ISSN (Print) : 0974-6846
Indian Journal of Science and Technology, *Vol 8(29), DOI: 10.17485/ijst/2015/v8i29/84650, November 2015* ISSN (Online) : 0974-5645

Comparison of Machine Learning Algorithm on Map Reduction for Performance Improvement in Big Data

Ananthi Sheshasayee[1*] and J. V. N. Lakshmi[2]

[1]PG and Department of Computer Science and Research, Quaid – E – Millath Government. College for Women, Chennai, - 600002, Tamil Nadu, India; ananthi.research@gmail.com
[2]Department of Research in Computer Science, SCSVMV University, Kanchipuram - 631561, Tamil Nadu, India; jlakshmi.research@gmail.com

Abstract

Background: As massive data acquisition and storage becomes increasingly affordable, a wide variety of enterprises are engaged in sophisticated data analysis. The amount of digital information which is majorly unstructured produced is exceeding day by day. **Method:** MapReduce programming method is easily applicable to many different learning algorithms. Machine Learning is at the core of data analysis. Traditional machine learning algorithms speed up at a time to fit the statistical query model on multicore computers. Data Sharing is avoided by Hadoop whereas Machine learning Algorithm needs data to be stored in single place. The method compares the machine learning algorithms on MapReduce paradigm for evaluating speed. **Findings:** MapReduce programming model enables easy development of scalable parallel applications to process large clusters of data. Hadoop Distributed File System runs the MapReduce jobs which influence the performance significantly while handling huge data set stored on different nodes of a multi node cluster. This paper analyses on developing machine learning algorithms on Hadoop to process large clusters of data. Analyzing logistic regression algorithms on MapReduce for evaluating the performance to speed up processing by developing a cost model. The attributes of the system are evaluated for improving time efficiency. The objective is to provide ad hoc performance for MapReduce programs which run on large data sets. **Improvements:** A method for optimizing job assignment on machine learning is implemented in order to minimize the total execution time. This feature improves the productivity of MapReduce users to optimize performance efficiency.

Keywords: Big Data, HADOOP, HDFS, Machine Learning, Map Reduction

1. Introduction

Big data analytics is the use of advanced analytic techniques against very large, diverse data sets that include different types such as structured/unstructured and streaming/batch and different sizes from terabytes to zettabytes[1]. Analyzing big data allows analysts, researchers, and business users to make better and faster decisions using data that was previously inaccessible or unusable. Using advanced analytics techniques such as text analytics, machine learning, predictive analytics, data mining, statistics, and natural language processing, businesses can analyze previously untapped data sources independently or together with their existing enterprise data to gain new insights resulting in significantly better and faster decisions[2]. High-performance analytics is necessary to process that much data in order to extract useful knowledge for decision making. The need for storage system is increasing rapidly in this world for handling massive size of data analysis[3].

* *Author for correspondence*

Machine learning is too big data as human learning is to life experience. By interpolating and extrapolating from past experiences to deal with unfamiliar situations. Machine learning with big data will duplicate this behavior, at massive scales. Big Data needs big compute for which Hadoop is a solution. Hadoop allows easier data processing across a large cluster of low price commodity servers[4]. But Hadoop MapReduce is a batch-oriented system, and doesn't lend itself well towards interactive applications; real-time operations like stream processing; and other, more sophisticated computations[5].

For predictive analytics, this needs an infrastructure that's much more responsive to human-scale interactivity. A lot of iteration needs to occur on a continual basis for the system to get smart, for the machine to "learn" - explore the data, visualize it, build a model and repeat the process. This feature of machine learning, intends to incorporate with big data analytics.

The importance of performance tuning on powerful programming model MapReduce is widely recognized[6]. Tuning jobs on MapReduce rapidly use no more resources than necessary[7,8]. First step defining relevant MapReduce job, second step application of machine learning algorithm finally analyzing the critical value parameters.

The paper is organized as section 2 introducing MapReduce and HADOOP framework. Section 3 various machine learning algorithms such as EM Algorithm, K-Means and logistic regression. Then in section 4 phases of MapReduce by applying cost model. Section 5 discussing evaluation and results. Finally, section 6 concludes.

2. MapReduce and Hadoop

MapReduce usually splits the input data sets into splits which are processed by map tasks parallel. The output of Map phase is assigned as input to reduce phase. Typically, both the input and output are stored in distributed file system[9].

MapReduce is the heart of Hadoop. It is this programming paradigm that allows for massive scalability across hundreds or thousands of servers in a Hadoop cluster. The MapReduce concept is fairly simple to understand for those who are familiar with clustered scale-out data processing solutions. The MapReduce framework and Hadoop distributed file system are running on the same set of nodes. By arranging the framework in these ways result in high bandwidth across the cluster it implements client sever on master slave architecture where master node is job tracker of Hadoop and slave node is the task tracker which handles the map tasks[10].

Hadoop launches MapReduce job by first splitting the input data set into even sized blocks. Each data block is then scheduled to one task tracker node and processed by a map task[11]. The task tracker receives a heart beat protocol. The task tracker notifies the job tracker when it is idle. The schedules assigns new task which is local data block. After map ( ) function finishes its task pairs of intermediate key/values are launched over reduce tasks to produce final results.

The Figure 1 describes MapReduce framework implemented on HDFS architecture. Data is initially

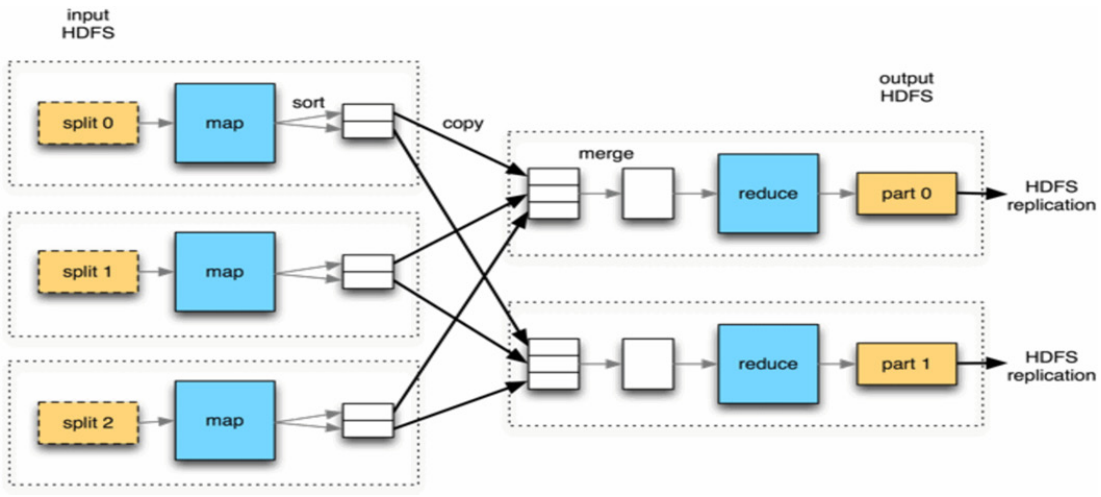

Figure 1. MapReduce in HDFS architecture source.

distributed as splits and each split is assigned to map phase. Then by sorting and shuffling the data generated is given as input to reduce phase. Finally produced output is replicated on HDFS[12].

3. Machine Learning Algorithms

Machine Learning has enormous applications in many fields as it is extensionally used to parallelize large data sets[13]. A technique for parallel programming like MapReduce is implemented with machine learning algorithms. MapReduce corresponds to iteration of the algorithm[14,15]. Map function in the next iteration requires the results of the reduce function is the previous iteration. A variety of learning algorithms, including logistic repression, expectation-maximization in a Gaussian mixture model and K-means satisfy these characterizations.

3.1 EM Algorithm

Clustering can be cast as a statistical inference problem like linear regression, in which we find the best model to fit our data. Here, the model is a *mixture model*, where there are several different mixture components with different parameters producing data points[16]. The data-generating process in a mixture-of-Gaussian model is as follows: Repeat this N times independently, where N is the number of data points.

Randomly select a mixture component m with probability $\pi(m)$ Select a point from the probability distribution of this component, using probability distribution $P(X|m) = N(X|\mu m, \Lambda m)$ where μ and Λ are means and covariances.

3.2 K - Means

K-Means uses an iterative algorithm that minimizes the sum of distances from each object to its cluster centroid, over all clusters. This algorithm moves objects between clusters until the sum cannot be decreased further. The result is a set of clusters that are as compact and well-separated as possible[17].

Given a set of observations $(x_1, x_2,\ldots, x_n)$, where each observation is a *d*-dimensional real vector, K-Means clustering aims to partition the *n* observations into *k* (≤ *n*) sets $S = \{S_1, S_2,\ldots, SK\}$ so as to minimize the Within-Cluster Sum of Squares (WCSS). In other words, its objective is to find: $\arg min\sum_{i=1}^{k}\sum_{x\in si}\|x-\mu i\|^2$ where μ_i is the mean of points in S_i. The most common algorithm uses an iterative refinement technique. Due to its ubiquity it is often called the *k*-means algorithm; it is also referred to as a Lloyd's algorithm.

- Assignment step: Assign each observation to the cluster whose mean yields the least Within-Cluster Sum of Squares (WCSS).

$$s_i^{(t)} = \left\{x_p : \left\|x_p - m_i^{(t)}\right\|2 \le \left\|x_p - m_i^{(t)}\right\|2\forall j;1 \le j \le k\right\}$$

- Update step: Calculate the new means to be the centroids of the observations in the new clusters.

$$m_i^{(t+1)} = \frac{1}{s_i^{(t)}}\sum_{x_i \in s_i^{(t)}} x_j$$

3.3 Logistic Regression

In this section, we show an example of logistic regression, which is a binary classifier that can be used in spam filtering. Let $x^{(i)}$ be the i^{th} training vector (email), $y^{(i)}$ be the i^{th} training label (spam or legitimate), q be the model parameter vector to be learned, and $h_\theta(x) = 1/(1 + \exp(-\theta^T x))$ be the hypothesis. Learning is done by fitting (to the training data using maximum likelihood estimation. Gradient ascent derives the update rule (1), which requires iteration until convergence. The 'm' denotes the number of training vectors and 'a' denotes the learning rate[18].

$$\theta(n) := \left((n+1) + \alpha\sum_{i=1}^{m}(y^{(i)} - h\theta(x^{(i)}))x^{(i)}\right)$$

The Map function calculates α (y (I) - h ((x (I))) x (I) for each training vector. Note that h_θ (x) requires the model parameter θ(n-1) which was calculated in the previous iteration.

The Reduce function sums up all the vectors, each Map function emits. Before the Map function in the next iteration starts, the result of the Reduce function needs to be added. In Hadoop, this operation can be done in a function called Setup, which is executed once before the map phase starts in each node.

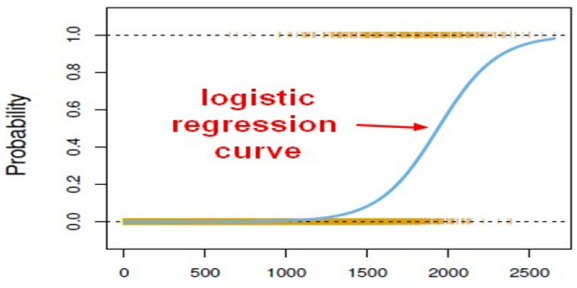

Figure 2. Logistic regression in machine learning source[19].

The above Figure 2 illustrates the curve of logistic regression. On X-axis is the time in seconds and on Y-axis shows the probability curve with increasing limits of 0.2.

4. Map Reduce Phases

An efficient method helps to configure Map-Reduce performance that is defined respectively.

Feature Declaration; we define the optimization problem of a cluster. Given C jobs using different parameters and M node clusters, we find the job assignment that minimizes total execution time. Assume that the job assignments are obtained by equally partitioning M nodes into G groups and assigning each job to one of the groups. The optimization problem turns into a problem to find the optimal number of nodes in a group. This is because the best assignment that minimizes total execution time is always to equally assign the jobs into G groups. The total execution time is bounded by the groups that process [c/g] Map-Reduce jobs. Thus, solving the optimization problem helps to find the optimal number of nodes in a group. Finally, we assume that the cluster consists of homogeneous nodes and that each job is fully parallelized within the group; the Map and Reduce functions are executed in [M/G] nodes in parallel.

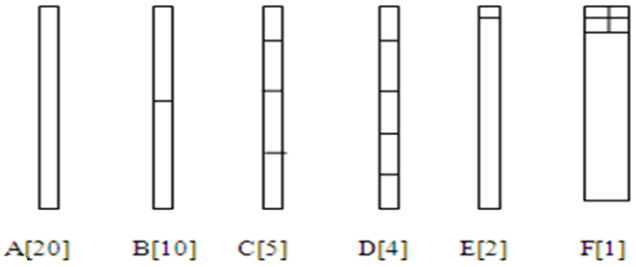

Figure 3. Example of partitioning patterns divided into twenty nodes cluster source.

The Figure 3 shows various partitions. Pattern A takes 20 nodes into a single cluster. Pattern B takes 10 nodes into one single cluster then pattern C is been categorized to 4 clusters handling 5 nodes in each. Pattern D is divided into 5 clusters each with 4 nodes. Pattern E and F is divided into 2 and 1 respectively.

In-order execution needs to load data over and over. The job integration technique integrates jobs and eliminates the redundant parts by interchanging the parameter and data loops as shown in the Figure 3.

Table 1. Data, size, no of jobs and amount of computation for each node source

S no		A	B	C	D	E	F
1	Data Size (GB)	2	4	8	10	20	40
2	No of Jobs	40	20	10	8	4	2
3	Amount of Computation	80	80	80	80	80	80

Feature Selection (Table 1); there are a large number of other parameters whose values have to be specified before the job can be run in a Map-Reduce framework like Hadoop. The settings of these parameters control various aspects of job behavior during execution such as memory allocation and usage, concurrency, I/O optimization, and network bandwidth.

Table 2. Parameters in HADOOP source

Parameters	Description
Mapred.map.tasks	The suggested number of map tasks for the job
Mapred.min.split.size	The minimum size of a split
Dbf.block.size	The file system block size in bytes of the input file
Inputformat.is Splitable	Whether this file may be split

In the Table 2 the various parameters to describe the Map task and reduce task are listed.

4.1 Map Phase

A cost model is constructed for the Map phase of multiple jobs. First, observations of the Map phase of the various partitioning patterns and determine the execution bottleneck. Then, stating a cost model for the Map phase for reading the data from the local disk as batch jobs. Furthermore expand the cost model for handling memory-based execution.

4.2 Reduce Phase

The extended cost model for the Reduce phase of multiple jobs and also included the cost of the Shuffle phase in it. Let RC be a cost model for the Reduce phase of a single job. Each group in a cluster executes the Reduce phase CP/M times.

5. Evaluation

In this section the first task is runtime of MapReduce. Second task is logistic regression using various parameters. Third task is prediction of best partitioning using optimization method.

5.1 Runtime of MapReduce

The Map-Reduce program is invoked when each node downloads its data from Hadoop Distributed File System (HDFS), stores the data in a local disk or memory, and starts execution[20]. The computation of results from two buffers is required for further data processing. While the runtime loads the next data element into one of the buffers in the background, the map function is being applied to the other buffer in the foreground.

5.2 Required Settings

The block size of the file system is set to the default value of 64 MB. The replication level is set to 3. The map task tokenize each line in to words whereas reduce task counts the occurrence of each word[21]. The parameters denotes label. Each job learns the binary classifier, which classifies the label by the given parameter.

6. Results

The results of optimization method which lists the values of the variables executing MapReduce jobs.

Table 3. Stable values of variables

Methods	Description
Cluster	M = 20
Regression analysis With Bayesian network	a1, a2, a3, a4
Job parameters	D = 32(GB)
Variables	P = 10

The Table 3 explains about the stable values for achieving experimental results such as cluster nodes m = 20, variables for Bayesian Network, required memory and pattern categorized in to A- F.

The horizontal axis represents the number of nodes in a group (0-20 nodes), which corresponds to the partitions A-F. The vertical axis represents the average execution time for iteration. The lines represent the time for Map, Reduce and cost.

From the Figure 4 if 20 nodes are involved the time consumed by the Map was more when the nodes are less in count. As they are increasing evenly the execution time gradually decreased to a greater extent. By considering the reduce phase the execution time has a minor changes when compared to map phase. Finally the cost model has a very little effect.

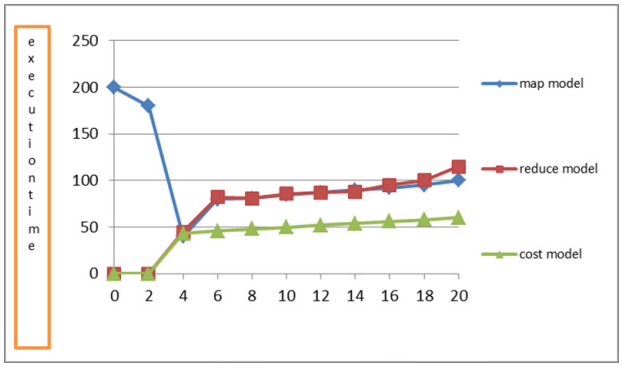

Figure 4. Results of logistic regression labels 1-20 A-F patterns.

From the above Figure 5 if the partitions is10 node clusters in one block then the cost is high. Map phase with 10 nodes has minute differences in execution time when there is an increase in nodes. Reduce model has minor differences in execution time. Optimal assignment reduces execution time by 77 and 25 % compared to the worst assignment respectively. From this optimal pattern can reduce execution time. As the no of nodes are being partitioned into smaller blocks the cost is increasing. As the no of partitions increases cost will become a bottleneck. The analyzed results in this way, if the nodes are increased to 40 partitions the cost is increased to be more than 150 and if the nodes are 80 the cost increases to 200.

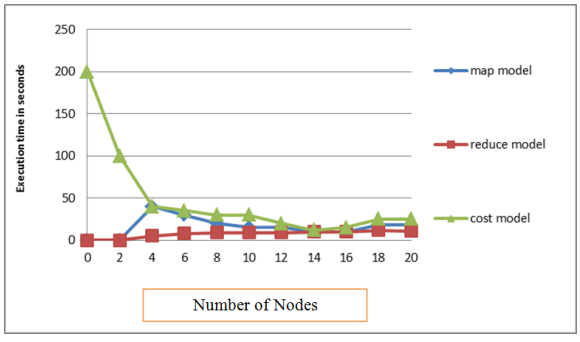

Figure 5. Results of logistic regression labels 1-10 A-F patterns.

7. Conclusion

A method for optimizing the job assignment on machine learning is implemented in order to minimize the total execution time. The setting of job configuration parameters for map - reduces programs focus on Hadoop[22]. The objective is to provide ad hoc performance for MapReduce programs which run on large data sets. This feature improves the productivity of MapReduce users to optimize programs and data being processed. Development of execution cost model to predict execution time and minimize the cost. Implementing MapReduce based on MPI and logistic regression. The results revealed that the optimal assignment reduced execution time by a maximum of 77% compared to worst assignment.

8. References

1. For big data analytics there's no such thing as too big by: 4syth.com emerging big data thought leaders. 2012 Mar.
2. Lu X, Wang B, Zha L, Xu Z. Can MPI benefit Hadoop and Map-Reduce applications? Proceedings of Fortieth IEEE International Conference Parallel Processing Workshops (ICPPW'11); Taipei City. 2011. p. 371-9.
3. Huston L, Satya Narayana M. Storage architecture for early discard in interactive search. FAST Conference Proceedings; 2004.
4. Yang HC, Dasdan A, Hsiao RL, Parker DS. Map-Reduce-Merge: Simplified relational data Processing on large clusters. Proc ACM SIGMOD Int'l Conf on Management of Data (SIGMOD'07); 2007. p. 1029–40.
5. Reily O. Big data now. Media. 2011. p. 17–75.
6. Gangm HH, Luo L. Star fish: A self tuning system for Big Data Analytics. USA: CIDR'11; 2011.
7. Gates AF. Building a high level dataflow system on top of MapReduce: The pig experience. 2009; 2(2):1414–25.
8. Jeff D, Sanjay G. MapReduce: Simplified data processing on large clusters. 2008. p. 107–13.
9. Sheshasayee A, Lakshmi JVN. A study on hadoop architecture for big Data analytics. IJATES Journal. 2014; 2(Spl issue 1):744–8.
10. Abadi K, Rasin DJ. Hadoop DB: An architectural hybrid of MapReduce and DBMS technologies for analytical workloads. 2009; 2(1):922–33.
11. Thusoo A. Sharma DSJ. HIVE: A warehousing solution over a MapReduce framework. VLDB'09. 2009; 2(2):1626–9.
12. Sravanthi A, et al. Building machine learning algorithms on Hadoop for Big Data. IJET Journal. 2013 Feb; 3(2):143–7.
13. Pavlo A, et al. A comparison of approaches to large-scale data analysis. Proc ACM SIGMOD; 2009. p. 165–78.
14. Chu CT, Kim SK, Yi-An L, Ng AY, Olukotun K. Map-Reduce for machine learning on multicore. Proc Int'l Conf NIPS; USA. 2007. p. 281–8.
15. Ekanayake J, Pallickara S, Fox G. Map-Reduce for data intensive scientific analyses. Proc IEEE Int'l Conf on eScience; Indian Polis, IN. 2008. p. 277–84.
16. Barroso LA, Dean J. Web search for a planet: The Google cluster architecture. IEEE Micro. 2003 Apr; 23(2):22–8.
17. Sheshasayee A, Lakshmi JVN. An analysis on machine learning algorithms implemented on Hadoop MapReduce. ICCCMIT International Con; 2014.
18. Walisa, et al. An adaptive ML on MapReduce for improving performance of large scale data analysis on EC2. Bangkok: 11th International Conference on ICT and Knowledge Engineering IEEE; 2013. p. 1–7.
19. Sheshasayee A, Lakshmi JVN. An analysis on linear regression algorithm implemented on HADOOP MapReduce. IJIRS Journal. 2014; 3(7):108–18.
20. Ronnie C, et al. SCOPE: Easy and efficient parallel processing of massive data sets. Proc VLDB. 2008; 1(2):265–76.
21. Dean J, Ghemawat S. Mapreduce: A flexible data processing tool. ACM Communications. 2010; 53(21):72–7.
22. Park HW, Yeo IY, Jang H, Kim NG. Study on the impact of Big Data Traffic caused by the Unstable Routing Protocol. Indian Journal of Science and Technology. 2015 March; 8(S5):59–62.