An Intelligent Storage Optimization Technique for Heterogeneous Hadoop Clusters

N. S. Kalyan Chakravarthy[1], N.Sudhakar[2], E. Srinivasa Reddy [3]

[1] University College of Engineering & Technology, Acharya Nagarjuna University.
[2] *Department of Computer Science*, Bapatla Engineering College.
[3] University College of Engineering & Technology, Acharya Nagarjuna University.
[1]suryaet@yahoo.com , [2]suds_nagalla@yahoo.com, [3]edara_67@yahoo.com

***Abstract* - Big data storage and processing can be carried out with the help of Hadoop technology. Hadoop distributed file system store data using chunks, whose size is multiple of 64MB. The default storage space occupied by the raw data is always 200% more, as the default configuration uses three-fold technique. The data stored over HDFS may have different priority, formats and redundancies. The data is stored in standard HDFS without applying either compression or encoding and redundancy. This traditional storage system makes the storage resource underutilized. To improve the usage of storage resources in big data platform an intelligent storage system is required. This paper proposes a Machine learning based intelligent storage system called iYARN, Yet Another Resource Negotiator, to maximize the storage capability. The proposed algorithm improves storage allocation for structured data by 20%, unstructured data by 30%, and semi-structured data by 35%.**

Keywords - ***Hadoop, Disk Schedulers, Cloud computing, Processor scheduling, Throughput, YARN, HDFS, Intelligence, Machine Learning, Interconnections, Storage automation, Task scheduling, Distributed processing***

I. INTRODUCTION

The recent growth in data radically improves the strategies in the business. The data produced by online companies using various tools such as sensors, scientific instruments, social media and mobile phones is rapidly increasing day by day. The traditional database management systems are facing challenges in storing, analysing and processing huge volumes of data. Hadoop, an open source distributed framework has been introduced to address the challenges with the volume of data. Bigdata denotes large amounts of data with three major attributes namely Volume, Variety and Velocity.

In recent times, Hadoop gained its importance as it succeeded in storing, processing and analysing large volumes of data. The primary data storage system utilized by Hadoop application is HDFS (Hadoop Distributed File System). HDFS is designed to handle big data. HDFS splits the data into separate blocks and distributes them among different nodes in a cluster. As HDFS spreads over multiple low cost machines, it indirectly helps to reduce cost at the same time increase reliability. Apache Hadoop uses its core component, YARN (Yet Another Resource Negotiator) to schedule the jobs and to assign resources to various jobs running on Hadoop cluster in Parallel.

One challenge with YARN is that though it best suits with some applications it doesn't fit with other applications since the characteristics of one application varies from the other in terms of I/O intensive, Memory intensive and CPU Intensive. Hadoop accommodate the new hardware very easily. HDFS is designed to store large volumes of data across multiple machines of a large cluster. HDFS uses a special process called replication to achieve fault-tolerance. That is, every block of a file stored in one node is duplicated in two more nodes to attain fault-tolerance. This replication feature makes HDFS reliable because if one node fails, the data can be easily retrieved from other nodes. At the same time, replication made HDFS overloaded by 200% as it is wasting the memory unnecessarily in maintaining duplicates. The challenges with replication can be addressed by employing one of the following techniques:

1. Data Compression
2. RAID
3. Erasure Coding
4. Principal Component Analysis

Data compression enables in the minimization of the space required to store documents and accelerates information exchange over the network. There are several compression techniques such as gzip, bzip2, LZO, Snappy. These techniques assist in compressing the files size by 10% to 15%.Replication is one of the very expensive process. The default option is 3 folds replication in HDFS and it causes 200% overhead in storage space. To optimize the storage and improve the distribution of the data, there exists a need to provide both fault-tolerance and less space. The RAID configuration using RAID Level 5 often provides some level of redundancy and replication, which is comparatively better when compared with the traditional n-fold setup in the Hadoop cluster.

Erasure code enables replication by using lesser number of disks for storage by utilizing parity information spread across the disks. The parity information is helpful to restore the data in the case of disk failure. The environment using erasure code has storage overhead less

than 50%. Principal Component Analysis (PCA) aims to diminish high dimensional datasets to low dimensions. PCA finds the similarities and dissimilarities among the patterns in data. PCA identifies principal components, which can be described as a pair of optimally-weighted variables. PCA analyses the dataset and identifies the random variables and independent variables. Thus, PCA technique assists in minimizing the dimensionality by discarding the identified random variables, which can be retrieved using independent variables if necessary.

II. LITERATURE REVIEW

Ankita Jain *et al*. implemented a new approach, tuning configuration parameter to assess and improve the performance of Hadoop. As the Hadoop performance is generally get affected by parameters related to MapReduce, the proposed approach focuses on minimizing the job execution time by altering the parameters associated with MapReduce. This method succeeded in minimizing the execution time and maximizing the disk usage. The experimental results showed that the Hadoop performance is improved by 38.51%.Damian Reeves *et al*. introduced an efficient alternative to HDFS namely Quant cast File System (QFS). C++ was used to implement QFS. It was designed in such a way that it is companionable with Hadoop provide several additional features over Hadoop. QFS uses erasure coding in the place of replication, which result in saving the disk space by 50%. [2].

Tyler Harter *et al*. had introduced that a multilayer study of the Facebook Messages stack, which is based on HBase and HDFS. Messages represent a new HDFS workload: whereas HDFS was built to store very large files and receive mostly sequential I/O, 90% of files are smaller than 15MB and I/O is highly random. It has been found that it is too large to easily fit in RAM and cold data is too large to easily fit in flash; however, cost simulations show that adding a small flash tier improves performance more than equivalent spending on RAM or disks. HBase's layered design offers simplicity. Finally, although Messages are read-dominated, several features of the stack amplify write I/O, causing writes to dominate disk I/O. Merging multiple HBase logs on a dedicated disk reduces logging latencies by a factor of 6. [3].

Mark Yong *et al*. presented about the large clusters that are held by Amazon, Facebook and Yahoo that are organized by Hadoop-MapReduce. Hadoop uses default FIFO in scheduling jobs and resource allocation. This paper proposed two efficient scheduling schemes to reduce contention in terms of for CPU and IO. These scheduler schemes can also be used to predict the resources required in the future based on the present usage. [4]. Shekhar Gupta, *et al*. implemented the throughput scheduler and described that there is a de-facto standard for big data analytics applications. This technique was applied on sample data and demonstrated that throughput scheduler reduced the total job execution time by 20% and 40% when compared to FIFO scheduler. The experimental results also proved that Throughput Scheduler minimized the average mapping time by 33% over other schedulers. [5].

Jungi Jeong *et al* had presented that short jobs need to be concentrated as they show high impact on user experience and system productivity of Hadoop. It has been identified that high latency is caused when short jobs are accidentally blocked with disk writes. The investigation results proved that the execution time of 99 and average percentile short jobs is reduced by 22% and 40% correspondingly [6]. Swapnil Patil *et al* focused on HDFS implementation issues in terms of disk bandwidth and how pipelined replication is beneficial. The author explained that Data intensive applications would generally come under the categories of either high-performance computing (HPC) or computing styles. Application performance plays the key role in both the cases. So, this paper proposed PVFS integrated into Hadoop. The paper also presents the comparative results with the proposed technique against HDFS [7].

Bryon Neitzel, *et al* presented that High performance Computing systems are producing large volumes of data every day. The proposed VAS framework comprises of three component techniques, virtualized analytics shipping with fast network and disk I/O to establish isolation amid I/O services and MapReduce analytics, pipelined intermediate data merging and stripe-aligned data distribution and task scheduling to optimize MapReduce to avoid explicit shuffling. The experimental results showed that VAS delivers fast and virtualized MapReduce programs [8]. Alexander Rasmussen, *et al*. explained that MapReduce programming model is being heavily used to process large data collections. The performance of MapReduce jobs can be enhanced by minimizing the number of I/O-bounds as MapReduce jobs are typically I/O based. So, this paper proposed various techniques to minimize the I/O bounds there by increasing the performance of MapReduce [9].

Antony Rowstron, *et al* presented the comparative and research study done over different clusters. Recent studies identified that, some of the analytical frameworks cannot even analyze 100GB of input where as popular infrastructures like MapReduce or Hadoop were designed in such a way that they can process over Big data. An evaluation across 11 representative Hadoop jobs shows scale-up to be competitive in all cases and significantly better than scale-out in some cases. [10]. Mr. Jinshuang Yan, *et al*. introduced certain techniques to enhance MapReduce jobs' execution performance. These methods are basically utilized to reduce the time cost of a job by optimizing its setup and clean-up tasks, by using instant message communication mechanism to exchange notifications among Task Tracker and Job Trackers

instead of heartbeat-based communication mechanism, and by using the pull-model task assignment mechanism in the place of a push-model. Investigational outputs proved that this technique succeeded in enhancing the job execution performance by 23% [11].

Mr. Mohd, *et al.* presented various challenges and issues with big data processing over Hadoop cluster. This paper proposed that even though Hadoop has proven to be the best technology to handle Big data and confirmed that there are some challenges with HDFS and MapReduce, which are still need to be addressed by the researchers. This paper mainly focused on the parameters that are majorly contribute to the overall performance of Hadoop. This paper also discussed how the HDFS scheduling and job processing can be enhanced [12]. Mr Douglas, *et al,* introduced a novel acceleration framework, Hadoop-A and presented the challenges faced by the existing Hadoop implementation such as the serialization barrier that delays the reduce phase and repetitive merges associated with disk access. Hadoop-A, an acceleration framework implemented new plugin components written in C++ for fast data movement, overcoming its existing issues. The investigational results thus generated proved that Hadoop-A doubles the data processing throughput of Hadoop, and reduces CPU utilization by more than 36**%** [13].

III. PROPOSED I-YARN

To address the challenges with the replication factor, this article suggested various techniques such as RAID, Eraser Coding (EC) and Principle Component Analysis (PCA). Among these techniques, some techniques may suit for structured data and some may suit unstructured data and semi structured data sets. So, this paper proposed iYARN (intelligent YARN) that uses XGBOOST, a machine learning algorithm by classifying the input data set and adopt a suitable technique to reduce the input data set size thereby optimizing the storage in HDFS. The proposed algorithm classifies the input data into three categories namely, Structured, Unstructured and Semi-structured. For structured data types, it uses Principal Component Analysis, for unstructured data enforces use of Eraser Coding and for semi-structured data Compression techniques were to manage the HDFS storage effectively and efficiently. The storage resources represented as S_a and the storage required by a client I is denoted by S_{a1}. The storage demand of the client is represented by S_{d1}. The demanded resource of a client is less than the available resource, the resource allocation is done without any prediction of resources. Otherwise the weighted arithmetic mean of the resources will be calculated to understand the resources usage pattern. XGBoost stands for eXtreme Gradient Boosting, The name xgboost, though, actually refers to the engineering goal to push the limit of computations resources for boosted tree algorithms.

Boosting is an ensemble technique where new models are added to correct the errors made by existing models. Models are added sequentially until no further improvements can be made. A popular example is the AdaBoost algorithm that weigh data points that are hard to predict. Gradient boosting is an approach where new models are created to predict the residuals or errors of prior models and then are added together to make the final prediction. It is called gradient boosting because it uses a gradient descent algorithm to minimize the loss when adding new models. The storage will be allocated only when the predicted storage is less than the required storage. Otherwise, the client wait continuously to get the required storage.

S_a: Available Storage in Cluster.
$S_{a1} = (S_{a1} \ldots S_{an})$ Storage Allocation.
$\#S_{ai}$ denotes storage allocation for client i.
$S_d = (S_{d1} \ldots S_{dn})$ storage resources demanded by tenants.

$\#S_{di}$ denotes resources demanded by client i
if S_{di} less than S_{ai} then
    U_a < - R_{di} #Allocate demanded storage resources
    U_{pi}< - $S_{ai} - S_{di}$ #Resources Preempted &
                Update heuristic table.
else
    W < - $\sum S_i * S_{di} / \sum S_i$
    E_{sd} < - XGBRegression(W)

while : execute pending tasks
    if $E_{sd} <= S_a$ then U_{ai}<- $U_p + S_{di}$
    else
    Wait until there is a released resource
        S_i from client i
end
Update heuristic table for client i.

IV. EXPERIMENTAL EVALUATION

Hadoop Cluster was setup with 6 nodes. In the cluster environment, each node has Intel dual core processor, 2GB RAM and 50GB HDD. All the cluster nodes are installed with Linux/CentOS 6.8 operating system, Java 1.8 and Hadoop 2.7.3 to establish hierarchical Hadoop Cluster. One node is dedicated to run Name Node Service and Resource Manager daemon service to communicate with slave nodes. The proposed iYARN framework is adopted to optimize the resource usage. In iYARN, XGBoost (eXtreme Gradient Boosting) trained with the following dataset. The classification and clustering of the data has based on the file parameters.

TABLE I. TRAINING DATA SET

File Size in MB		Demand		Allocation		Preemption				
Tenants	**Time Interval**	**New**	**Total**	**Current**	**Total**	**Current**	**Total**	**File Type**	**File Size in MB**	**Extension**
Client A	t1	20	20	20	20	30	30	Structured	20	csv
Client B	t1	100	100	80	80	-30	-30	Unstructured	100	txt
Client C	t1	40	40	40	40	10	10	Semi-Structured	40	xml
Client D	t1	80	80	80	80	-30	-30	Structured	80	csv
Client E	t1	30	30	30	30	20	20	Structured	30	csv
Client A	t2	40	60	40	60	10	40	Structured	40	csv
Client B	t2	60	140	50	130	0	-30	Structured	60	csv
Client C	t2	20	60	20	60	30	40	Semi-Structured	20	xml
Client D	t2	70	150	70	150	-20	-50	Semi-Structured	70	xml
Client E	t2	10	40	10	40	40	60	Semi-Structured	10	xml
Client A	t3	70	130	70	130	-20	20	Semi-Structured	70	xml
Client B	t3	60	190	40	170	10	-20	Unstructured	60	txt
Client C	t3	40	100	40	100	10	50	Unstructured	40	txt
Client D	t3	60	210	40	190	10	-40	Unstructured	60	txt
Client E	t3	70	110	70	110	-20	40	Unstructured	70	txt
Client A	t4	80	210	80	210	-30	-10	Unstructured	80	txt
Client B	t4	20	190	20	190	30	10	Unstructured	20	txt
Client C	t4	80	180	80	180	-30	20	Structured	80	csv
Client D	t4	10	200	10	200	40	0	Structured	10	csv
Client E	t4	60	170	60	170	-10	30	Structured	60	csv
Client A	t5	30	240	30	240	20	10	Structured	30	csv
Client B	t5	50	240	50	240	0	10	Structured	50	csv
Client C	t5	60	240	60	240	-10	10	Unstructured	60	txt
Client D	t5	40	240	40	240	10	10	Unstructured	40	txt
Client E	t5	70	240	70	240	-20	10	Unstructured	70	txt

A. Input Dataset:

HTNO	CODE	RT41051	RT41052	RT41053	RT41054	RT4105B	RT4105L	RT4105M	RT4105N	RT4105O	Gtotal	Credits	Failed
14221A0501	22	60	52	53	68	50	65	66	73	73	70	23	0
14221A0502	22	53	56	69	63	68	62	73	71	70	73.13	23	0
14221A0503	22	56	45	34	57	24	70	65	70	67	61	17	2
14221A0504	22	57	78	68	59	79	67	73	74	72	78.38	23	0
14221A0505	22	57	58	60	60	71	70	67	74	72	73.63	23	0

Fig 1. Structured data set sample

TABLE II. CPU UTILIZATION IN YARN

File Size	File Type	CPU % user	CPU % system	CPU % i/o wait
12MB	Structured	1.296	17.199	3.285
12MB	Structured	3.888	51.597	9.855
22MB	Unstructured	2.17	22.051	5.594
22MB	Unstructured	6.51	66.153	16.782
18MB	Semi Structured	2.346	19.439	4.223
18MB	Semi Structured	7.038	58.317	12.669

In Table II User CPU Time shows the percentage of CPU utilization that occurred while executing at the user level (application). System Time shows the percentage of CPU utilization that occurred while executing at the system level (kernel). IO Wait Time denote the CPU or CPUs were idle during which the system had an outstanding disk I / O request.

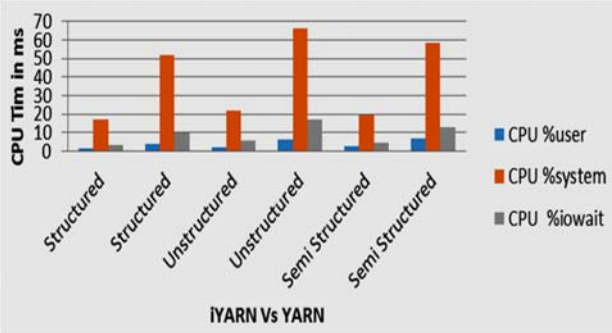

Fig 2. iYARN CPU usage comparison

Table III shows the Memory usage in YARN and iYARN frameworks. In table 4.3 Blk_read/s indicates the amount of data read from the device expressed in a number of blocks per second. Blocks are equivalent to sectors with kernels 2.4 and later and thus have a size of 512 bytes. With older kernels, a block is of indeterminate size. Blk_ wrtn/s indicates the amount of data written to the device expressed in a number of blocks per second. Blk_read denotes the total number of blocks read. Blk_wrtn represents the total number of blocks written.

TABLE III. MEMORY UTILIZATION IN YARN

File Size	File Type	Memory Usage	Remarks
12MB	Structured	1455	iYARN
12MB	Structured	4365	YARN
22MB	Unstructured	2454	iYARN
22MB	Unstructured	7362	YARN
18MB	Semi Structured	1925	iYARN
18MB	Semi Structured	5775	YARN

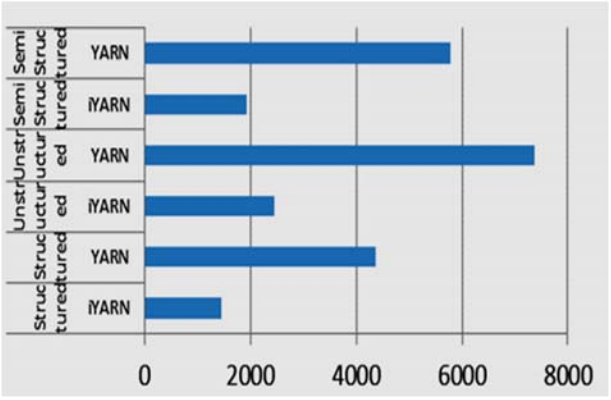

Fig 3. Memory usage comparison

TABLE 4. DISK IO UTILIZATION IN YARN

File Size	File Type	Disk IO TPS	Disk IO Reads in Sec	Disk IO Writes in Sec
12MB	Structured	7.72	491.77	335.04
12MB	Structured	23.16	1475.31	1005.12
22MB	Unstructured	9.57	651.71	535.13
22MB	Unstructured	28.71	1955.13	1605.39
18MB	Semi Structured	8.24	537.45	415.54
18MB	Semi Structured	24.72	1612.35	1246.62

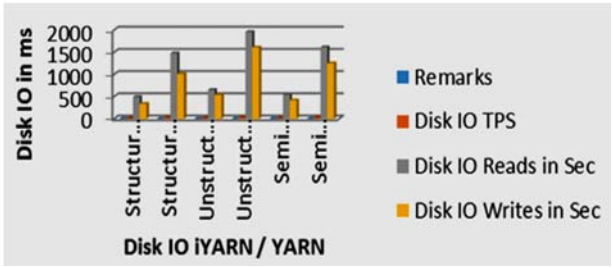

Fig 4. Disk IO Usage comparison

V. CONCLUSION

This work confirms that iYARN scheduler helps to improve storage utilization in a heterogeneous environment. The existing scheduling policy used in YARN for disk utilization did not consider size, format, and other metrics of the data. iYARN is a pluggable scheduler in Hadoop platform to optimize the storage based on the input data type, size, priority and other metrics. The data stored in HDFS using iYARN will reduce its storage by 20% in the case of structured data with the help of multi-dimension reduction techniques and 30% of the storage space is reduced for unstructured data type. Nearly 35% of the storage space is reduced for semi-structured data when compared to the existing Hadoop storage policies and associated algorithms.

REFERENCES

[1] Ciavotta, M., Ardagna, D., & Gibilisco, G. P. (2017). A mixed integer linear programming optimization approach for multi-cloud capacity allocation. Journal of Systems and Software, 123, 64–78.

[2] Ciavotta, M., Ardagna, D., & Koziolek, A. (2016). Palladio Optimization Suite: QoS optimization for component-based Cloud applications. In Proceedings of the 9th EAI International Conference on Performance Evaluation Methodologies and Tools (pp. 170–171).

[3] Ciavotta, M., Dotoli, M., Fanti, M. P., Hammadi, S., Koubaa, S., & Meloni, C. (2006). Genetic algorithms for the setup coordination in consecutive stages of a production chain. In International Workshop on Logistics & Transportation (pp. 218–223).

[4] Ciavotta, M., Gianniti, E., & Ardagna, D. (2016). D-SPACE4Cloud: a design tool for big data applications. In Algorithms and Architectures for Parallel Processing (pp. 614–

629). Springer International Publishing.

[5] Ciavotta, M., Meloni, C., & Pranzo, M. (2009). Scheduling dispensing and counting in secondary pharmaceutical manufacturing. AIChE Journal, 55(5), 1161–1170.

[6] Ciavotta, M., Meloni, C., & Pranzo, M. (2016). Speeding up a rollout algorithm for complex parallel machine scheduling. International Journal of Production Research, 54(16), 4993–5009.

[7] Condie, T., Conway, N., Alvaro, P., Hellerstein, J. M., Elmeleegy, K., & Sears, R. (2010). MapReduce online. In Nsdi (Vol. 10, p. 20).

[8] Curino, C., Difallah, D. E., Douglas, C., Krishnan, S., Ramakrishnan, R., & Rao, S. (2014). Reservation-based scheduling: If you're late don't blame us! In Proceedings of the ACM Symposium on Cloud Computing (pp. 1–14).

[9] Dao, T. C., & Chiba, S. (2016). HPC-Reuse: Efficient Process Creation for Running MPI and Hadoop MapReduce on Supercomputers. Proceedings - 2016 16th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing, CCGrid 2016, 342–345. https://doi.org/10.1109/CCGrid.2016.72

[10] Delgado, P., Didona, D., Dinu, F., & Zwaenepoel, W. (2016). Job-aware Scheduling in Eagle: Divide and Stick to Your Probes. In SoCC 2016 - Proceedings of the Seventh ACM Symposium on Cloud Computing (pp. 497–509).

[11] Delgado, P., Dinu, F., Kermarrec, A.-M., & Zwaenepoel, W. (2015). Hawk: Hybrid Datacenter Scheduling. In Usenix ATC 2015.

[12] Deng, W., Liu, F., Jin, H., Liao, X., Liu, H., & Chen, L. (2012). Lifetime or energy: Consolidating servers with reliability control in virtualized cloud datacenters. CloudCom 2012 - Proceedings: 2012 4th IEEE International Conference on Cloud Computing Technology and Science, 18–25. https://doi.org/10.1109/CloudCom.2012.6427550.

[13] Dinu, F., & Ng, T. S. E. (2014). RCMP: Enabling Efficient Recomputation Based Failure Resilience for Big Data Analytics. In IPDPS 2014 - 28th IEEE International Parallel & Distributed Processing Symposium.