International Journal of Computer Science Engineering & Technology
APPROVED BY
NATIONAL SCIENCE LIBRARY (NSL)
NATIONAL INSTITUTE OF SCIENCE COMMUNICATION AND INFORMATION RESOURCES (NISCAIR)
COUNCIL OF SCIENTIFIC AND INDUSTRIAL RESEARCH (CSIR)- NEW DELHI INDIA .
ISSN :2455-9091

KNN-DP: HANDLING DATA SKEWNESS IN KNN JOINS USING MAPREDUCE

[1] R. Sandhoshraja, [2] Santhi Baskaran, [3] V. Geetha
[1, 2, 3] Department of Information Technology,
[1, 2, 3] Pondicherry Engineering College, Puducherry, India.

ABSTRACT: The MapReduce programming model has been successfully used for big data analytics. However, data skewness invariably occurs in big data analytics and seriously affects efficiency. To overcome the data skewness problem in MapReduce, in the past proposed a data processing algorithm called kNN-DP with reallocatedfree space partition technique. The robustness and efficiency of the algorithm were tested on a wide variety of simulated datasets. The results showed that kNN-DP with Reallocate Free Space partition technique can prevent data skew in MapReduce efficiently. Data skewness is not fully rectified. While applying the result of the reallocated free space (in phase-1) as a input to the proposed algorithm called repartitioning algorithm (PTSH-Partitioning Turning based Skew Handling), it totally avoid the causes of the skewness. In comparison with the one-stage partitioning strategy used, PTSH uses a two-stage strategy and the partition tuning method to disperse key-value pairs in virtual partitions and recombines each partition in case of data skew. PTSH improves the performance of MapReduce jobs in comparison with the native Hadoop, Closer, and Locality-aware and Fairness-aware Key partitioning (LEEN). Even though PTSH is very effective, two-stage partitioning strategy is overhead. This overhead can be prevented by using spark. The time needed for rule extraction can be reduced significantly by adopting the PTSH algorithm using spark.

Key terms: [kNN-DP, Partitioning Turning based Skew Handling, LEEN.]

1. INTRODUCTION

The use of big data analytics in E-commerce offers many attractive opportunities while posing significant challenge. The traditional data processing and analytical algorithms cannot satisfy the requirements of big electronic gadgets data and cloud computing. Fortunately, advances in data management, particularly such parallel computational models as MapReduce, can be applied to process and analyse diverse and large-scale datasets. However, big data is so large and complex that it cannot be managed under traditional methods. For example, when using association rule mining (ARM) on MapReduce, algorithms must extract the necessary information from big data in a timely manner. MapReduce is a powerful and cost-effective tool for massively parallel analytics.

It can distribute data and computational tasks to thousands of cheap physical nodes, hence providing massive storage capacity and parallel computing capabilities. MapReduce is a programming model that allows the easy development of scalable parallel applications to process big data on large clusters of commodity machines.

A MapReduce job typically runs in two main phases: a map phase and a reduce phase. In each phase, distributed tasks process datasets on a cluster of computers. When a map task is completed, the reduce tasks are notified to pull newly available data. This transfer process is referred to as a shuffle. All map tasks must be completed before the shuffle part of the reduce phase to allow the latter to complete. We consider a case where computational load is unbalanced among map tasks or reduce tasks. We call such an unbalanced situation map skew or reduce skew, respectively. Skew can lead to longer job execution times and lower cluster throughput, thus affecting the performance of MapReduce.

Kwon [3] analysed the types of skew that arises in a variety of MapReduce applications but did not provide a relevant solution to unbalanced partitioning in the reduce phase. Ibrahim et al. designed the LEEN algorithm[5] to determine the corresponding partition of a map output based on the frequency of key-value pairs. When a large amount of data and keys are unevenly distributed, data skew may occur, resulting in an unbalanced input of reduce tasks. Xu et al. [6] focused on resampling partitioning strategy to deal with unbalanced partitioning in the reduce phase. However, when dealing with massive amounts of data, the sampling overhead incurred by this strategy is high and affects the performance of MapReduce.

Ramakrishna et al. [7] proposed techniques to split each key with a large record size into sub keys to allow for a more even distribution of workload among reducers. However, it requires waiting until all map tasks are completed to gather partition size information before reduce tasks can begin.

Considering this issue, we want to use a two -stage strategy to divide the map output into fine-grained partitions and recombine them based on global output information to disperse skewed data. In this paper, we propose a data processing algorithm called Partition Tuning -based Skew Handling (PTSH) to address the problem. First, we first use a virtual partitioning method to divide the original partitions into fine-grained partitions and collect real-time stats regarding the data size of each partition. Second, the partitioning g information of the map task is extracted and the corresponding index sent to the reduce tasks for repartition.

Finally, the repartitioning process divides the collected virtual partitions into new partitions of the same number as the reduce tasks. The main contributions of the paper lie on the following:

(1) Based on a two-stage partitioning strategy, we propose a partition tuning method to divide skewed partitions into fine-grained partitions and use a repartition method to solve the problem of unbalanced data division. As partitioning is an NP-hard problem, we propose a repartition algorithm, which can effectively balance skewed partitions.

(2) We conducted several experiments on simulated datasets and real datasets. Compared with one-stage strategies, the results showed that our method could effectively mitigate data skew in MapReduce jobs and improve efficiency.

(3) A case study of ARM for real healthcare data was carried out on MapReduce. Combining an Apriori algorithm and PTSH, it could balance the data distribution of reduce tasks and improve the efficiency of ARM on healthcare data.

2. LITERATURE REVIEW

This section provides the previous techniques and frameworks. It also provides the numerous methodologies.

Xujun Zhao, Jifu Zhang and Xiao Qin IEEE Transaction on parallel and distributed system. KNN DP Handling Data Skewness in kNN joins using MapReduce [1] 2017, Static optimization of the input data partition can improve the running time in the presence of skew, caused by uneven processing times. The optimization time is short when compared to the actual query execution time.

Y. Kwon, M. Balazinska, and J. Rolia International Conference on Management of Data. SkewTune: Mitigating Skew in

MapReduce Applications [7] 2012, Repartitions the load automatically and detects the straggler node and tries to mitigate by allocating the task to the idle node.

Y. Xun, J. Zhang, and X. QinIEEE Transaction on Parallel and Distributed System Fidoop-DP: Data Partitioning in Frequent Itemset Mining on Hadoop Clusters [2] 2017, Redundant transaction is reduced. Not suitable for heterogeneous clusters.

A. Stupar, S. Michel, and R. Schenkel Large scale Distributed System for Information Retrieval. Rank reduce processing K nearest neighbour queries on top of MapReduce, [3] 201, It exploits the inherent parallelism available in the FP-Growth algorithm. It is suitable only for very large sets of data.

M. Jang, Y.S. Shin, and J.W. Chang IEEE International Conference on High Performance Computing and Communication. A grid-based knearest neighbor join for large scale datasets on MapReduce [5], 2015, Performance of processing large-scale hierarchical data in distributed scientific applications is high. MapReduce application's fault tolerance capability appears as a high overhead construct.

Q. Chen, Y. Yao and Z. Xiao IEEE Transaction on Parallel and Distributed Systems. Libra: Lightweight data skew mitigation in MapReduce [6], 2015, Real data driven experiments validate the efficiency and effectiveness.

3. METHODOLOGIES

KNN assumes that the data is in a feature space. More exactly, the data points are in a metric space.

The data can be scalars or possibly even multidimensional vectors. Since the points are in feature space, they have a notion of distance – This need not necessarily be Euclidean distance although it is one of the commonly used methods.

The kNN uses training data as reference to classify the new data points collectively called testing dataset. Each of the training data consists of a set of vectors and class label associated with each vector.

In the simplest case, it will be either + or – (for positive or negative classes). But kNN, can work equally well with arbitrary number of classes. A single number 'k' is given.

This number decides how many neighbors (where neighbors are defined based on the distance metric) influence the classification. This is usually an odd number if the number of classes is 2. If k=1, then the algorithm is simply called the nearest neighbor algorithm.

The outcome of kNN-DP is the partitioned Data with some skewness and load imbalance. To reduce the load Imbalance ,the free space should be reallocated to the remaining filled partition. Then repartitioning Techniques is used to repartition the data and reduce the data skewness.

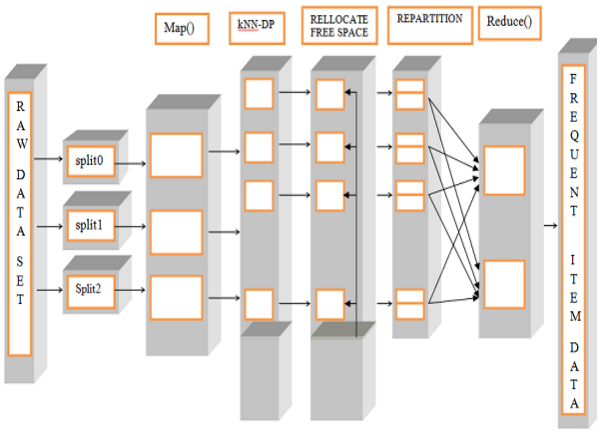

Figure 3.1 Data Partitioning Architecture

In MapReduce ,the mapper checks for the redundant data in all the available nodes. Once the data is mapped the job tracker and task tracker are started and parse the whole data and fetch the relevant data to the reduce phase. The data is balanced in this section and reducer iterate the output data. The output from reducer is pushed to the different partitions and kNN-DP is performed to reduce the data skewness. The outcome of the kNN-DP doesn't effectively reduce the data skewness and some of the partitions are left empty. To utilize free space, reallocate the partitions. Then the result will be repartitioned using the repartitioning technique to reduce the data skewness.

3.1 kNN-DP

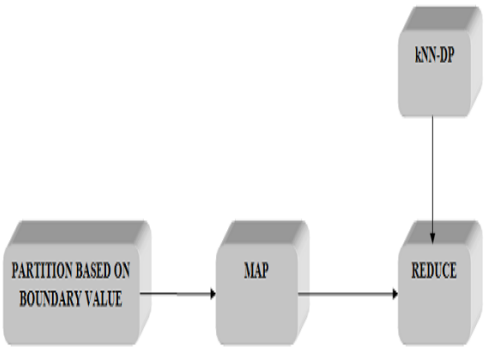

Figure 3.2 kNN-DP Design

An all k-nearest neighbor Data Partition (kNN-DP) is a variation of a k-nearest neighbor query and determines the k-nearest neighbors for each point in the given dataset in one query process. It is a useful operation for batch-based processing of a large distributed point dataset. Consider, for example, a location-based service which recommends each user his or her nearby users, who may the candidates of new friends.

Given that users' locations are maintained by the underlying database, we can generate such recommendation lists by issuing a kNN query (e.g., k = 5) on the database. In a centralized database environment, we can use the existing kNN-DP algorithms.

Although efficient algorithms for kNN queries are available for centralized databases, we need to consider supporting distributed environments where the target data is managed in multiple servers in a distributed way. User location information of a location-based service in the above example may be distributed in many servers. In such a case, we need to consider using cloud computing technologies for efficiently executing queries.

Especially, MapReduce, which is a fundamental framework for processing large-scaled data in distributed and parallel environments, is a promising method for enabling scalable data processing. In our work, we focus on the use of Apache Hadoop since it is quite popular software for MapReduce-based data processing.

3.2 Reallocate Free Space Partition

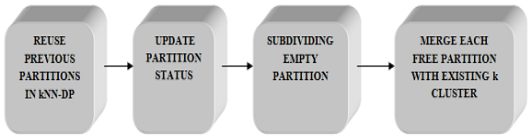

Figure 3.3 Reallocate Free Space Partition Design

In kNN-DP the partitions not fully utilized and some of the partition are empty. These Empty partition can be reallocated to the existing partitions .This can be done by calculating total free space in each partitions per total number of partitions and merge the sub divided empty partition to the existing partitions. Therefore all the partitions will be effectively utilized. It results in effective utilization of space and task allocation to each partition. Reallocate Free Space Partition (RFSP) utilize the previous partitions result of kNN-DP. It update the status of the each partition whether its empty are not.

If the partition are effectively utilized then it will proceed with the process or else empty partitions will be reallocated evenly to the existing partitions in balanced manner. Therefore all the partitions will be effectively utilized. It results in effective utilization of space and task allocation to each partition.

3.3 MapReduce Using Repartitioning Techniques in Sparks

(i) Partitioning Turning-Based Skew Handling Approach

Based on the virtual partition in the map phase, the repartition in the reduce phase recombines the virtual partitions into new partitions to ensure that the number of reduce tasks is equal to the final number of new partitions. Meanwhile, the size of new data in each partition maintains a certain balance.

(ii) Virtual Partitioning in Map Phase

After all map tasks are completed, all key-value pairs are sorted by partition number. Inside the partition, all key-value pairs are sorted following the key order. When

dealing with large-scale datasets, the output data generated by each map task usually occupy a large amount of memory, which is spilled to the local disk. All spilled files are then me raged and written to the disk after all map tasks are completed. Throughout the process of spilling and merging, the index corresponding to each partition is established by the map tasks.

When reading data, it can speed up the task of obtaining subsequent d ate for the reduce partitions. In the repartitioning process, the partition results in the map phase are divided and combined once again. The key-value pairs in one partition are hence separated and merged into another. When a reduce task requests partita ion data based on the results of a new partition, the requested data is distributed in different places in the spilled files, resulting in a consequential and inefficient reading of data. The key challenge in virtual partitioning is choosing the partition number of key-value pairs R in function hash key %R. By default, R is the number of reduce tasks; but, ideally, R should be determined by the number of types of input key-value pairs.

We think that the appropriate number of virtual partitions is between these two values. When the value of R is determined, the partition number is no longer correspondent to the reduce task number through hash key %R. The data in each partition in the map phase can be processed by an uncertain reduce task; such a partition is called a virtual partition. Each virtual partition is an integral part of an actual partition that has been repartitioned.

(iii) Obtaining Global Output Information

Based on the global output metadata of map tasks, the repartitioning process makes full use of the communication between map tasks and reduces tasks to divide the original communication process into two phases: (1) Obtaining the metadata output of each map task and (2) Recombining the information in the reduce tasks.

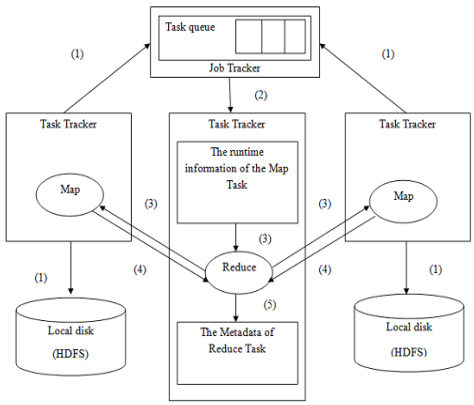

Figure 3.4 the process of acquisition of metadata for reduce tasks.

(iv) Repartitioning

The repartitioning process divides the collected virtual partitions into new partitions of the same number as reduce tasks. The data size of the biggest partition can be minimized after repartitioning process. It can also reduce the processing time needed for the maximum partition, thereby speeding up the completion of the entire reduce phase and increasing the rate of completed jobs as well as system throughput.

4. EVALUATION

All experiments to measure the performance of PTSH were performed on a 7-node cluster with six slave nodes and onemaster node. Each node used two 2 GHz quadcore CPUs with 16GB of RAM and 500GB SATA disk drives. All nodes were used as both compute and storage nodes. The HDFS block size was set to 64MB, and a common gigabit Ethernet SwitchconnectedeachnodeWeevaluatePTS Hperformance on a virtual cluster: five virtual machines were deployed on each of the six machines, reaching a cluster size of 30 data nodes. All virtual machines were configured with one CPUand 1GB memory. The baseline for our deployment was Hadoop 1.1.2 [19], and we configured the HDFS to maintain three replicas for each data block in this cluster.

Measures of Data Skewness and Data Locality. Some distributions of data, such as the bell curve, are symmetric. Thismeansthattherightandtheleftpartsofthedi

stribution are perfect mirror images of each other. Not every distribution of data is symmetric. We know that data skew arises out of the physical properties of objects and hotspots on subsets of the entire domain (e.g., the word frequency appearing in documents obeys a Zipfian distribution). The measure of how asymmetric a distribution is is called skewness and is used as a fairness metric in the literature [20]. We use the coefficient of variation to numerically calculate the measure of data skewness as follows:

Cov= stdev/ mean ×100% 3 The data distribution is completely fair if the coefficient of variation is zero. As Cov increases, skewness does as well. Data locality is important for performance evaluation. The data locality is the sum of the frequencies of keys in nodes, which are partitioned to that of the frequencies of all keys [5]:

$$\text{Locality}_{\min} = \frac{\sum_{i=1}^{K} \min_{1\le j\le n} FK_i^j}{\sum_{i=1}^{K} FK_i},$$

$$\text{Locality}_{\max} = \frac{\sum_{i=1}^{K} \max_{1\le j\le n} FK_i^j}{\sum_{i=1}^{M} FK_i},$$

where mini≤j≤nFKj i indicates the minimum frequency of key ki in data node nj and maxi≤j≤nFKj i is the maximum frequency of key ki in data node nj.

Performance of PTSH on Applications. First, to compare native Hadoop and PTSH, we performed our evaluations on PUMA [11], which represents a wide range of MapReduce applications exhibiting characteristics with high/low computation and high/low shuffle volumes. Second, we evaluated PTSH with Closer [9], LEEN [5], and native Hadoop through the Word Count application. The applications used in our evaluation were as follows:

Inverted Index (II): It takes a list of documents as inputs and generates word-to-document indexing.Map emits <word, docId> tuples with each word emitted once per docId. Reduce combines all tuples on key<word>and emits<word, list(docId)>tuples after removing duplicates.

Word Count (WC): This application counts the occurrences of each word in a large collection of documents. Map emits <word,1> tuples. Reduce adds the counts for a given word from all map tasks and outputs the final count. Weused frequent item dataset [22] for Inverted Index and generated skewed data by RandomWriter [23] for Word Count. In our experiments, we used the frequency variation of the keys and their distribution as parameters in the motivation of the design. Since the former clearly causes variation in the data distribution of the inputs of the reducers, the variation in the latter affects the amount of data transferred during the shuffle phase [5]. We present the results of executing these applications with varying sizes of input data, frequency of variationofthekeys, and averagevariationinkeydistribution. We ran each application at least five times and used the average performance results.

The number of virtual partitions depends on the tuning ratio (TR) set by user, which can be computed as follows:

$$\text{TR} = \frac{V}{R}.$$

In the above, V is the number of virtual partitions and R is the number of reduce tasks. To compare the proposed algorithm with the native Hadoop system, we ran each application by using the PTSH algorithm with different partition turning parameters. The value of TR varied from 1 to 50. When TR=1, this meant that PTSH was not used and reached the uniform distribution of each key among the data nodes (key distribution variation=0%). However, in the map phase, the combining process affected the amount of data to be transferred during the shuffle phase, emphasising the amount of input data for the reduce tasks. Therefore, the map native combine was not a factor in our experiments

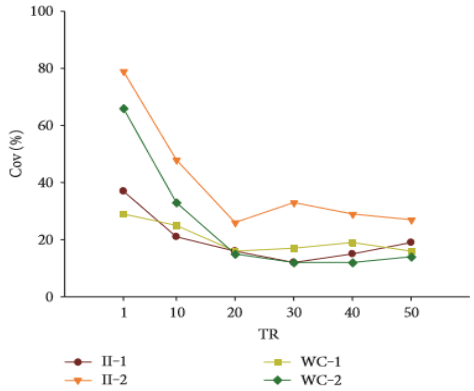

Figure 4.1 Performance of II and WC with different variations in the frequency of keys as well as different key distributions.

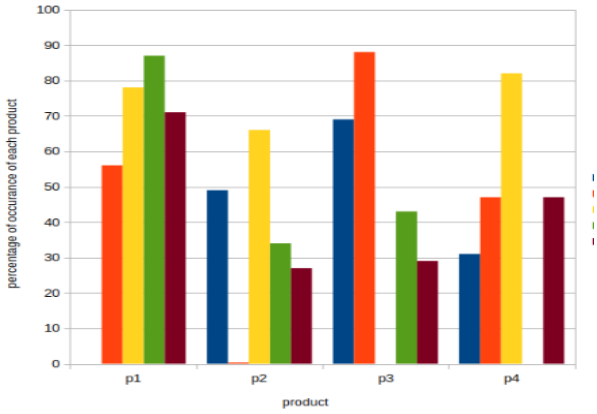

Figure 4.2Percentage of occurence of each product

CONCLUSION

Big data is changing impact in many ways, such as shopping, social media, and education. One of the most promising areas where big data can be applied for improvements is E-commerce. The fields of E-commerce generate large volumes of data, for instance, electronic medical records. Both the volume and the velocity of data in e-commerce are truly sufficiently high to require big data today. Understanding these data with methodologies using big data processing can help analytics for financial analysis, and fraud and waste monitoring. PTSH algorithm proposed to balance the input data of reduce tasks, which aims to process data related to e-commerce. Performance studies carried out on a seven-node MapReduce cluster showed that PTSH outperformed native Hadoop, Closer, and LEEN.

Compared with one-stage partitioning strategies, two -stage partitioning can mitigate skew data in reduce tasks. It was found that data skewness and workload balance simultaneously influenced the efficiency of MapReduce. The experimental results showed that MapReduce is sensitive to workload balance, although good skewness is also important. MapReduce was effective in the best case of high balance and high skewness.

The combination of high balance and moderate skewness was the second-best case. The two-stage partitioning performed better, and PTSH improved the efficiency of E-commerce data. The current strategy requires obtaining all metadata outputs by map tasks before the reduce phase. In processing applications of large-scale data, overhead due to transmission between the map phase and the reduce phase may increase. This overhead can be reduced by using sparks via yarn.

REFERENCES

[1] Xujun Zhao, Jifu Zhang and Xiao Qin "kNN DP Handling Data Skewness in kNN joins using Mapreduce" U.S. National Science Foundation under Grant CCF–0845257(CAREER) 2017, pp. 1–14.

[2] Y. Xun, J. Zhang, X. Qin, and X. Zhao, "Fidoop -dp: Data partitioning in frequent itemset mining on hadoop clusters," IEEE Transactions on Parallel and Distributed Systems, vol. 28, no. 1, pp. 101–114, 2017.

[3] A. Stupar, S. Michel, and R. Schenkel, "Rankreduce -processing k-nearest neighbor queries on top of mapreduce," in Proc. 8thWorkshop on Large-Scale Distributed Systems for Information Retrieval, 2010, pp. 13–18.

[4] C. Zhang, F. Li, and J. Jestes, "Efficient parallel knn joins for large data in mapreduce," in Proc. ACM 15th International Conference on Extending Database Technology, 2012, pp. 38–49.

[5] M. Jang, Y.-S.Shin, and J.-W. Chang, "A grid-based k-nearest neighbor join for large scale datasets on mapreduce," in Proc. IEEE International Conference on

High Performance Computing and Communications (HPCC), 2015, pp. 888–891.
[6] Q. Chen, J. Yao, and Z. Xiao, "Libra: Lightweight data skew mitigation in mapreduce," IEEE Transactions on Parallel and Distributed Systems, vol. 26, no. 9, pp. 2520 –2533, 2015.
[7] YongChul Kwon, Bill Howe, Magdalena Balazinska and Jerome Rolia, "SkewTune: Mitigating Skew in MapReduce Applications", in Proc. of the ACM SIGMOD International Conference on Management Data, pp. 25-36, ACM, 2012.
[8] YongChul Kwon, Bill Howe, Magdalena Balazinska and Jerome Rolia, "A Study of Skew in MapReduce Applications", in Proc. of the Open Cirrus Summit, 2011.
[9] Benjamin Gufler, Alfons Kemper, Angelika Reiser and Nikolaus Augsten, "Handling Data Skew in MapReduce", International Conference on Cloud Computing and Services Science, CLOSER 2011.
[10] J. Vinutha and R. Chandramma, "Skew Types Mitigating Techniques to Increase the Performance of MapReduce Applications", International Journal of Emerging Technology and Advanced Engineering, ISSN 2250 – 2459 (Online), Volume 5, Special Issue 2, May 2015.
[11] V. A. Nawale and Priya Deshpande, "Survey on Load Balancing and Data Skew Mitigation in MapReduce Applications", International Journal of Computer Engineering & Technology, ISSN 0976 – 6367 (Print), ISSN 0976 – 6375 (Online), pp. 32-41, Volume 6, Issue 1, January 2015.
[12] QiChen, Jinyu Yao, and Zhen Xiao, "LIBRA: Light Weight Data Skew MIitigation in MapReduce", IEEE Transactions on Parallel and Distributed Systems, 26(9), pp. 2520-2533, IEEE, 2015.
[13] D. S. Tamhane and S. N. Sayyad, "Big Data Analysis Using HACE Theorem", International Journal of Advanced Research in Computer Engineer ing & Technology, ISSN 2278 – 1323, Volume 4, Issue 1, January 2015.
[14] YongChul Kwon, Magdalena Balazinska, Bill Howe and KaiRen, "Managing Skew in Hadoop", IEEE Eng. Bull., 36(1), pp. 24-33, 2013.
[15] Qifa Ke, Vijayan Prabhakaran, Yuan Yu, Yinglian X ie, Junfeng Yang and Jingyue Wu, "Optimizing Data Partitioning for Data-Parallel Computing", in proc. of the HotOS, pp. 13, 2013