ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Website: www.ijirset.com

Vol. 6, Issue 9, September 2017

A Survey on Partitioning Skew Diminishing Techniques in Hadoop MapReduce Environment

Y.Sravani Devi [1], K.Sindhura [2]

Assistant Professor, Dept. of Computer Science & Engineering, G.Narayanamma Institute of Technology & Science, Shaikpet, Hyderabad, India[1]

Assistant Professor, Dept. of Computer Science & Engineering, G.Narayanamma Institute of Technology & Science, Shaikpet, Hyderabad, India[2]

ABSTRACT: In the era of Big Data, it creates large size of structured and unstructured data. MapReduce is an effective tool for parallel data processing. One significant issue in practical MapReduce applications is data skew: the imbalance in the amount of data assigned to each task. This causes some tasks to take much longer to finish than others and can significantly impact performance. Parallel data processing is the spirit of the Apache's Hadoop. Hadoop is an open source execution of Google's MapReduce schedule. It has two components. Hadoop Distributed File System (HDFS) for storing data part and MapReduce for processing the data part. MapReduce has become a popular programming model for constructing data processing applications. While being widely used, existing MapReduce schedulers still suffer from an issue known as partitioning skew, where the output of map tasks is unevenly distributed among reduce tasks. It causes most of the tasks to take much longer to finish than other tasks and can significantly impact on performance. This paper reviews the types of partitioning skew and several skew diminishing techniques to solve these issues.

KEYWORDS: MapReduce, partitioning skew, mitigation, partitioning.

I. INTRODUCTION

Now a days all the applications developed in internet are data intensive in nature.As we know that, the applications are generating huge volumes of data day by day.This makes the web indexing and data mining applications to access the huge data sets ranging from gigabytes to several tera or peta bytes.To process that huge data sets in a parallel fashion, MapReduce model was introduced.Due to its parallel processing,Scalability,flexibility and fastness, MapReduce has greatly increased in various applications like web indexing, log analysis, data mining, scientic simulations,machine translation, etc.

Mapreduce support various parallel computing frameworks like Hadoop [2], Google MapReduce, and Microsoft Dryad [3].These are currently used in many real time parallel data processing but there is still need for improvement in the scenarios called join, acceleration etc.,Among those Data skew is the most serious issue we observed.In data processing, the term data skew means that the imbalanced amount of data allocating to each task.Data skew can be differentiated in to two types:

1. uneven data input.
2. In processing the intermediate result among various computing nodes with respect to cost.

Skewness is the statistical term, which refers to the row distribution on a Map/Reduce model. If the data is highly skewed, it means some mapreduce are having more rows and some very less i.e. data is not properly/evenly distributed. This affects the performance/Teradata's parallelism. The data distribution or skewness can be controlled by choosing indexes.

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Website: www.ijirset.com

Vol. 6, Issue 9, September 2017

II. RELATED WORK

In MapReduce systems, each map task processes one piece of the input data, and generates a sequence of intermediate key-value pairs.

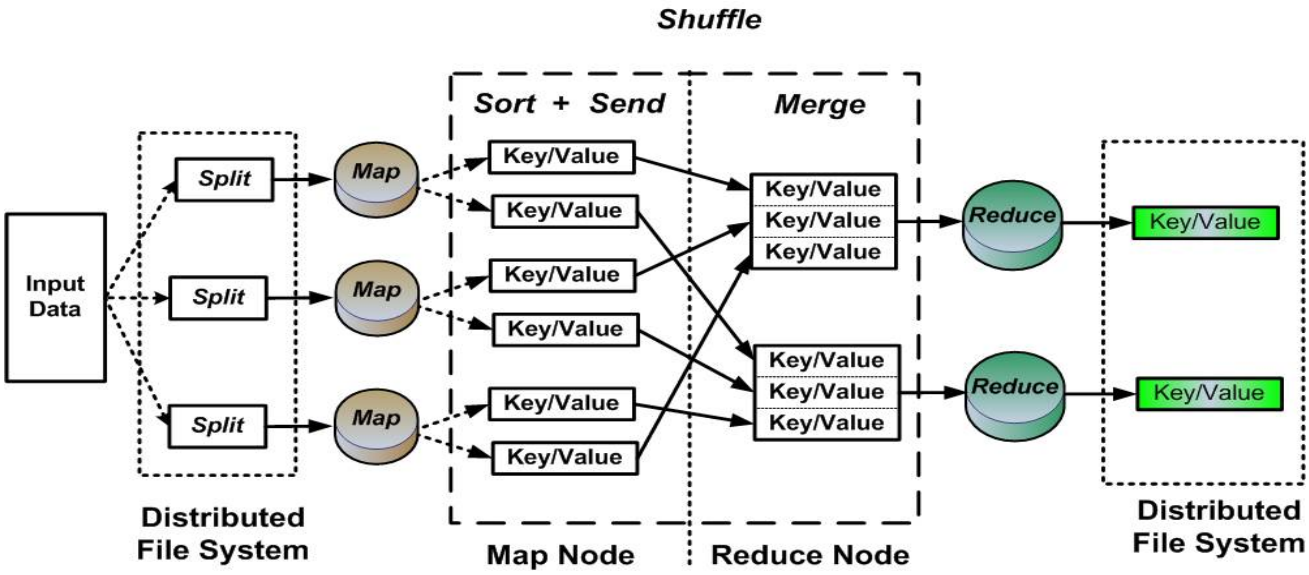

Fig. 1. MapReduce partitioning model.

During the reduce stage, each reduce task takes one partition (i.e., the intermediate key-value pairs corresponding to the same hash value) as input, and performs a user-specified reduce function on its partition to generate the final result. This process is presented in Fig. 1. Significantly, the hash function is expected to generate equal structure of partitions if the key frequencies, and sizes of the key-value pairs are equally distributed. However, in reality, the hash function often fails to achieve equal partitioning, resulting into skewed partition sizes. For example in the Inverted Index job [9], the hash function divides the intermediate key-value pairs based on the occurrence of words in the files. Therefore, reduce tasks processing more popular words will be assigned a big number of key-value pairs. As shown in Fig. 1, partitions are no uniformly distributed by the hash function. P_1 is greater than P_2, which causes workload imbalance between R_1 and R_2. Zacheilas and Kalogeraki [3] presented the following reasons of partitioning skew:

- Skewed key frequencies: Some of the keys occur more frequently in the intermediate data. As a result, partitions that consists of these keys become extremely large, thereby overloading the reduce tasks that they are assigned to.
- Skewed tuple sizes: In MapReduce processes where sizes of the values in the key-value pairs manipulate significantly, even though key frequencies are uniform, uneven.

In a MapReduce approach, a classic job execution consists of the following steps:

1) Once the job is submitted to the Map-Reduce system, the input data are divided into different parts and assigned to a group of map tasks for parallel processing.

2) Each map task converts its input (Key, Value) tuples in map node into intermediate (Key, Value) tuples in reduce node based on some user defined map and diminishing functions, and outputs them to the local disk.

3) Each reduce task copies its input parts from all map tasks, sorts them into a single stream by a multi-way merge, and generates the final (Key, Value) to distributed file system results according to some user defined reduce function.

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Website: www.ijirset.com

Vol. 6, Issue 9, September 2017

In the above procedure, the intermediate data generated by a map task are divided according to some user defined partitioner. For example, Hadoop uses the hash partitioner by default. Each partition is written as a continuous part of the output file. Since all map tasks use the same partitioner, all tuples that share the same key will be given to the same partition. We call these tuples as a cluster. In the result, the number of clusters is equal to the number of distinct keys in the input data. Each reduce task copies its partition (containing multiple clusters) from every map task and pro-cesses it locally.

Partitioning skew refers to the no uniform distribution of data assigned to each task, or the imbalance in the amount of work required to process such type of data. The job completion time in MapReduce depends on the slowest running task in the job.

We can say this is as straggler. If one task in job takes longer to finish than others (the so-called straggler), it can delay the progress of the entire job. Stragglers can occur due to various reasons, among which data skew is an important one. Stragglers are caused by some external factors such as hardware faults, slow running machines, etc., speculative execution is a best solution where the slow task also runs on an alternative machine with the hope that it can finish there faster. Google has observed that speculative execution can decrease the job execution time by 44 percent [1]. Unfortunately, when a straggler is caused by data skew (i.e., it has to process more data than the other tasks), it cannot be solved by simply duplicating the task on another machine.

Data skewness causes certain tasks with heavy workload run slower than other tasks. This in turn extends the job finishing time. Data skew is not a new problem specific to MapReduce. It has been studied previously in the parallel database.

III. TECHNIQUES FOR DIMINISHING PARTITIONING SKEW ISSUES

Data skew is nothing but the imbalance in the amount of data assigned to each task. The fundamental cause for the occurrence of data skew is that the user doesn't know the type of data to be distributed beforehand.

In Hadoop dataprocessing is done through MapReduce technique. Hadoop uses hash partitioner by default. For preserving efficiency every process must finish at sametime, but not all the tasks finish at once and some tasks may unusally take longer to complete thuis increasing overall complexity of the entire task.This is because slaves have different capacities in processing the data.

1. LIBRA (Lightweight Implementation of Balanced Range Assignment):

LIBRA does not need any pre-run sampling of the input data or prevent the overlap between the map and the reduce stages.

When the performance of the underlying computing platform is heterogeneous, LIBRA allocate the work-load accordingly and can deliver improved performance even in the absence of data skew. LIBRA implemented in Apache Hadoop and evaluate its performance for some popular applications. Experimental results show that LIBRA can improve the job execution time by up to a factor of 4.

There are three strong types of partitioners in previous work: hash, range and bin-packing [13], [14], [15]. The hash partitioner is the simplest works well only with the uniform data distribution. In the other two partitioners we use the range partitioner in our Libra system.

Enabling cluster splitting can have a profound impact on data skew diminution. If cluster splitting is not allowed, an entire cluster have to be allocated as a whole to a single reducer. If some keys in the distribution are far more popular than others, it can be difficult for even the best skew mitigation algorithm to perform well.

For example, suppose the intermediate data contain three keys: A, B, C and D. The count of them is 100, 30,10 and 10. Now we want to partition them into three reducers for processing. When cluster split is not allowed, the best solution is shown in left bottom corner of Fig. 3 which assigns the large key A to $reducer_1$ and the rest keys to $reducer_2$ and $reducer_3$. As we can see, it still exhibits data skew.

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Website: www.ijirset.com

Vol. 6, Issue 9, September 2017

LIBRA considers the performance of each worker node when partitioning the data. It assigns large tasks to fastest nodes and small tasks to slowest nodes so that all of them can be expected to finish around the same time. The following Figure gives an example on how LIBRA partitions its intermediate data in heterogeneous environments.

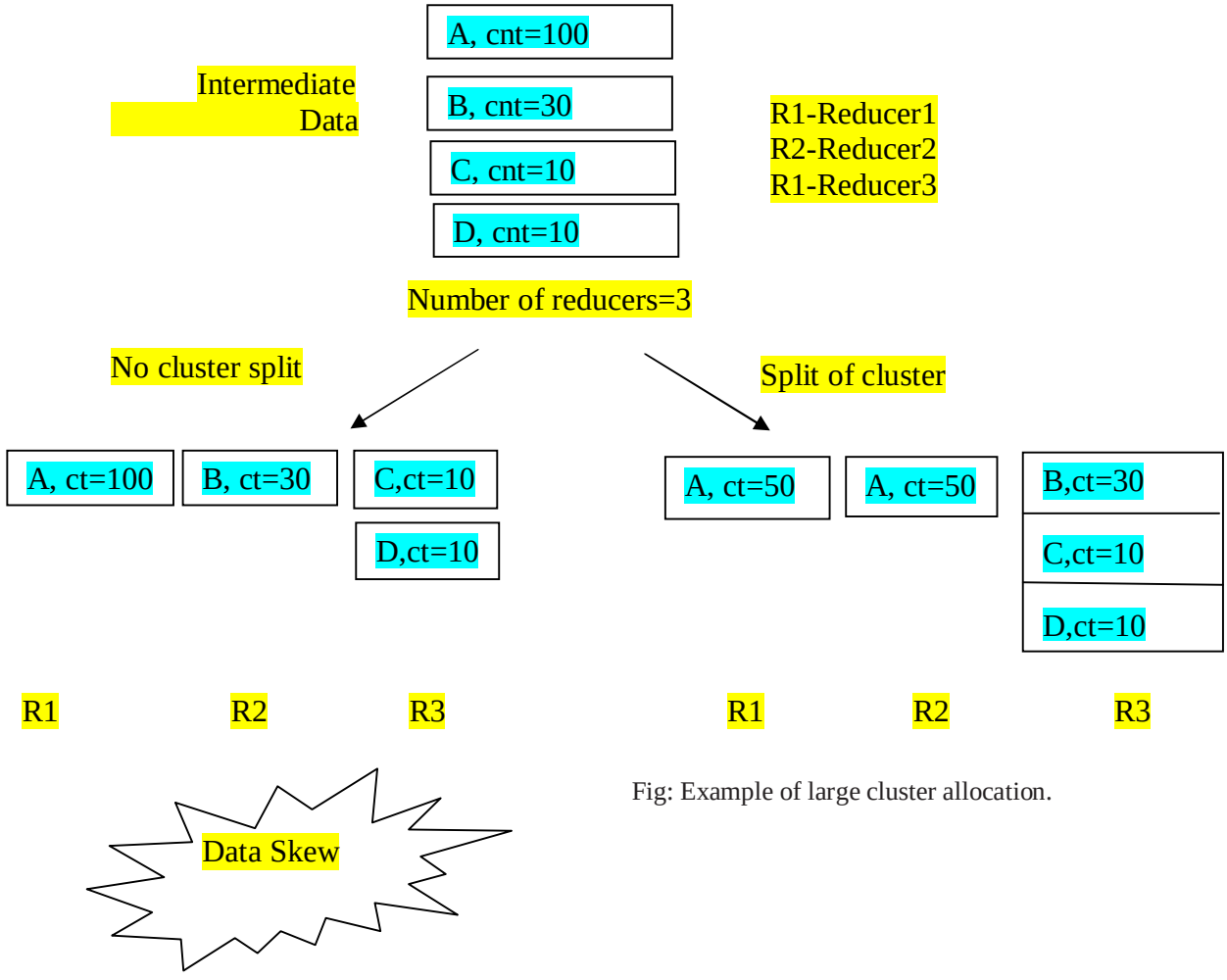

Fig: Example of large cluster allocation.

Drawbacks of LIBRA:
1. Static Data Splitting among nodes.
2. Uneven capacity of slave nodes.
3. Long running jobs.
4. Pre-sampling.

2. Block Chain Method:

An efficient dynamic data splitting strategy on Haddop, which monitors the samples while running batch jobs and allocate resources to slaves depending on the complexity of data and the time taken for processing. It comparatively reduces the data skew problem which eventually reduces the time complexity thus providing a successful overall output.

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Website: www.ijirset.com

Vol. 6, Issue 9, September 2017

The first step is to show how the data skew can be created. Here the data size is given statistically irrespective of the system size. So the start of the job in each slave is delayed by few seconds. So either slave 1 or slave 2 will start first, then the other two will start but not at the same time. The job completion is not the same for each slave. So here the book record information is taken as input. The size of the record to be processed can be thousand or 10,000 or even a lakh. So when we send 10,000 as input, the processing time to complete in slave 1 can be 37647ms, in another slave it is around 476829 m, in slave 3 it will be 32447ms. So the job completion depends on the specification of the system. Thus we can see the **data skew** is being **created** in this process. After the records are processed it is stored as an object file in each slave. Here skews are created as a batch process. In the next module, we use another set of input to process the slave to show the reducer side phase. The input taken is Product and CD data. This data is first sent to the block chain methodology to be processed first. Here all the 3 slaves start to process at the same time. The intermediate key is first processed. The data processing is dynamically balanced based on the specification of each slave, by migration of data to another server which processes soon. Balance and scheduling of resources are done according to the specification of each server. The reducer side skews are reduced by block chain methodology.

Algorithm structure:

Block chain()

{

Input dataset;
Processing the input to intermediate key;
Reducing the intermediate key into serialized object;
Save the serialized object in each slave's local;

}

The output is saved in each of the local storage where the object is saved asserialized object.

Advantages of Block Chain:
1. Dynamic data splitting strategy.
2. Process large sets of data.

3. Micro Partitioning technique:

Hadoop sorts terabyte of data using Tera sort method. Tera sort uses trie to partition the data among the reducers. A trie is a tree based data structure used for storing strings. In tera sort only two-level trie is used which considers only first two characters in the string. This two-level trie is build using cut-point algorithm. The cut-points are obtained by dividing the total number of strings to be inserted in trie by number of reducers. Using cut-points strings are divided among the reducers.

However without knowing the number of strings in each prefix two-level trie leads to load imbalance among the reducers. In order to overcome the load imbalance problem faced in using two-level Trie to partition the data, a technique called Xtrie is proposed to partition the data. In this for each word in the trie it maintains the counter value. Consider the example set of keys, {"act", "acid", "add", "adult", "born", "good", "grip", "grow", "peal", "pearl", "pedal", "soft", "stork", "storm"}.

The above strings are stored in the trie as in Figure 1. A trie for keys {"A", "to", "tea", "ted", "ten", "i", "in", and "inn" }. Consider there are 4 reducers and the above set is divided into 4 partitions. Each partition is Figure 1.Strings stored in the trie send to one reducer. Thus,
Reducer-1 process keys starting with {"to","te"}.Total:4 keys
Reducer-2 process keys starting with {"A"}.Total:1 keys
Reducer-3 process keys starting with {"in"}.Total:1 keys

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Website: www.ijirset.com

Vol. 6, Issue 9, September 2017

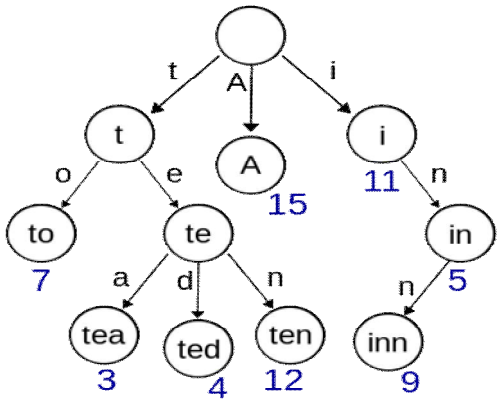

Fig. Strings stored in the Trie

4. Idempotent Task Cache System

The Idempotent Task Cache System (ITCS) is proposed to handle intermediate data skew in MapReduce on cloud computing. ITCS handles intermediate data skew stemming from the uneven key distribution in intermediate data. Unlike other proposals, ITCS neither changes the native hash function of a MapReduce system to redistribute intermediate data nor uses speculative execution to alleviate negative impacts of intermediate data skew. ITCS uses caches to decrease performance penalties resulting from the high workload of processing skewed intermediate data in certain Reduce tasks. In experiments, ITCS is tested with several popular cloud applications and compared with a native MapReduce system like Hadoop. According to experiment results, ITCS greatly outperforms the native MapReduce system. This method has contributions highlighted as follows. Using caches to handle intermediate data skew, which is extremely different from other proposals. Design the Idempotent Task Cache System (ITCS) to handle intermediate data skew at the premise of keeping intact the native hash function and speculative execution of a MapReduce system, which implies that our ITCS has high portability among the existing MapReduce systems.Implement ITCS and test it with several cloud applications to identify its performances, which indicates its high practicability. We observe that ITCS greatly outperforms a native MapReduce system like Hadoop, which shows that our ITCS indeed has a strong effect on performance improvement.

Exemplar based Inpainting technique is used for inpainting of text regions, which takes structure synthesis and texture synthesis together. The inpainting is done in such a manner, that it fills the damaged region or holes in an image, with surrounding colour and texture. The algorithm is based on patch based filling procedure. First find target region using mask image and then find boundary of target region. For all the boundary points it defined patch and find the priority of these patches. It starts filling the target region from the highest priority patch by finding the best match patch. This procedure is repeated until entire target region is inpainted.
The algorithm automatically generates mask image without user interaction that contains only text regions to be inpainted.

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Website: www.ijirset.com

Vol. 6, Issue 9, September 2017

IV. COMPARITIVE ANALYSIS

S.No	Techniques	Map/Reduce phase	Comparative Analysis
1	LIBRA	Reduce Phase	In this, it can reduce the straggler by using the cluster split in the partitioning technique. By using this cluster split obviously it minimizes the overhead. This technique can fit well in both homogeneous and heterogeneous environments.
2	Block Chain		Unlike LIBRA, it is an efficient data splitting strategy on Hadoop. It comparatively reduces the data skew problem which eventually reduces the time complexity thus providing a successful overall output.
3	Micro Partitioning Technique		The effectiveness of the load balancing in Tera Sort method was reduced by constructing the Trie. In a two-level Trie only first two characters of a string is considered during the Partitioning phase. Splitting the map output into many more partitions than reduce machines in order to produce smaller tasks. These tasks are assigned to reduce machines as workers become idle, allowing the task scheduler to dynamically mitigate skew and stragglers.
4	Idempotent Task Cache System.		A MapReduce system can avoid negative performance impacts of intermediate data skew with ITCS by using caches to skip the high workload of processing skewed intermediate data in certain Reduce tasks. In experiments, a MapReduce system is tested with several popular applications to prove that ITCS not only alleviates performance penalties when intermediate data skew happens, but also greatly outperforms native MapReduce systems without any help of ITCS.
5	DREAMS		This approach work for run-time partitioning skew mitigation. Unlike LIBRA it try to balance the reducers' workload by repartitioning the workload assigned to each reduce task.In DREAMS we cope with partitioning skew by adjusting run-time resource allocation to reduce tasks.

V. CONCLUSION

In MapReduce framework, the performance time of a task gets delayed by many reasons. One of the main reasons is partitioning skew in reducer side. This paper provides a survey of the various types of partitioning skew that arises in Hadoop MapReduce framework and also describes the techniques to diminish the skew such as LIBRA using cluster split, micro partitioning technique and DREAMS (dynamic) to improve the performance and try to avoid the data skew. In future research will be encompassed by providing new diminishing techniques for the skew issues in Big Data environment.

REFERENCES

[1] Qi Chen, Jinyu Yao, and Zhen Xiao, "LIBRA:Light Weight Data skew Mitigation in Mapreduce,," IEEE Trans. Parallel Distrib. Syst., vol. 26, no. 9, sep, 2015.
[2] Nawab Wajid, S. Satish and T. N. Manjunath, *"A Survey on Hadoop MapReduce Framework and the Data Skew Issues"*, International Journal of Scientific Research Engineering & Technology, ISSN 2278-0882, Volume 4, Issue 4, April 2015.
[3] M.Jenifer, B.Bharathi, " Survey on the solution of data skew in bigdata " ,2016 Online International Conference on Green Engineering and Technologies(IC-GET).

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Website: www.ijirset.com

Vol. 6, Issue 9, September 2017

[4] Zhihong Liu, Qi Zhang, Reaz Ahmed, Raouf Boutaba, Yaping Liu, Zhenghu Gong,"Dynamic Resource Allocation for Mapreduce with Partitioning skew" „" IEEE Trans. Parallel Distrib. Syst., vol. 65, no. 11, Nov, 2016.
[5] 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.
[6] 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 https://www.ijircce.com/upload/2014/march/14Q_A%20Micro.pdf
[7] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters", CommunACM 51:107–113, (2008).
[8] S. Ghemawat, H. Gobioff, S-T Leung, "The Google file system", Proceedings of the 19th ACM symposium on operating systems principles (SOSP), (2003).
[9] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," Commun. ACM, vol. 51, no. 1, pp. 107–113, 2008.
[10] (2016). Apache hadoop yarn [Online]. Available: http://hadoop. apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/ YARN.html.
[11] Y. Le, J. Liu, F. Ergun, and D. Wang, "Online load balancing for MapReduce with skewed data input," in Proc. IEEE INFOCOM, 2014, pp. 2004–2012.
[12] N. Zacheilas and V. Kalogeraki, "Real-time scheduling of skewed MapReduce jobs in heterogeneous environments," in Proc. 11th Int. Conf. Autonomic Comput., 2014, pp. 189–200.
[13] Y. Kwon, M. Balazinska, B. Howe, and J. Rolia, "Skewtune: Miti-gating skew in MapReduce applications," in Proc. ACM SIGMOD Int. Conf. Manage. Data, 2012, pp. 25–36.
[14] B. Gufler, N. Augsten, A. Reiser, and A. Kemper, "Handing data skew in MapReduce," in Proc. 1st Int. Conf. Cloud Comput. Serv. Sci., 2011, vol. 146, pp. 574–583.
[15] B. Gufler, N. Augsten, A. Reiser, and A. Kemper, "Load balancing in MapReduce based on scalable cardinality estimates," in Proc. Int. Conf. Data Eng., Apr. 2012, pp. 522–533.
[16] Q. Chen, J. Yao, and Z. Xiao, "Libra: Lightweight data skew miti-gation in MapReduce," IEEE Trans. Parallel Distrib. Syst., vol. 26, no. 9, pp. 2520–2533, Sep. 2015.