive hashing procedure, Hadoop puts to a great degree comparative exchanges into an information segment to lift neighborhood while not making AN exorbitant assortment of excess exchanges. We tend to execute Hadoop on a 34-hub Hadoop bunch, driven by a decent change of datasets made by IBM quest market-basket manufactured data generator. Trial uncovers the fact that Hadoop contributes towards lessening system and processing masses by the uprightness of dispensing with excess exchanges on Hadoop hubs. Hadoop impressively outperforms and enhances the other models considerably.

1. Introduction

Traditional parallel recurrent item-set mining techniques focused only on balancing the data among the computer clusters Because of lack of analysis of correlation it leads to poor data locality. So it increases data shuffling cost and network traffic. In this study, we focused on network traffic and partitioning of data between different clusters. [1]. Existing concept deals with providing backend by using MySQL which contains lot of drawbacks i.e. data limitation is that processing time is high when the data is huge and once data is lost we cannot recover so thus we proposing concept by using Hadoop tool. Existing technique uses MySQL db. MySQL is a relational database .in these we can store the data and processing but some limitations. The limitations are we can process limitation of data, we get results with take more time and maintenance cost is very high.

Initially, Dean and Ghemawat [3] presented a model of MapReduce. The aim is to make simple from the large cluster of data. MapReduce is a programming methodology deals with implementation and generating a large data sets. It helps to generate the key/value pairs from intermediate key/value data. Paper retain in this form are parallelized and specify on large cluster forms. Similarly, Sakr [4] proposes a model using map reduce and large scale data processing mechanism. This paper covers a set of established systems that have been implemented to provide declarative programming interfaces on top of the map reduce framework. In addition, the author's also assess overlarge scale data processing systems that resemble some of the ideas of the map reduce framework for different

purposes and application scenarios. Finally, this paper discusses some of the future research directions for implementing the next generation of Map Reduce-like solutions.

Later, Wang [5] projected data model using efficient algorithm of frequent item sets mining based on map reduce. This model contains normal parallel algorithms for mining frequent item sets (patterns). It is designed by implementing fixed partition (FP) growth or Apriori algorithms on map reduce (MR) framework. Existing MR FP-growth algorithms cannot distribute data equally among nodes, and MR apriori algorithms utilize multiple map/reduce procedures and generate too many key-value pairs with value of 1; these disadvantages hinder their performance. This paper proposes an algorithm of fixed item set mining on map reduce (FIMMR).It initially mines local frequent item sets for each data chunk as candidates, applies prune strategies to the candidates, and then recognizes global frequent item sets from candidates. Experimental results show that the time efficiency of FIMMR outperforms parallel fixed partitioning (PFP) and SPC significant; and under small minimum support threshold, FIMMR can achieve one order of magnitude improvement than the other two algorithms; meanwhile, the speedup of FIMMR is also satisfactory.

Again, Yahya [6] introduced Hadoop-map reduce model using apriori algorithm.. In this paper, MapReduce was implemented by using the new type of algorithm called apriori algorithm. This algorithm is similar to Hadoop system map reduce model. It consists of two phases. One phase to find the map of similar data and another phase is reduced to aggregate the map data. Experimental results showed that the proposed MR Apriori algorithm outperforms the other two algorithms.

Further, Lu [10] offered an efficient processing model using k-nearest neighbour join (KNN join) and map reduce. The main aim is to find the nearest neighbour from a given set of data. This algorithm was followed by many data mining approach. It was very expensive because. It involved both join operation and k-nearest neighbour. In this paper, KNN join is performed using MapReduce model. The mappers help to cluster the object into groups and perform the KNN joins in each group. This type of algorithm helps to reduce the data shuffling cost and network overhead.

Likewise, Li [11] describe about frequent item set mining (FIM) based on map reduce. Mainstream parallel algorithms for mining frequent item sets (patterns) were designed by implementing FP-growth or apriori algorithms on map reduce framework. To speed up the process of performance many large numbers of FIM algorithm was developed. When data is very large then computational cost and memory usage will be high. To avoid this, the parallel FP-growth algorithms on distributed machine were developed. One specialty of PFP partition is that the machine executes on an independent group of the mining task. In this type of partitioning, the computational dependencies between machines will be reduced and communication between them will be increased.

Recently, many researches has been carried out on large dataset using hybrid rough computing techniques like rough set and formal concept analysis [12], dominance based rough set [13][14], containing order rough set [15] in various fields such as networks, information security, agriculture, marketing, image processing etc., to deal with uncertainties for data reduction, classification and rule generation.

In this paper we analyse item set pattern data using Hadoop tool along with some Hadoop ecosystems such as map reduce, hive and pig. By using these tools we prevent limitation of data, data lost problem and obtain high throughput, less maintenance cost. These tools are supported by open source and java based. Hence they are compatible on all the platforms. The rest of the paper is organized as follows. Section 2 portray about foundation of dataset. Section 3 deals with data analysis and system architecture. The proposed model is presented in section4. Experimental analysis is carried out in Section 5 followed by conclusion and future enhancement in Section 6.

2. Foundation of Dataset

Let a dataset $D=\{T_1,T_2,\cdots,T_m\}$ consists of m transactions and n dissimilar items $IT=\{I_1,I_2,\cdots,I_n\}$. Every transaction $T=\{SID,PR\}$ in D contains item set PR and a sole identifier SID and $PR\subseteq IT$.

$|D|$ Stand for the size of data set D. An item set $X = \{I_{j_1}, I_{j_2}, \cdots, I_{j_r}\}$ where $\{1 \le j_1 \le j_2 \le \cdots \le j_r \le n\}$ holding r different items are termed r - item set and r is the length of item set X .The least support threshold($LstSup$) is a client specific percent amount of transactions in dataset D; then least support number,($LstSupNum$), in D is labelled by $LstSupNum = LstSup \times |D|$ .The support number ($SupNum$) of item set X is the amount of transactions holding X in a dataset. An item set X is a recurrent item set if its support number ($SupNum$) is greater than or equal to least support number ($LstSupNum$).

Let a dataset D be partitioned into L small datasets $D_1, D_2, \cdots, D_L$. $LstSup \times |D_i|$ is said to be limited least support number of D_i $(1 \le i \le L)$. In dataset D_i, item set X is a limited recurrent item set if its limited least number is greater than or equal to limited least support number ($LimLstSupNum$).Any subset X ($X \ne Null$) of a recurrent item set is a recurrent item set; any superset of a non-recurrent item set is not a recurrent item set [16].

3. Data Analysis and System Architecture

It consists of five different modules as illustrated in Figure 1 namely data preprocessing, data migration with sqoop, data analytics with hive, data analytics with pig and data analytics with map reduce. Data preprocessing module is used to create data set for making item-set of product. This item set contain set of table such that set of item detail, set of transaction detail, set of user detail. With the help of these dataset we analyze this paper.

The data migration with sqoop module is used to transfer the dataset into Hadoop. Sqoop is a tool for transferring data between databases and Hadoop. With the help of this module we fetch the dataset into Hadoop using sqoop tool. Using sqoop we perform lot of the function, such as if we want to fetch the particular column or if we want to fetch the dataset with specific condition that will be support by sqoop tool and data will be stored in Hadoop.

Data analytic with hive module is used to analyze structure language. Hive is a data ware house system for Hadoop. It runs structured query language (SQL) like queries called hive query language (HQL). The HQL is converted internally to map reduce jobs. Hive was introduced by Facebook. Hive supports functions like data definition language (DDL), Data manipulation language (DML) and user defined functions. In this module we have to analysis the dataset using hive tool which will be stored in Hadoop. To analyze dataset we use hive with HQL. Using hive we perform Partition, Bucketing concept.

The module of data analytic with pig is also used to analyze data set. Pig handles both structure and unstructured language. The language of pig is pig latin script. Pig latin script is a data flow language. Apache pig is a high level data flow platform to execute map reduce programs with Hadoop.

Data analytic with map reduce module is also used analyze data set with map reduce. Map reduce is a processing technique using program model of java for distributed computing. The map reduce algorithm contains two important tasks such as map and reduce. The task map is used to map with chart, record, plot, drawing, plan and diagram etc. Whereas task reduce is used to minimize the dimension.

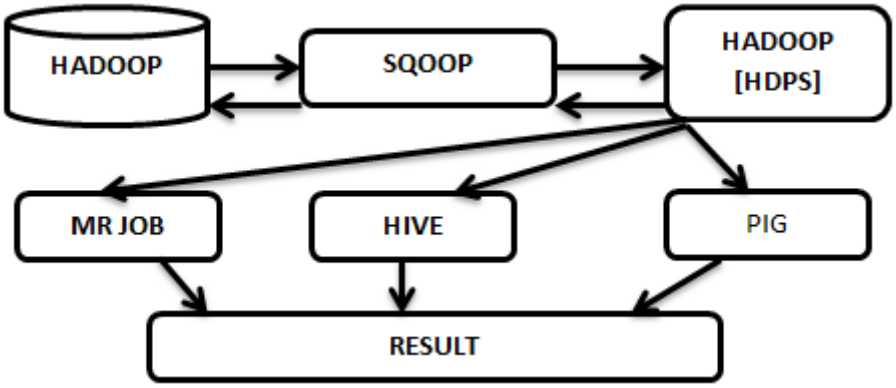

Figure 1. System architecture of proposed model

4. Proposed Model

Hadoop is an open source framework and is managed by apache software foundation. It is used for storing and processing huge datasets with a cluster of commodity hardware. We use Hadoop tool to store and process datasets. Hadoop tool contains two objects HDFS and mapreduce. We also use Hadoop ecosystems such as sqoop, hive and pig to prevent data loss problem and to perform efficient data processing.

Hadoop contains processing layer and computation layer (map reduce), storage layer (Hadoop distributed file system (HDFS)). Map reduce is an effortless and influential programming model that allows trouble-free progress of scalable comparable applications to process data sets on large clusters of goods machines. Hadoop file system is generated from distributed file system design. It is running on hardware. HDFS is very cheap and have highly resistance to failure. HDFS helps to work with large amount of data in less time.

System technique uses map reduce concept. MapReduce one of special techniques mainly built for distributed machine. This algorithm consists of two steps one steps is a map and another step is reduce. The work of map is to find the set of specific data from a group of data and work of reduce is to aggregate those specified find out data.

Algorithm

Step 1: Initially, send an application to resides the data where data has been located.

Step 2: Perform map reduce tasks such as map, shuffle and reduce on data.

- The map will process the data which was stored in the Hadoop system and data can be any format either in form of structure or semi-structure data. The mapper function will check the data sequentially. Then map will process the data and divide into smaller chunks of data sets.
- In reduce both shuffle and reduce task has been integrated. It process data coming from map. Once it is done it will produce new set of data and then it will be transferred to HDFS.

5. Experimental Analysis

In this section, we assess the recital of the projected model Hadoop map-reduce (HMR) and judge against it with three models PFP, SPC, FIMMR on two datasets: T10.I4.D10000K and T20.I10.D10000K. These four models are accurate models, i.e., they will obtain the similar out come in similar circumstances; so our goal is to judge against the execution time of these models in different constraints. All the above models were written in java programming language. The above two datasets were created by the international business machine (IBM) [16].

The investigational platform is a group of 37 nodes, together with 1 key node, 1 restraint node, and 34information or data nodes. The hardware of every node is assembled on 2.6 Ghz dual-core central processing unit (CPU) and 10GBrandom access memory (RAM) cache memory; software design is on Ubuntu 12.04 and Hadoop 0.22.0. Every node can hold2 map processes concurrently. To make complete use ofall34 information or data nodes, every testing information or data file is slice in to 24

parts with identical sizes. Table 1 and 2 illustrates the execution time under distinct least support threshold for the dataset T10.I4.D10000K and T20.I10.D10000K correspondingly.

Table 1. Execution time under distinct least support threshold for the dataset T10.I4.D10000K

Least Support Threshold in %	**Execution Time Per Second**			
	PFP	**SPC**	**FIMMR**	**HMR**
2.0	205	141	52	12
1.8	2015	156	154	25
1.6	4067	257	257	36
1.4	6089	1804	358	49
1.2	8078	4023	454	56
1.0	9050	7512	557	62

Figure 2 and 3 illustrates the execution time of the four models PFP, SPC, FIMMR and HMR in various least thresholds on two datasets. It is clear that the model HMR better perform than the other three models noticeably. Since the recurrent item-sets are more for smaller the least support threshold, and also the execution time, recurrent items will boost rapidly along with the decline of the least support threshold. PFP produces too many lengthy sub-transaction item-sets, SPC produces too many contenders, FIMMR produces better global contenders item-sets in the initial round of map-reduce against the other two models. On the other hand, HMR produces better global contenders' item-sets in the initial round of map-reduce against the all three models. The number of contenders may reduce considerably, in several cases, the second round of map-reduce including support numbers of the contenders is no more needed, so the performance of both HMR and FIMMR is more stable. Table 3 and 4 shows the scalability on distinct size of dataset T10.I4.D10000K and T20.I10.D10000K correspondingly.

Table 2. Execution time under distinct least support threshold for the dataset T20.I10.D10000K

Least Support Threshold in %	**Execution Time Per Second**			
	PFP	**SPC**	**FIMMR**	**HMR**
2.0	302	203	50	25
1.8	2208	404	210	42
1.6	3805	621	305	63
1.4	5704	2100	420	87
1.2	7925	3802	515	106
1.0	8723	7405	645	128

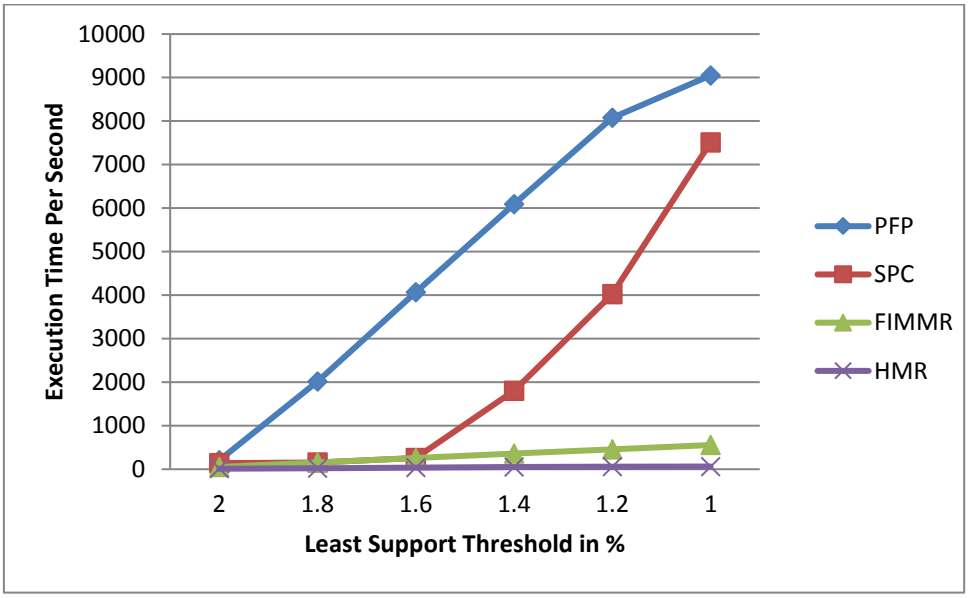

Figure 2. Execution time under distinct least support threshold for the dataset T10.I4.D10000K

Figure 4 and 5 assessments the time recital of the four models on distinct dimension of datasets; the least support threshold is 1.2% and 1.6% in Figure 4 and Figure 5 correspondingly. When the size of information varies, the vary of the number of recurrent 1.2 item-sets is not fairly considerable, but the amount of alike transactions for contender item-sets boosts relatively in SPC, and the amount of sub-transactions produced by PFP will also boosts rapidly, so mostly

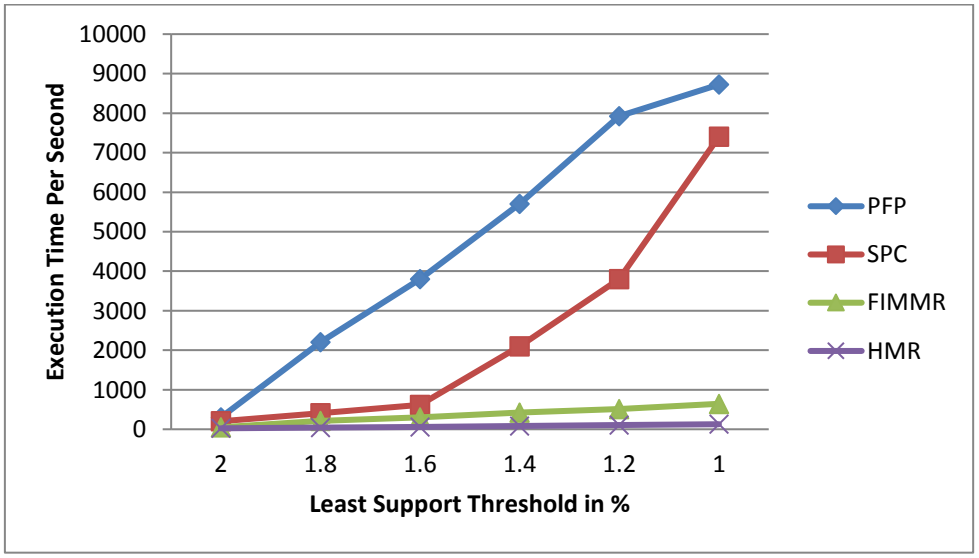

Figure 3. Execution time under distinct least support threshold for the dataset T20.I10.D10000K

Table 3. Scalability on distinct size of dataset T10.I4.D10000K

Dataset Size	Execution Time			
	PFP	**SPC**	**FIMMR**	**HMR**
1000 K	100	80	70	60
2000 K	215	150	120	105
3000 K	325	200	170	150
4000 K	440	270	220	185
5000 K	535	315	250	215
6000 K	610	375	280	240
7000 K	705	410	300	270
8000 K	790	470	320	295
9000 K	860	520	340	310
10000 K	890	600	370	330

Table 4. Salability on distinct size of dataset T20.I10.D10000K

Dataset Size	Execution Time			
	PFP	**SPC**	**FIMMR**	**HMR**
1000 K	110	90	80	65
2000 K	225	180	140	115
3000 K	345	240	200	160
4000 K	470	300	270	195
5000 K	565	370	320	225
6000 K	660	415	380	250
7000 K	745	450	415	280
8000 K	830	510	460	305
9000 K	890	570	480	320
10000 K	940	640	505	340

The execution time of SPC, PFP and FIMMR will boost relatively to the dimension of the dataset. However as for the proposed model, as illustrated in Figure 4 and 5 its execution time only pruning tactic, the next round of map-reduce including the support number of the contender is omitted, leaving

a more plane curvature for HMR. Table 5 and 6 illustrates speedup comparison on cluster size of dataset T10.I4.D10000K and T20.I10.D10000K Correspondingly.

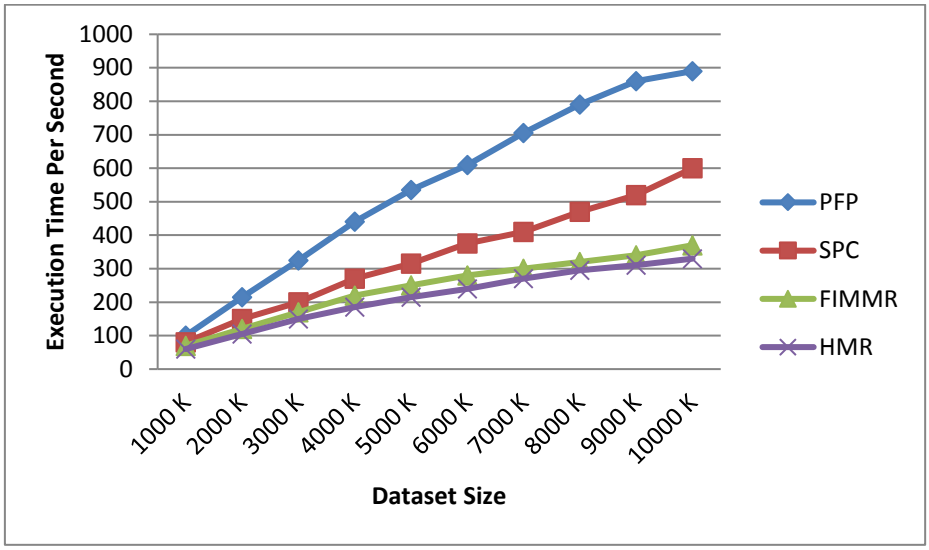

Figure 4. Scalability on distinct size of dataset T10.I4.D10000K ((LstSup = 1.2%))

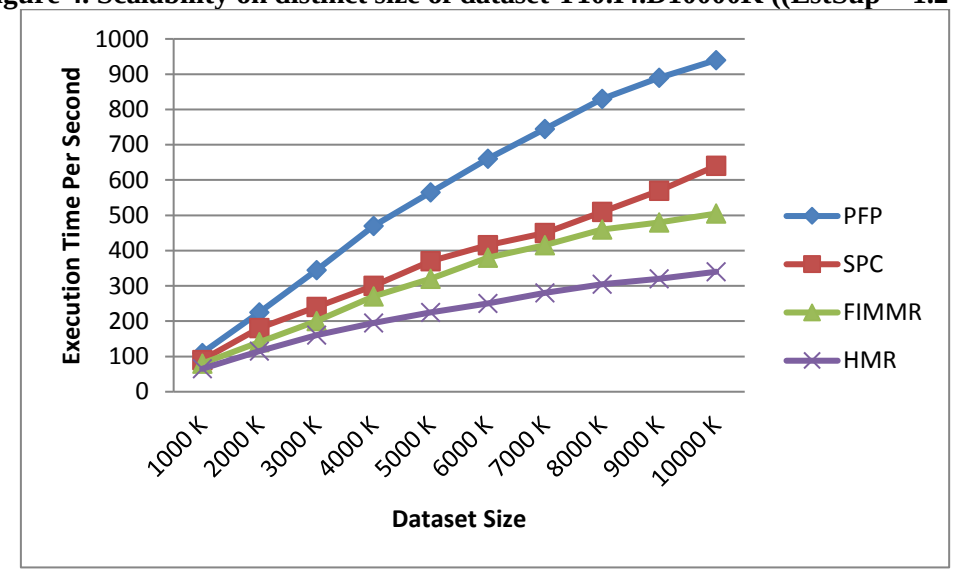

Figure 5. Salability on distinct size of dataset T20.I10.D10000K (LstSup = 1.6%)

Table 5. Speedup comparison on cluster size of dataset T10.I4.D10000K

Number of	Speedup

Nodes	PFP	SPC	FIMMR	HMR	Ideal
1	1	1	1	1	1
2	1.2	1.8	1.9	2	2
3	1.4	2.3	2.5	2.7	3
4	2.0	3.1	3.2	3.4	4
5	2.5	3.9	3.95	4.2	5
6	2.9	4.4	4.7	4.9	6
7	3.1	5.7	5.9	6.1	7
8	3.3	6.2	6.8	7.0	8
9	3.5	6.7	7.3	7.8	9
10	3.7	7.2	8.1	8.6	10

Table 6. Speedup comparison on cluster size of dataset T20.I10.D10000K

Number of Nodes	Speed Up				
	PFP	SPC	FIMMR	HMR	Ideal
1	1	1	1	1	1
2	1.35	1.85	1.91	2	2
3	1.55	2.35	2.52	2.76	3
4	2.1	3.14	3.23	3.46	4
5	2.6	3.95	3.97	4.28	5
6	3.1	4.43	4.72	4.96	6
7	3.3	5.74	5.94	6.17	7
8	3.5	6.25	6.83	7.07	8
9	3.7	6.74	7.35	7.84	9
10	3.9	7.23	8.14	8.66	10

Figure 6 and 7 is the speedup assessment of the four models on various cluster dimensions for two datasets. Speedup means the velocity of recital improved when accumulating more nodes to the group i.e.,speedup = old time / new time, where old time is the execution time of a model on a single node, and new time is the execution time on various nodes.

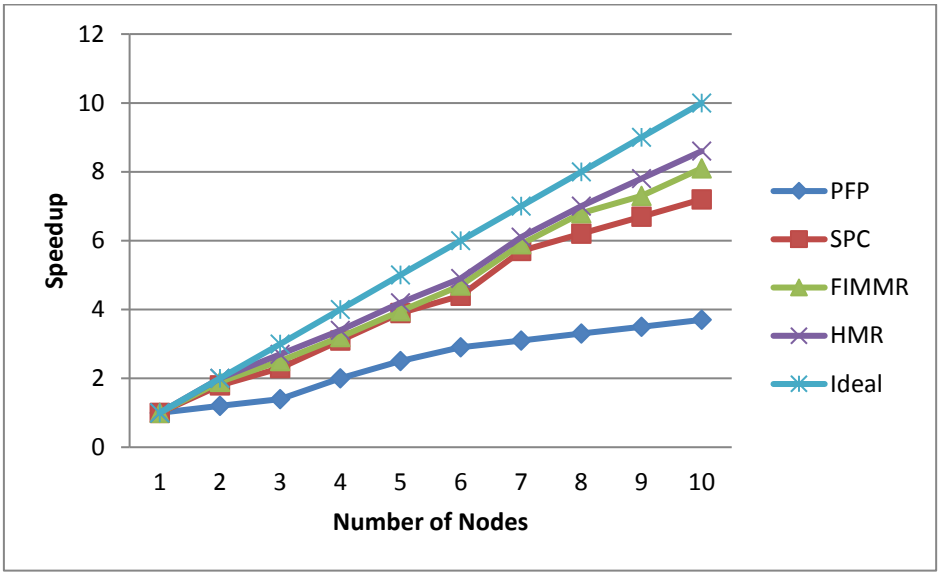

Figure 6. Speedup comparison on cluster size of dataset T10.I4.D10000K (LstSup = 1.2%)

We can see that the speedup of both HMR and FIMMR are near to the Ideal speedup (linear); the cause they can't be ideal is that when additional nodes are added, communication rate among nodes will also increase, and this increases the whole execution time. Model PFP, SPC can't dispense information evenly between nodes, so its whole execution time is considerably pretentious by the recital of the majority of heavily-loaded node. To appraise, the speedup of models FIMMR and HMR are acceptable.

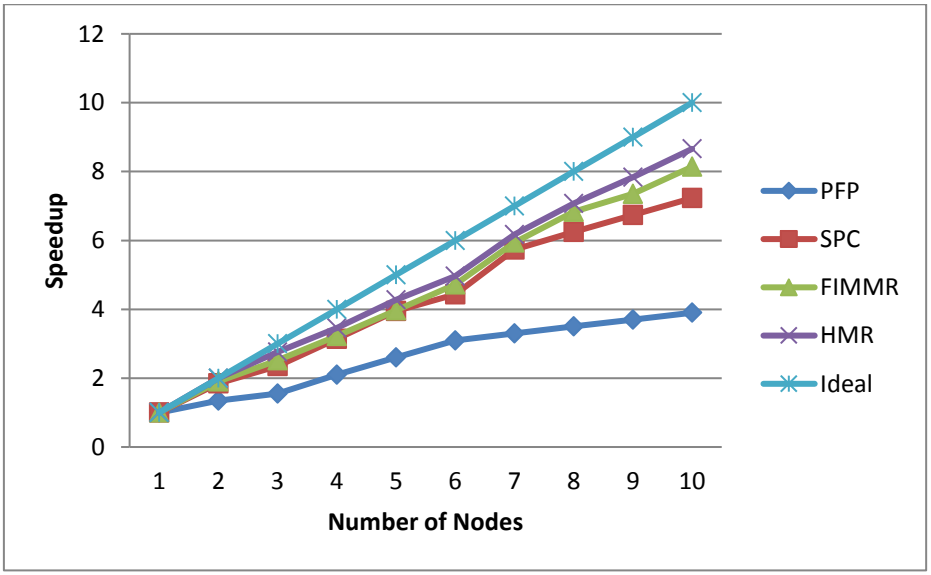

Figure 7. Speedup comparison on cluster size of dataset T20.I10.D10000K (LstSup = 1.6%)

6. Conclusion

We projected a recurrent item-set mining model HMR for the large data environment. HMR is constructed upon present recurrent item-set mining model of Hadoop and the map-reduce support. It initially introduces similar local recurrent item-sets as contender and the contenders are sorted with not empty. In next round of map-reduce is carried out to count the remaining contenders of global support

number. From experimental analysis we can analyze that the projected pruning strategy can sort the contender item-sets accurately, and in several cases, outcome of empty contender item-set, exposing the next round of map-reduce needlessly. The time accuracy of HMR outperforms FIMMR, SPC and PFP considerably using small least support threshold. HMR can get one order of degree enhanced with other three models; in the meantime the speedup of HMR is acceptable.

In future we plan to use spark in place of Hadoop. Spark has the feature of executing hundred times faster than Hadoop, executes in memory on the cluster and not integrated with two tasks. It helps in executing the data much faster than Hadoop. Spark is a single machine that works on the top of the HDFS. Spark will reduce data loads and execution time of large datasets.

References

[1] Zaki M J 1999 Parallel and distributed association mining: A survey *IEEE Concurrency* **7** 14-25
[2] Rathee S, Kaul M and Kashyap A 2015 R-Apriori: An efficient apriori based algorithm on spark *Proceedings of the Conference on Partitioning Item sets Knowledge Mining* 12- 19
[3] Dean J and Ghemawat S 2008 Map Reduce: Simplified data processing on large clusters *ACM Communications* **51** 107-113
[4] Sakr S, Liu A and Fayoumi G 2013 *ACM Computing Surveys* **46** 11—37
[5] Wang L, Feng L, Zhang J and Liao P 2014 An efficient algorithm of frequent item sets mining based on map reduce *J. of Information & Computational Sc.* **11** 2809 -2816
[6] Yahya O, Hegazy E and Ezat E 2012 An efficient implementation of apriori algorithm based on Hadoop map reduce model *International Journal of Reviews in Computing* **12** 59-67
[7] Xia D, Zhou Y, Rong Z and Zhang Z 2014 IPFP: An improved parallel FP-growth algorithm for frequent item set mining *International J. of Computational Intelligence Sys.* **6** 17-27
[8] Curino C, Jones E, Zhang Y and Madden S 2010 Schism: A workload-driven approach to database replication and partitioning *Proceedings of the Conference on Very Large Data Bases Endowment* **3** 116-127
[9] Riondato M, DeBrabant J A Fonseca E and Upfal E 2012 Parma: a parallel randomized algorithm for approximate association rules mining in MapReduce *Proceedings of the 21st ACM international conference on Information and knowledge management* ACM 85–94
[10] Lu W, Shen Y, Chen B S and Ooi C 2012 Efficient processing of k nearest neighbor joins using map reduce *Proceedings of the Conference on Very Large Data Bases Endowment* **5** 1016-1027
[11] Li H, Wang Y, Zhang D, Zhang M and Chang E 2008 PFP: Parallel FP-growth for query recommendation *Proceedings of the ACM Conference on Recommender systems* 107-114
[12] Ahmed N S S, Acharya D P and Sanyal S 2017 A framework for phishing attack identification using rough set and formal concept analysis *International Journal of Communication Networks on Distributed Systems* **18** 186-212
[13] Ahmed N S S and Acharjya D P 2015 Detection of denial of service attack in wireless network using dominance based rough set *International J. of Advanced Comp. Sc. and Apps.* **6** 267-278
[14] Ahmed N S S and Acharjya D P 2015 A dominance based rough set approach for the detection of jamming attack *International J. of Philosophies in Comp. Sc.* **1** 45-66
[15] Ahmed N S S 2016 An application of containing order rough set for analyzing data of intrusion detection An Interdisciplinary J. of Scientific Research & Edu. **2** 52-57
[16] Wang L, Feng L and Jin B 2013 Sliding window-based frequent item sets mining over data streams using tail pointer table *International J. of Computational Intelligence Sys.* **7** 1-12