International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-9 Issue-1S4, December 2019

Improving Mapreduce Process By Introducing Aggregator Repartition Data for Big Data Analytics

K. Sathesh Kumar, S. Ramkumar, K.Shankar, M.Ilayaraja

Abstract: ***This work suggested data aggregator is used in between the mapper and reducer to enhance the performance of MapReduce. Initially the massive amount of data is partitioned into number of subset of data through the n number of independent mappers and it produces key value pairs for each partitioned data. Then the key value pairs are fed into aggregator where the data from different mappers are combining with smaller amount than the input. Followed by data aggregation data de duplication is carried over then repartition the data based on content, computation and network aware of data. Finally reducer merges the data to produce the final output, the proposed Content, computation and Network Aware (CCNA) MapReducer is compared with the existing Content Aware (CA) MapReducer and Content, computation Aware (CCA) MapReducer.***

Keywords: Data Aggregator, Big data, Aggregator Node, Mapreducer.

I. INTRODUCTION

A. Background

Big data analytics is most often related with the cloud due to the reason that the analysis of large data sets in real-time require a platform like Hadoop to build up large data sets over a distributed cluster and MapReduce to arrange, merge and process data from multiple sources.Data aggregation is the technique of converting distributed data fro numerous sources into a brand new one. The purpose of data aggregation can be to unite sources together by itself that the output is lesser than the input [15].This facilitates processing massive quantity of data in batch jobs and in real time applications.This diminishes the network traffic and improves the performance while in progress. This is required for the big data handling management to gain meaningful information from the massive quantity of data.Data aggregation is basically a type of data and information mining process where data is explored, grouped and viewed in a report-based, summarized format to achieve specific business objectives or processes and/or conduct reading people[10] [11]. Data aggregation's key applications would be the grouping, utilization and management of data that really is available and present upon the global Internet [1-3].

Manuscript published on 30 December 2019.
***** Correspondence Author (s)
K. Sathesh Kumar*, School of Computing, Kalasalingam Academy of Research and Education, Krishnankoil, Virudhunagar, Tamil Nadu, India, sathesh.drl@gmail.com.
S. Ramkumar, School of Computing, Kalasalingam Academy of Research and Education, Krishnankoil, Virudhunagar, Tamil Nadu, India, ramkumar.drl2013@gmail.com.
K.Shankar, School of Computing, Kalasalingam Academy of Research and Education, Krishnankoil, Virudhunagar, Tamil Nadu, India, shankar.k@klu.ac.in.
M.Ilayaraja, School of Computing, Kalasalingam Academy of Research and Education, Krishnankoil, Virudhunagar, Tamil Nadu, India, ilayaraja.m@klu.ac.in.

Data aggregation usually performs on huge data or data marts that don't offer much valuable overall information. Data de-duplication happen to be the technique of eliminates duplicates within the dataset. The duplicate copies of data increases the redundancy, memory and the storage cost. It can be overcome by data de-duplication process, it identifies the unique chunks of data and stored in a memory and it referred the remaining chunks in the dataset when the redundant chunks occur it replaced with a small reference from the stored chunks. Data repartition partition the data based on some characteristics of data such as content aware, computation aware or network aware [4] [5]. In content aware partitioning users statistically fix the partitioning size based on their requirementsIn network aware partitioning in which the network is partitioned into sub partitions around on graph partitioning algorithm for distributing load for parallel computation. In computation derieved partitioning the partition relies upon computation sharing characteristics like, storage andprocessing capability [6-8]. By partitioning the data based on content, computation and network aware the accuracy will be increased, time consumption will be reduced and error rate will be reduced [9].

II. PROPOSED APPROACH

A. Mapper Node

The mapper node gets input as huge volume of data from different sources. Mapper node is responsible for partition the big data into number of sub data. First the partitioning column and the size would be decided.This intermediate data is hash partitioned regarding the various reduce tasks and printed towards the local hard disk drive of the worker executing the map task. Each map task practices a logical recreation of the input data that typically resides on any distributed file system. The map task applies the user-defined map function on each input record and buffers the consequential output.Then computation capacity of the corresponding resources would be calculated. After partition, key value indexing would be done for every partition. Afterpartitioning, mapper node will transfer the data's to the next intermediate node which is called as aggregator node. The algorithm for partition data [14] [13].

Algorithm 1: Partition data
INPUT: partition Range M, Size K, Data set T
OUTPUT: Sub data Block $r_1, r_2, \ldots r_n$

1. Extract range M and size K from meta data information
2. Partition (M, K, T)
a. Data Block A = get Parameters (M, K)
3. For each data in Data Block Ai
a. For each Computation node Cj
i. Check computation Property (data)
ii. If the computation property is efficient for data block A
1. Place the data block Ai in Cj
iii. Calculate processing Capacity
Calculate memory storage
iv. End if
b. End for
4. End for
5. End for
6. Return Partitioned Data PT

B. Aggregator Node

Aggregation is performed effectively by aggregate the data with the same intermediate key value generated by Mapper node.In that case data beginning the multiple sources may contain duplicate copies of records it degrade the storage efficiency and it also raise the storage cost it may be avoided by data de-duplication methodology.The proposed approach of the research combines merits all of the existing methodologies to provide a sufficient frame work for big data handling management Within this byte level chuck de-duplication is executed for taking from the redundant data's. This procedure partition the data set in a the consideration of content, computation and network traffic properties.

. The proposed work is given in figure 1.The procedure for byte level chuck de-duplication is explained in the algorithm 2. After data de-duplication repartition the data based on content, computation and network aware of data combining three techniques namely content, computation and network traffic aware repartitioning methodologies.

C. Computation Aware Based Partition

This powerful computation derived partitioning is useful to split the data's in sub sets units attracted by computation revealing qualities just like development potential, processing and storage space etc., in highly effective clusters may be achieved having the comparable qualities related to data's between the same clusters. With regard to computation derived partitioning technique, the results set put divided in sub units driven by unique computation capacity represent among the many employers as well as workers between the appropriate computation nodes. This data's along with same processing development qualities really are clustered collectively to develop the various clusters. The clusters related to identical type could be processed proficiently as a consequence of it's own unique characteristics. This computation partitioning is decidedly carried out primarily depending on overall the computation skew and numerous step computation characteristics. [11] {Processing related to computation derived partitioning obtains as shown below.

Each individual map task can be read start having the input shard that's allotted for it. It really parses the outcomes as well as yields ({key, value) pairs with regards to data of wonderful fascination. in parsing overall the input, the map function in fact possible to actually chuck from a large number of records that is related to no more interest. By processing several map workers do that in similar, we have the opportunity to linearly size behavior the task of transferring data. The stream of (key, value) pairs that in fact each individual worker produces is in fact buffered in storage memory as well as saved occasionally on top of the nearby disk considering the map worker. Each of these numbers really are divided in R regions from the partitioning function. The partitioning function is accountable for selecting which actually considering the R reduce workers could work from any exact key. The default partitioning function is only a hash of key modulo R an individual can easily change all of this by utilizing a customized partition capabilities in instances if there is basically wish to include particular keys important factors manufactured from the specific reduce worker.

Searching is required provided which will most likely be the event generally there occurs different situations considering the different keys as well as same key is going to map in the direction of the same cut down worker (same partition). After searching, incidences taking into consideration the same key are categorized collectively making sure it can be easy to know every one of the data that is related to a single key. most of the map workers have finished how they are going to act, the master results the reduce workers to begin with functioning. The primary fact a reduce worker must is to direct the data that it also must need to present towards the user's reduce function. The cut down worker connect with every single map worker via secluded processes calls to see the (key, value) data which has been aimed toward its partition. This data will certainly be selected from beginning with the keys. This part is often generally known as rearrange phase. The user's decline function could be known as utilizing keys implemented for searching data. The reduce worker calls the Reduce function as soon as for every single unique key. The function is handed two attributes: the key and of course the collection of moderate values that may become related to the key. The Reduce function profits sent to file.This work involved two stages namely map and reduce. In the map stage data partitioning would be done. In the reducer stage, data merging and aggregation would be done.

Algorithm 2: Aggregation, de duplication and repartition of data
Input: Partitioned Data Blocks $C_1, \ldots C_n$
Output: Aggregated Data Blocks
for each Partitioned data Block $C_i \in C$
Aggregate data Blocks
Perform de-duplication using byte level chunk de-duplication approach
Compute hash value of C_i
Byte level chunk de-duplication approach:
For every Data Block Except C_i
If hash values matches together then
Compare the hash values
Remove the duplicated copies
Else
Perform Data Fusion to reduce the network traffic cost
For each Data Block A_i
Extract the key value
Compare with key value of other data blocks except A_i
Repeat until null
If it matches together then
Fuse those blocks together
Else
go to next Data Block
End if
End for

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

Divide the Data Blocks into chunks
Compare with other data chunks of remaining data blocks except $_i$
Compute hash values of chunks
If it matches then
Remove the duplicated contents
End if
End if
End For

Total migration cost is calculated as

$$C_M(t) = \sum_{k,k' \epsilon R} \sum_{p \epsilon P} y_k^p (t-1) y_{k'}^p (t) \Phi_{kk'} \cdot \left(\sum_{\tau=1}^{t} \sum_{j \epsilon A} \sum_{k \epsilon R} g_{jk}^p (\tau) \right) + \sum_{j,j' \epsilon N} z_j (t-1) z_{j'} (t) \varphi_{jj'}$$

Change the key value based on re partition data
Select the reducer with minimized migration cost
Perform repartition based on content, computation and network aware of data
Forward the data Block to corresponding reducer
. End for

Algorithm 3: Enhanced Map Reduce (Proposed)
Input: Repartition Data Blocks RD
Output: Final Result
1. DataFinal = null
2. For each Repartition Data Blocks RD
3. DataFinal + = RD
4. End for
5. Emit (DataFinal)

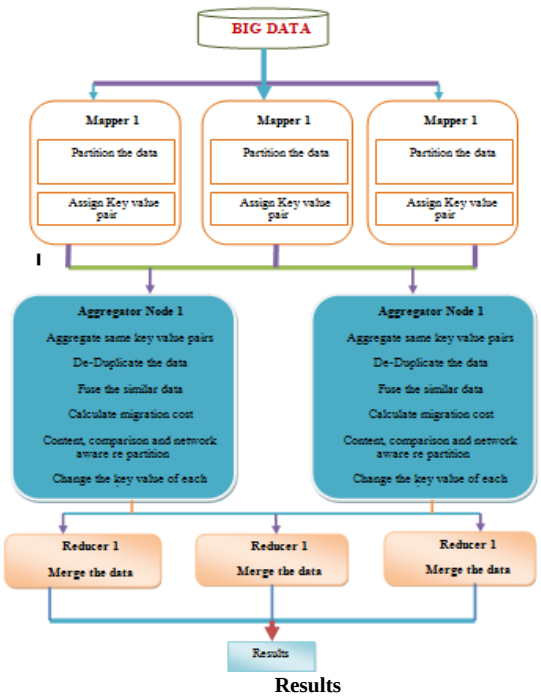

Figure 1 Overall Flow of the MapReduce

D. Description of Algorithm 1, 2 and 3

Algorithm 1 illustrates partition of data based on the partition range size and input dataset. In this algorithm, the partition range is initialized and big data is given as input. From Meta data information get the partition range and size of big data. Partition the data based on processing capacity, memory storage of each computation node. When the computation property is highly effective to carry the data block stored the particular data block into that specific node.Then aggregate the partitioned data using algorithm 2 and by using byte chunk de-duplication approach. In the byte chunk de duplication, the redundant data in aggregated data is removed based on the hash values of data block and select the reducer with minimum migration cost. Then re-partition the de-duplicated data with the consideration of content, computation and network aware of data to reduce the error rate and increase the accuracy of MapReduce and based on the repartition data change the key values of each reducer. Finally merge the data from different mappers by using final algorithm 3.

III. EXPERIMENTAL RESULTS

For the experimental purpose, the proposed Content, computation and Network Aware (CCNA) MapReducer is compared with the existing Content Aware (CA) MapReducer and Content, computation Aware (CCA) MapReducer are compared in terms Aggregation Time Vs Output Ratio, Partition Time Vs Output Ratio, with the KDD cup data set and Lipid profile dataset.

A. Aggregation Time Vs Output Ratio

Aggregation time is defined as the entire time taken to aggregate the data set. Aggregation time of proposed approach ought to be under the present methodologies for the better performance improvement.

Table 1 Aggregation Time Vs Output Ratio for KDD Dataset

Output Ratio	Aggregation time (secs)		
	CA MapReducer	CCA MapReducer	CCNA MapReducer
0.2	276	220	160
0.4	415	360	310
0.6	610	560	520
0.8	710	640	610
1	780	710	690

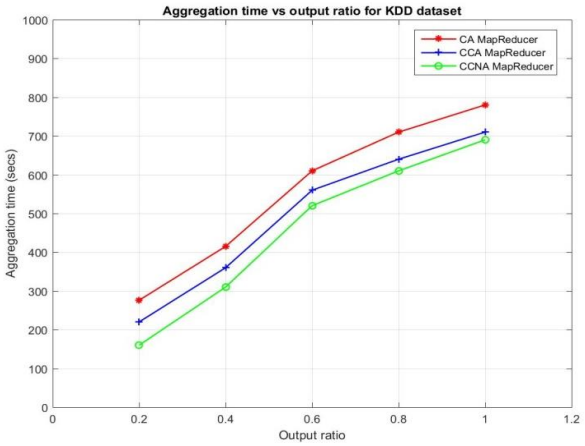

Figure 2 Aggregation Time Vs Output Ratio for KDD Dataset

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

From the figure 2 and Table 1 shows that the aggregation time in the aggregator node. It is proved that the proposed CCNA MapReducer method takes 690 *secs* to aggregate the data but the existing CA MapReducer takes 780 *secs*to aggregate the data and the existing CCA MapReducer takes 710 *secs*to aggregate the data at the point of 1 output ratio. At the point of 0.6 output ratio the proposed method takes 520 *secs* to aggregate the data but the existing CA MapReducer takes 610 *secs*to aggregate the data and the existing CCA MapReducer takes 560 *secs*to aggregate the data for KDD dataset.

Table 2 Aggregation Time Vs Output Ratio for Lipid Profile Dataset

Output Ratio	Aggregation time (secs)		
	CA MapReducer	CCA MapReducer	CCNA MapReducer
0.2	226	170	110
0.4	365	310	260
0.6	560	510	470
0.8	600	590	560
1	730	660	640

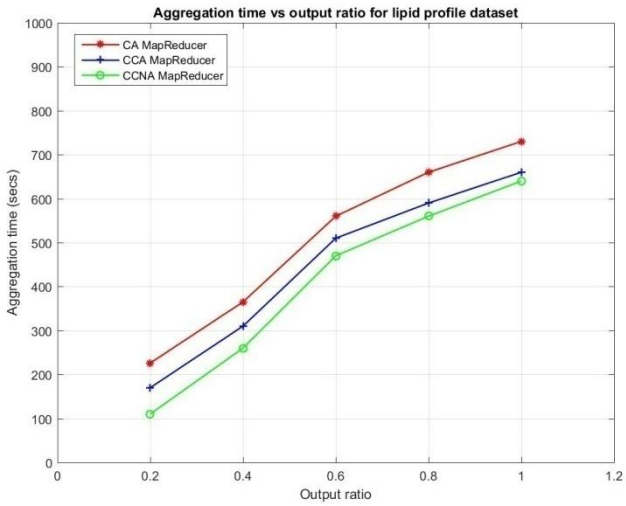

Figure 3 Aggregation Time Vs Output Ratio for Lipid Profile Dataset

From the figure 3 and Table 2 shows that the aggregation time in the aggregator node. It is proved that the proposed CCNA MapReducer method takes 640 *secs* to aggregate the data but the existing CA MapReducer takes 730 *secs*to aggregate the data and the existing CCA MapReducer takes 660 *secs*to aggregate the data at the point of 1 output ratio. At the point of 0.6 output ratio the proposed method takes 470 *secs* to aggregate the data but the existing CA MapReducer takes 560 *secs*to aggregate the data and the existing CCA MapReducer takes 510 *secs*to aggregate the data for lipid profile dataset.

B. Partition Time Vs Output Ratio

Partition time is defined as the entire time taken to partition the data set. Partition time of proposed approach should be less than the existing methodologies for the better performance improvement. In this partition time is calculated against the output ratio.

Table 3 Partition Time Vs Output Ratio for KDD Dataset

Output Ratio	Partition time (secs)		
	CA MapReducer	CCA MapReducer	CCNA MapReducer
0.2	253	197	137
0.4	392	337	287
0.6	587	537	497
0.8	687	617	587
1	757	687	667

From the figure 4 and Table 3 shows that the partition time in the mapper node. It is proved that the proposed CCNA MapReducer method takes 667 *secs* to partition the data but the existing CA MapReducer takes 757 *secs* to partition the data and CCA MapReducer takes 687 *secs* to partition the data at the point of 1 output ratio. At the point of 0.6 output ratio the proposed CCNA MapReducer method takes 497 *secs* to partition the data but the existing CA MapReducer takes 587 *secs* to partition the data and CCA MapReducer takes 537 *secs* to partition the data for KDD dataset.

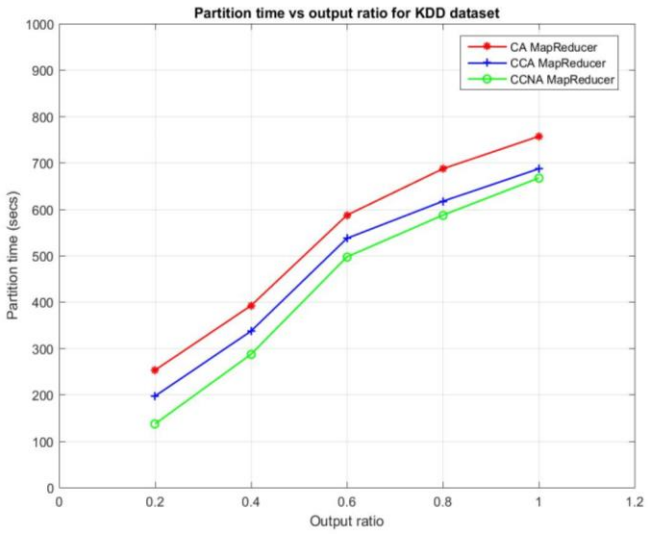

Figure 4 Partition Time Vs Output Ratio for KDD Dataset

From the figure 5 and Table 4 shows that the partition time in the mapper node. It is proved that the proposed CCNA MapReducer method takes 617 *secs* to partition the data but the existing CA MapReducer takes 707 *secs* to partition the data and CCA MapReducer takes 637 *secs* to partition the data at the point of 1 output ratio.

Table 4. Partition Time Vs Output Ratio for lipid profile dataset

Output Ratio	Partition time (secs)		
	CA MapReducer	CCA MapReducer	CCNA MapReducer
0.2	106	147	87
0.4	203	287	237
0.6	342	487	447
0.8	537	567	537
1	707	637	617

At the point of 0.6 output ratio the proposed CCNA MapReducer method takes 447 *secs* to partition the data but the existing CA MapReducer takes 342 *secs* to partition the data and CCA MapReducer takes 487 *secs* to partition the data for lipid profile dataset.

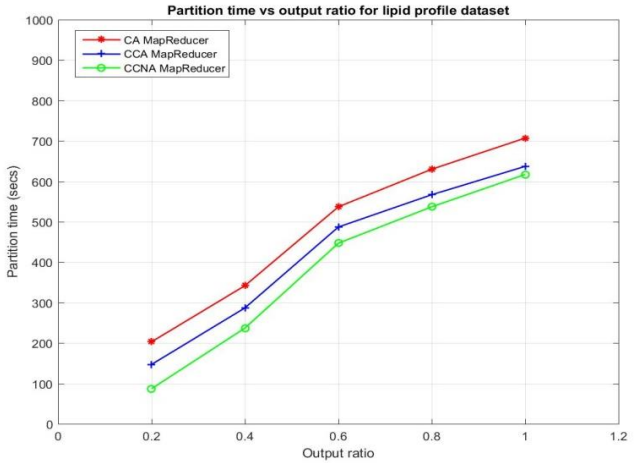

Figure 5 Partition Time Vs Output Ratio for Lipid Profile Dataset

IV. CONCLUSION

In this work the problem of network traffic, computation cost, accuracy and error rate in MapReduce framework is considered and these problems can be overcome by using an aggregator in MapReduce which aggregate the data from mappers, remove duplicate copies of data and repartition the data based on content, computation and network aware of data. Thus the proposed MapReduce framework performs better than the existing MapReduce framework however there is considerable scope for further improvement.

REFERENCES

1. Wang, Y. X., Luo, J. Z., Song, A. B., & Dong, F. Partition-based online aggregation with shared sampling in the cloud. *Journal of Computer Science and Technology*, *28*(6), 989-1011, (2013).
2. Shuijing, H. Big data Analytics: Key technologies and challenges. In International Conference on Robots & Intelligent System, 141-145, 2016.
3. Song, C. Research of association rule algorithm based on data mining. In *Big Data Analysis (ICBDA), 2016 IEEE International Conference on*, 1-4, 2016.
4. Zerhari, B., Lahcen, A. A., & Mouline, S. 'Big data clustering: Algorithms and challenge'. In *Proc. of Int. Conf. on Big Data, Cloud and Applications (BDCA'15), 2015*.
5. Sharma, R., Singh, S. N., & Khatri, S. Medical Data Mining Using Different Classification and Clustering Techniques: A Critical Survey. In *Computational Intelligence & Communication Technology (CICT), 2016 Second International Conference on*, 687-691, 2016.
6. Grolinger, K., Hayes, M., Higashino, W. A., L'Heureux, A., Allison, D. S., & Capretz, M. A. Challenges for mapreduce in big data. In *2014 IEEE World Congress on Services*, 182-189, 2014.
7. Bing, L. I., & Chan, K. C. A paralleled big data algorithm with mapreduce framework for mining twitter data. In *Big Data and Cloud Computing (BdCloud), 2014 IEEE Fourth International Conference on* IEEE, pp. 121-128, 2014
8. Costa, P., Donnelly, A., Rowstron, A., & O'Shea, G. Camdoop: exploiting in-network aggregation for big data applications. In *Presented as part of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12)*, 29-42, 2012.
9. Daltrophe, H., Dolev, S., & Lotker, Z. Data Interpolation: An Efficient Sampling Alternative for Big Data Aggregation. *arXiv preprint arXiv:1210.3171, 2012*.
10. Ke, H., Li, P., Guo, S., & Guo, M. On Traffic-Aware Partition and Aggregation in MapReduce for Big Data Applications. *IEEE Transactions on Parallel and Distributed Systems*, *27*(3), 818-828, 2016.
11. Pandey, N. K., Zhang, K., Weiss, S., Jacobsen, H. A., & Vitenberg, R. Distributed event aggregation for content-based publish/subscribe systems. In *Proceedings of the 8th ACM International Conference on Distributed Event-Based Systems*, 95-106, 2014.
12. Zhang, Y., Chen, S., Wang, Q., & Yu, G. i MapReduce: Incremental MapReduce for Mining Evolving Big Data. *IEEE Transactions on Knowledge and Data Engineering*, *27*(7), 1906-1919, 2015.
13. Khan, N., Husain, M. S., & Beg, M. R. Big Data Classification using Evolutionary Techniques: A Survey. In *Proc. of IEEE International Conference on Engineering and Technology (ICETECH)*, 243-247, 2015.
14. Karthick, N., and X. Agnes Kalarani. "An Improved Method for Handling and Extracting useful Information from Big Data", Indian Journal of Science and Technology, 2015
15. Rajesh, M., Kumar, K. S., Shankar, K., & Ilayaraja, M. Sensitive data security in cloud computing aid of different encryption techniques. *Journal of Advanced Research in Dynamical and Control Systems*, *18*, 367-387.

AUTHOR PROFILE

Dr. K. Sathesh Kumar completed M.C.A., Ph.D. He is presently working as an Assistant Professor in the Department of Computer Science & Information Technology, Kalasalingam University, Krishnankoil, India He has Seven years of experience in teaching and research level and also he published many research Papers in both International and National Journals. His research areas include Data Mining, Image Processing, Computer Networks, Cloud Computing, Software Engineering and Neural Network.