International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868
Foundation of Computer Science FCS, New York, USA
Volume 7– No.11, November 2014 – www.ijais.org

Comparative Study Load Balance Algorithms for Map Reduce environment

Hesham A. Hefny
Computer & Information Systems Department, Institute of Statistical Studies and Research, Cairo University, Egypt

Mohamed Helmy Khafagy
Computer Science Department, Fayoum University, Egypt

Ahmed M Wahdan
Computer & Information Systems Department, Institute of Statistical Studies and Research, Cairo University, Egypt

ABSTRACT
MapReduce is a famous model for data-intensive parallel computing in shared-nothing clusters. One of the main issues in MapReduce is the fact of depending its performance mainly on data distribution. MapReduce contains simple load balance technique based on FIFO job scheduler that serves the jobs in their submission order but unfortunately it is insufficient in real world cases as it missed many factors that impact the performance such as heterogeneity factor and data skewness, so Load balancing is important to make all resources utilized evenly and more efficiently. There are two main schemes in load balancing a- Static Load Balancing Schemes b- Dynamic load balancing. The main aim of this work is to study and compare existing Load Balance algorithms also to illustrate the features of Load Balance algorithms

Keyword
Static Load Balance, Map reduce,Dynamic Load Balance,static load balance,comparative study

1. INTRODUCTION
In order to handle huge data sets, using parallel processing tasks running on large clusters of computers, utilizing their combined resources is necessary. However, since developing such parallel programs from scratch is very difficult job and causing errors, programming models that support the parallelization automatically of processing jobs have gained a lot of importance in recent years. [1]

MapReduce is important for data-analysis in parallel computation in shared-nothing clusters. Its open-source implementation Hadoop is very famous and establishes the basis of many parallel algorithms in the last years [2] [3] [4] .

According to key/value data model, MapReduce allows programmers to define complex functions which are wrapped by map or reduce functions (second order functions) that guarantee that the input data is passed correctly to the user function parallel instances at runtime.

In MR, much of the processing is done by black-box user code.

The computation is expressed using two functions:

Map (k1,v1) $\rightarrow$ list(k2,v2);

Reduce (k2,list(v2)) $\rightarrow$ list(k3,v3). [2]

A Map Reduce Dataflow depend on three further functions. First, the PART function that partitions the map output and distributes it to the unused reduce tasks. Second COMP function that help in sorting all keys. Finally, GROUP function that for each reduce task it groups results to determine the data blocks for each reduce function call.[5] [6] Load balancing is used to make all resources are utilized evenly[7].

To balance load distribution, the load can be migrated from the source nodes (over utilized) to the underutilized one. [8]

The cost-effectiveness and scalability of MR implementations Relies on load balancing approaches to utilize available nodes. This is particularly challenging for data-analysis tasks where non balanced data can make some nodes bottlenecks.

Hadoop framework supplies a simple job scheduler FIFO which serves the jobs in order of their submissions. The sequential scheduler could ease the management of job to some extent and sometimes it is efficient when the framework deals with the job queue but in some cases this is not sufficient enough.

Unbalanced reducer workloads lead to high runtime differences, poor parallelism and the overall runtime increases. Hadoop framework performance is degraded on heterogeneous clusters due to load imbalance. There are four issues in imbalance:

- Imbalance in input splits,
- Imbalance in computations,
- Imbalance in partition sizes and
- Imbalance in heterogeneous hardware.

The rest of this paper is organized as follows section 2 describe and study Static Load Balancing algorithms, section 3 describe and study dynamic Load Balancing algorithms, section 4 describe comparative analysis of Load Balancing algorithms in environment of Map Reduce and finally section 5 shows the conclusion and future work.

2. STATIC LOAD BALANCING TECHNIQUES
In these type decisions are made before starting the execution. The system performs several experiments to collect information like execution time on a single processor, memory usage and so on. [9][10]

There are three categories:

Min-Min algorithm: first it finds the smallest fitness value of all the machines for each subtask. Then it finds the smallest value among the first step results. This algorithm is recursive and the end point occurs when the dispatch of all the subtasks happens. Both Levelized Weight Tuning (LWT) and Bottom Up algorithm are based on the DAG (Directed Acyclic Graph)

where subtasks dependencies are represented and dispatch these subtasks level-by-level, top to bottom or bottom to up,

Searching algorithms: uses search tree to find the best solution, it assures that the complexity is acceptable. The tree depth depends on the number of subtasks or available machines number. One example is A* algorithm.

Machine learning algorithms: widely used static load balancing algorithms. Its idea to randomly generate some dispatch patterns and generate new patterns from them. Let the fittest patterns survives in each step, then it generates new patterns in the next step.

All these algorithms consider execution time and energy cost. [10]

2.1 Cogset load balance

This part describe a static load balancing technique It depends on the replication of the data[11] partition on many nodes and uses this structure in load balancing. It transfers load between neighbor nodes. The key step in this algorithm is selecting which partition to process next, effectively implementing a distributed scheduling algorithm. To select the next partition to be processed, the node estimates the total amount of work remaining for itself, and for its neighbors, then selects the node with most work remaining, and determines which of the remaining partitions hosted by that node has the highest estimated processing cost, this partition will be the selected one [12]

3. DYNAMIC LOAD BALANCING TECHNIQUES

It applies load balancing and calculating cost during runtime, these algorithms must have less complexity than static algorithms. It is not recommended to use calculation intensive algorithms because the best solution at given instant may change according to new events. It is recommended to reach an effective solution in a short time. This can be done by two approaches according to the execution

Direct: it selects final destination node in one step.

Iterative: it determines the final destination node through several iterations [8]

Dynamic Load balancing can be done either in distributed or non- distributed way In the distributed one, the algorithm is performed by all nodes existed in the system and the load balancing task is shared between them.

Nodes can interact to reach load balancing in two methods: cooperative where nodes work together and non-cooperative where nodes are working independently. [13]

3.1 Adaptive MapReduce using Situation-Aware Mappers

In Adaptive MapReduce using situation aware mappers, the technique is based on breaking an assumption in MapReduce that all the mappers are independent, here the mappers are communicated asynchronously through a metadata store that is distributed so they are aware of the global state. The adaptive techniques are packaged as a library that can be used by Hadoop developers through an API this algorithm doesn't change the original MR APIs. It adds new runtime options to Hadoop to make them adaptive to the runtime environment. Adaptive algorithms are better in performance and more stable, they show resilience to tuning errors and changing runtime conditions. All systems offer various query "hint" mechanisms not normal cost-based optimizers. So using adaptive run-time algorithms is a logic choice. MapReduce can be adaptive and more flexible by providing the communication method between mappers. [14]

In this technique they are using asynchronous communication channel between mappers, transactional, distributed meta-data store (DMDS). So, the mappers can post some metadata about their state and be aware of the state of all other mappers ("situation-aware mappers" (SAMs)). SAM tasks can alter their execution, at runtime, depending on the global state.

SAMs are used in a number of adaptive techniques:

- Adaptive Mappers dynamically control the checkpoint interval to balance between performance, load balancing, and fault tolerance. After every split, AMs make a decision either to checkpoint or take another split and join it to already processed one(s). So the obtained result is: Minimum task startup overhead and dynamic load balancing.
- Adaptive Combiners use hash-based aggregation of map outputs with frequent keys and keeping the sort-based aggregation as a fallback option for non-frequent ones In case of a cache miss there are 2 replacement policies. No Replacement (NR), Least Recently Used (LRU). They maintain the benefit of shuffling and combine data in the reducers, while overhead eliminated.
- Adaptive Sampling uses map outputs to produce a sample of their keys and aggregates them in the global histogram. AMs are producing samples in separate files and when reach the condition needed to stop, the first mapper discovers that this condition is satisfied is nominated as leader and collects all samples in the global histogram. AS dynamically decides when to stop sampling. It eliminates the need for the stage of sampling to be rerun un the main query phase and dynamically determine when to stop , Adaptive Partitioning dynamically partitions map outputs based on the histogram produced from sampling phase.
- Adaptive Partitioning During job evaluation, AP determines the partitioning function. The main idea is for mappers to start processing data, but not produce any output. The AP piggybacks on AS, which aggregated map outputs into a histogram. This happens at a leader mapper. [14]

3.2 Block-based Load Balancing for Entity Resolution with MapReduce

ER (entity resolution) is the task of identifying entities referring to the same real-world object.

ER techniques compare pairs of entities by which is insufficient in Large Datasets [15].

Evaluating various similarity measures using Cartesian product is the approach used here. The complexity of this approach is O(n2) Using the blocking techniques is good for improving the efficiency. They utilize a blocking key depending on values of one or many entity attributes this will lead to partition the input data. The matching id limited to entities of the same block. [16]

3.2.1 Block-Based Load Balancing

In ER the input is a set of entities and the output are the entity pairs that are considered to be the same. ER is processed within two MR jobs. Both jobs are based on the same number of map tasks and the same partitioning of the input data. Block distri-

bution matrix (BDM) is calculated by the first job. It specifies the entities number for each block separated by input partitions. The matrix is used by the map phase of the second Map Reduce Job.

BlockSplit generates one or several coincide tasks per block and distributes matched tasks among reduce tasks.

It uses the following:

In a single match task, small blocks are processed similar to MR implementation while large blocks are split into m sub blocks.

BlockSplit determines the number of comparisons per match task and assigns matches tasks in descending size among reduce tasks [17] as in. Fig(1) that shows an overview of the MR matching process with block based load balancing

3.3 Load Balancing in MapReduce based on Scalable Cardinality Estimates

Data skew is very common in distributed databases systems and solutions for operations like joins and grouping/aggregation have been proposed [18][19][20] but these solutions are not applicable for MapReduce.

Scalable cardinality estimates algorithm uses the idea of approximately global histogram to address this problem.

Depending on their cost, the clusters are grouped into partitions that are distributed to the reducers. The cost of a partition is the sum of the costs of its clusters. This estimation is difficult due to the face that each mapper sees only a portion of the data.so a controller should be used to estimate all the costs which is based on short summaries. In addition, not all mappers do necessarily run at the same time.

Thus the controller cannot incrementally retrieve information as is done. Scalable cardinality estimates algorithm discusses TopCluster, a sophisticated distributed monitoring approach for MapReduce systems.

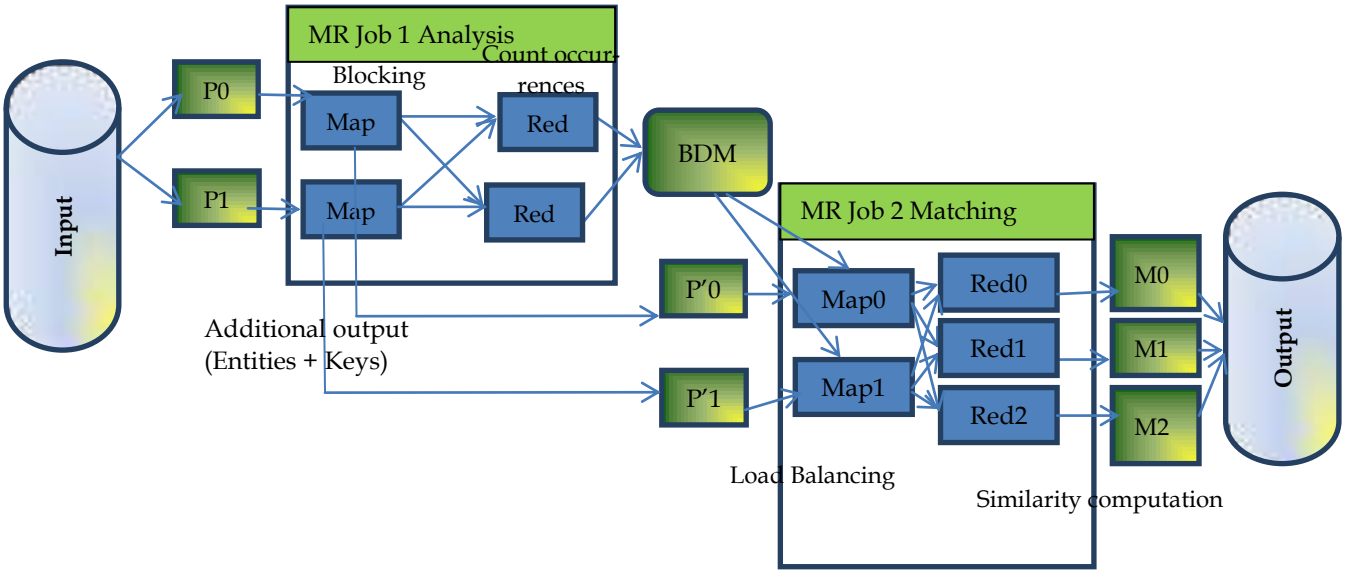

Fig 1 Overview of the MR matching process with block based load balancing [17]

TopCluster requires one parameter, the cluster threshold (τ), which controls the size of the local statistics that are sent from each mapper to the controller. The result is a global histogram of (key, cardinality) pairs.

TopCluster guarantees:

Completeness: All clusters with cardinalities above the cluster threshold τ are in the global histogram.

Error Bound: The approximation error of the cluster cardinalities is bound by τ/2.

Getting the largest clusters with high precision is critical for accurate cost estimation.

Also it provides an automatic good value for τ based on skew in data

3.3.1 Exact Global Histograms

Global Histogram

The exact global histogram stores all information required to compute the exact cost for all partitions. This is not feasible for large datasets. TopCluster algorithm is efficient and effective approximation of the exact global histogram.

Local Histogram definition: Let Ii be the bag of all intermediate (key, value) pairs produced by mapper i. The local histogram Li is defined as a set of pairs (k; v), where $k \in \{x \mid \exists y((x, y) \in Ii)\}$ is a key in Ii and v is the number of tuples in Ii with key k.

Global Histogram definition: Given m local histograms Li, $1 \leq i \leq m$, of pairs (k, v), where k is the key and v is the associated cardinality, and the presence indicator pi(k), which is true for k if and only if k exists in Li

$$Pi(k) = \begin{cases} \text{true if } \exists v\ ((k,v) \in L_i \\ \text{false otherwise} \end{cases}$$

The global histogram G is the set {(k, v)} with

$$\exists v\ \big((k,v) \in G\big) \Leftrightarrow \exists_i\ (P_i(k) = \text{true}$$

$$\forall v\ \big((k,v) \in G\big) : v = \sum \frac{1 \leq i \leq m}{(k,v) \in L_i} v'$$

3.3.2 TOPCLUSTER

To determine the global histogram TopCluster algorithm uses 3 steps

Local histogram (mapper): a local histogram Li is maintained on mapper i for each partition

Communication: For each partition it sends the presence indicator for all local clusters and the histogram for the largest local clusters (histogram head) to the controller.

Global histogram (controller): The controller approximates the global histogram using aggregation of the heads of local histograms and by estimation cardinality for all clusters that are not in the local histograms. [21]

3.4 Improving MapReduce Performance through Data Placement in Heterogeneous Hadoop Clusters

The current Hadoop implementation has 2 assumptions, computing nodes in a cluster are homogeneous Most maps are data-local (Data locality has not been considered,).

Unfortunately, both assumptions are not satisfied in virtualized data centers[22].

Data placement in a heterogeneous environment algorithm addresses the problem of placing data across nodes to achieve a goal that each node has a balanced data.

Data locality is a determining factor for the MapReduce performance. In a heterogeneous cluster nodes with higher performance can complete processing their local data faster than lower performance ones.

After the fast node finished processing their own data, it has to deal with unprocessed data in a far slow node. The expense of this data transferring from slow to fast nodes is high if the amount of moved data is huge.

The proposed algorithm discusses a data reorganization and redistribution algorithms in HDFS.

The MapReduce program routes queries to a name-node, which in passes the file requests to the correct data nodes. Then, they input large amount of data to Map functions. File fragments of the file are stored on multiple data nodes within the cluster during writing new data to a file in HDFS.

It is powerful to move data processing tasks to nodes where application data are located if the cluster has nodes with a local disk for each one, To improve the performance of Hadoop in heterogeneous clusters, data movement between fast and slow nodes should be minimized.

This can be done by a data placement scheme that balance data storage across nodes based on their computing capacities.

Making a complete replica of the data on each node is not suitable due to waste of resources and this solution is very expensive.

To address limitations of the data-replication approach, the algorithm is using an approach for files partitioning and distribution across multiple nodes in a Hadoop cluster without duplication.

In this mechanism, two algorithms are implemented and incorporated into Distributed file system [23, 24, 25] like HADOOP's HDFS. The first algorithm is initially distribute file fragments to heterogeneous nodes in the cluster. Then, these file fragments are distributed to the computing nodes. The second algorithm is used to reorganize file fragments; this will solve the problem of data skew.

The initial data placement starts by dividing a large input file into a number of equal sized fragments.

Then, assigns fragments to nodes in a cluster in accordance to the nodes' data processing speed.The calculation of nodes speed is done by computing ratio. After placement, Input file fragments distributed [26]

3.5 Randomized Hydrodynamic Load Balancing:

Randomized Hydrodynamic Load balance algorithm uses the hyper approach for dynamic load balance Here the load balancing algorithm balance the disk space usage on HDFS cluster when some data nodes became full or when new empty nodes joined the cluster. The balancer starts with a threshold value, (a fraction from 0% to 100%).

With smaller threshold values the cluster will become more balanced and the balancer will take longer time. The cluster is considered balanced if for each data node, The difference of the node utilization and the cluster utilization <= threshold value Utilization of the node: used space of the node/total capacity of the node Utilization of the cluster: used space of the cluster/total capacity of the cluster Moving the blocks from the highly utilized data nodes to the seedy used ones is made by iterations.

Depending upon the utilization rating of each node, nodes are classified as

- Highly-utilized.
- Average-utilized
- Under-utilized.

The algorithm steps:

The module gets neighborhood details:

When a DataNode load reaches the threshold level, it sends a request to the NameNode which had information about the load levels of the nearest neighbors of this node.

The details about the freest neighbor nodes are sent to the requester DataNode after comparing Loads by the NameNode.

Each DataNode compares its own load amount with the sum of the load amount of nearest neighbors.

If a DataNode's load level is greater than the sum of its neighbors, then load-destination nodes (direct neighbors and other nodes) will be chosen at random.

Load requests are then sent to the origin nodes.

Last, the request is received.

Buffers are preserve at every node to received load

A message passing interface (MPI) manages this buffer.

A main thread will listen to the buffered queue and will service the requests it receives.

The nodes enter the load-balancing-execution phase. [27]

3.6 Resource-Aware Adaptive Scheduling for MapReduce Clusters

This work present RAS, a Resource-aware Adaptive Scheduler for MapReduce to increase resource utilization which is guided by completion time goals. It also addresses the system administration problem of configuring the number of slots for each machine. [28]

Resource-Aware Adaptive Scheduling algorithm has the following features:

Extends the abstraction of "task slot" to "job slot" that is related to a job with market profile.

Leverages resource profiling information for better resource utilization Dynamically allocating resources to jobs to Adapt to change in resource demand At decision related to resource-aware scheduling it differentiate between map and reduce tasks RAS is a combination between resource awareness which is used to determine task placement on Task-Trackers over time and continuous job performance management the algorithm uses this information to determine the number of parallel tasks for each job depending on performance.

Most of the logic behind RAS resides in the Job-Tracker in each job submitted there are two pieces of information configured in the job configuration file Job completion time goal (optional) Resource consumption profile Active jobs list and active tasks list are maintained in the Job-Tracker. Information stored for each active job includes their submitted time information and state information like number of pending tasks. Regarding task-tracker it stores its resource capacity.

When the task completed, the Task-Tracker notifies the Job Status Updater, which triggers an update of pending maps and reduces in the job descriptor. The Job Status Updater also has the information about the average task lengths; this information is used to estimate the completion time for each job.

Placement Algorithm and the Job Utility Calculator are considered the core of RAS. They operate in control cycles of period T. They produce a new placement matrix P that will be active until reaching (current time + T). If it is required from the system to change in the task and respond fast to new job submissions. To choose the best placement choice available, the Job Utility Calculator evaluates a value for utilization to input placement matrix then it is used by the Placement Algorithm.

Placement decisions are used by the Task Scheduler. The Task Scheduler schedules tasks using Placement Controller decision. After task completion, Task Scheduler selects a new task to execute in the free slot. fig(2) shows Resource-Aware Adaptive Scheduling system architecture [29]

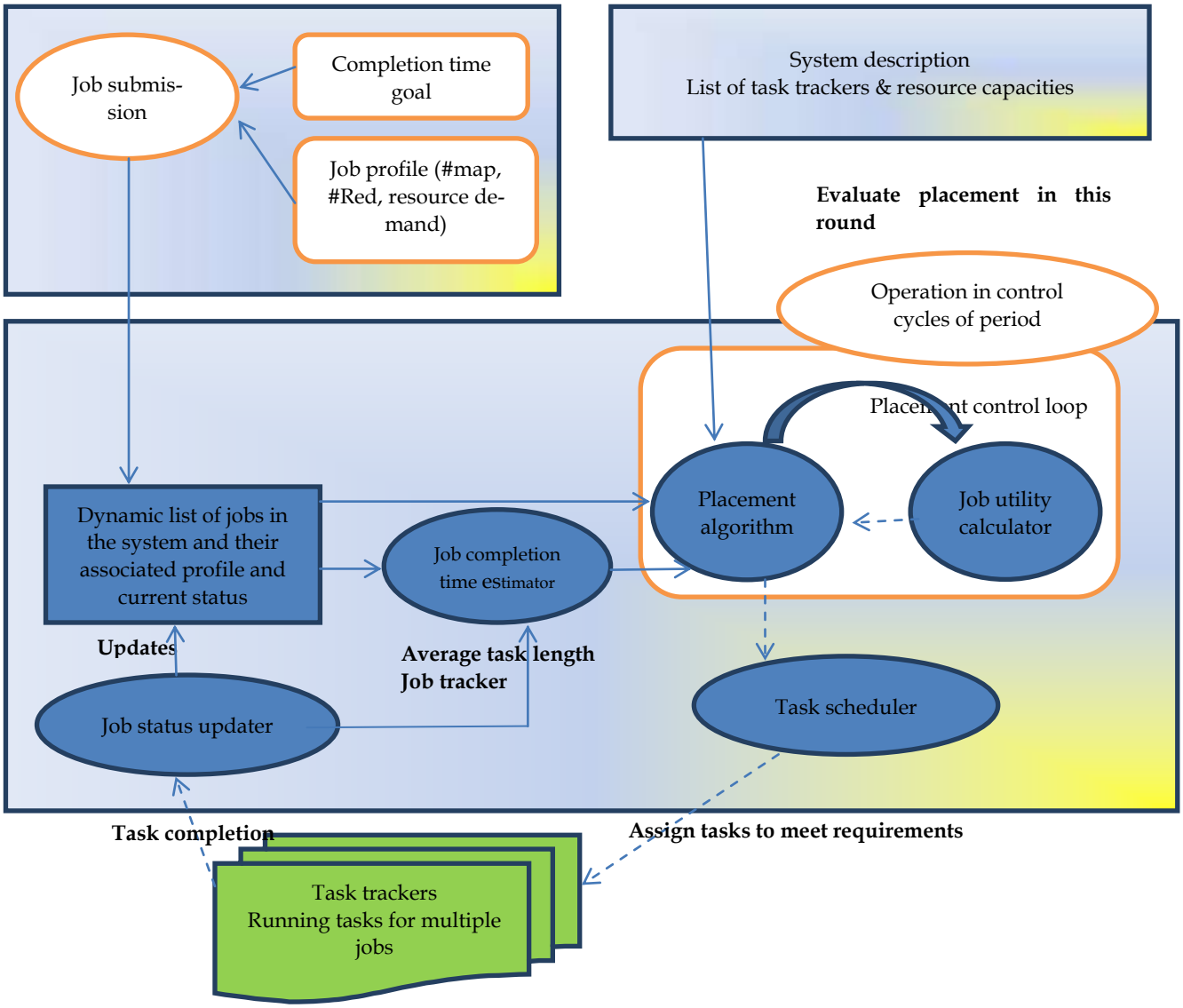

Fig 2 Resource-Aware Adaptive Scheduling system architecture [29]

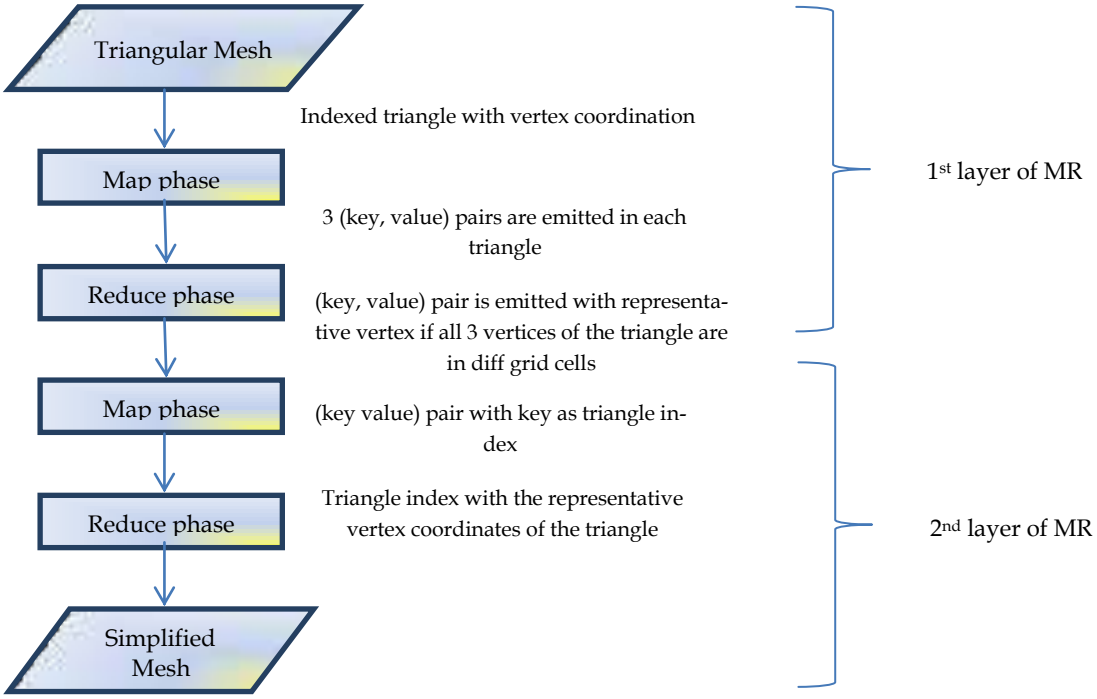

Fig 3 Resource-Aware Adaptive Scheduling system architecture [25]

3.7 Mesh Simplification Algorithm

Mesh algorithm suggests an algorithm of load balancing based on hashing key values over a certain range then construct a histogram contains these hash values which are then assigned to reducers

This task is done over sample data randomly taken it suggests 25% of the input data as sample. Then it uses this histogram to assign data to reducers. The master data and histogram are available in the master node after that it uses the results from previous phases to do the real map-reduce task

The reducers' inputs are decided by the custom Partitioner which uses the partitions provided in the partition file [30] fig(3) shows Mesh Simplification algorithm

3.8 Dynamic Load Balancing Schemes in Stream Based Scenarios

Stream based scenarios usually involve a leveled network and each level of network is responsible for particular operations. Each task includes a series of operations and each operation can be performed by one level in the network. These tasks are unpredictable. The network does not have previous knowledge how many tasks will arrive over the next few seconds. Event simulation is also not synchronized during execution.

So it is recommended to use machine learning algorithms as they tend to perform much better.

Ant-colony algorithm is one example for stream-based scenarios dynamic load balancing. The suggested algorithm is based on three types of ants with different functionalities. The modified algorithm keeps store more information but the main idea still involves searching the path randomly and leaving pheromones in the path while passing by.

Unlike other systems, extremely selfish behavior in this system is acceptable. The goal of the algorithm here is to reach the minimum average latency. The first task should be served as fast as it can because nobody knows how many other tasks may arrive in the near future. Also, the algorithm usually does not work as efficiently in the beginning as it is in the learning phase. [9][31]

3.9 Variable-sized map and locality-aware reduce on public-resource grids

Variable-sized map and locality-aware reduce on public-resource grids discusses Ussop algorithm.

The Ussop portal chooses several grid nodes to run the application when this MR application is submitted to portal. It nominates one node as the master of the application and the others as workers. Each idle worker requests a map reduce task from the master.

As the grid nodes used by Ussop are chosen on market, the input data cannot be stored in these grid nodes in advance. The input data is come from the user's node that submitted the job or from remote replica.

The grid nodes are usually from various geographically distributed sites and they are heterogeneous and non-dedicated.

3.9.1 Variable-sized map scheduling (VSMS)

During the execution, a task may be performed at the different rate even if it is executed by the same worker all the time. The speculative execution scheme is not robust enough for the Ussop.

Ussop has to balance the workload between all workers by adjusting the size of a task dynamically according to computing capability of the workers. Also it can avoid misjudging a poor performance node as a faulty one. Ussop uses the variable-sized map scheduling algorithm (VSMS) to determine the suitable size of the task that should be assigned to each worker. VSMS is based on the concept that the master should assign coarser-grained tasks to the workers with more powerful computing availability.

The VSMS algorithm consists of two procedures.

When a given worker requests a map task, the master re-

estimates the appropriate map task size

Each worker updates its current computing power and the estimated remaining time of task every (n) seconds.

VSMS assumes that the progress of a map task is proportional to how much input data has been processed.

So the computing power (CP) is how much input data can be processed by the worker per second MB/s

3.9.2 Locality-aware reduce scheduling (LARS)

Ussop uses a just-in-time scheduling algorithm called locality-aware reduce scheduling (LARS). When a worker requests a reduce task, the master uses the LARS algorithm to choose an appropriate reduce task, and assign it to the work.

When the reduce task is assigned to the node, LARS starts input data transfer. This algorithm aims to minimize the data transfer cost. The master knows that which non-assigned task processes the largest local region. Then it assigns the reduce task which processes the largest local region to the idle worker. So each node can avoid transferring large local regions that the node owns to other nodes.

3.10 Ant colony Optimization

One of the most successful and widely recognized algorithmic techniques based on ant behavior is the ability to find the shortest paths.

Ant based control system was designed to solve the load balancing in the cloud environment. Each node in the network was configured with Capacity Probability of being a destination.

Pheromone: probability routing table.

Every row in this table defines the routing preference for each destination, and each column represents the possibility of selecting the next hop from neighbors. Ants are started with a random target from the node.

In this approach, incoming ants update the entries.

The updated routing information can influence the routing ants that have as their destination. This approach for updating the pheromone is only for routing in symmetric networks.

If there is no pheromone it makes a random decision. Paths from its colony are preferable.

In the case of load balancing in cloud environment, as the web server demands increases or decreases, the Services are assigned dynamically to regulate the changing demand of the user. Virtual Server (VS) is a group of all servers with virtual queue. Each server processing a request from its queue calculates a reward. The server after processing a request can post their profit in the pheromone table. The server can choose a queue of a virtual server by a probability. A server serving a request, calculating its reward and compare it with the colony reward and then set the probability.

One limitation of this solution is that it will be more efficient if cluster is formed in the cloud. So, the research work can be proceeded to implement the total solution of load balancing in a complete cloud environment [32]

3.11 The Partition Cost Model

It considers both skewed data distributions and complex reducer side algorithms. Based on cost for each partition they are distributed on reducers so work in reducers is balanced. The cluster cost, is a function of the cluster cardinality and the complexity of the reducer side algorithm. The reducer complexity is a user parameter but the cluster cardinalities are monitored by the framework.

Two algorithms have been developed to use the partition cost model: fine partitioning which splits the input data into n partitions. N is larger than reducers number, the goal here is to distribute the partitions in a way that makes similar execution times for all reducers. Fine partition algorithm achieves load balancing by allocating costly partitions to different reducers.

The second algorithm is dynamic fragmentation where each mapper splits expensive partitions locally during their creation; replication of tuples can be done at necessity. So, the partitions cost distribution is more uniform and better load balancing can be achieved easier for highly skewed data.

In partition cost model, the reducer workload should be distributed evenly to all nodes participating in the model. By doing so, it maximizes utilization of resources. In addition to that, it minimizes the job completion time due to load balancing, as parallel processing is used better. [33][34]

3.12 Load Balancing Methodology based on Divisible Load Theory

Divisible Load Theory (DLT) [35] describes a divisible data set as one that can be split into several independent splits or chunks of random size to be parallelly processed.

LB based on DLT algorithm monitors the processing time of data splits to specify their schedule order for future examinations, and to adapt the partition factor dynamically

When partitioning costs are high, multiple data set partitions are created by using different partition factors before the application execution, and then select the most suitable one according to the situation and circumstances.

Communication cost, usage of memory and the resources availability are considered besides the time of processing.

The algorithm includes:

- The creation of multiple data set splits before the application run in case of high partitioning cost.
- The monitoring of the processing time
- The changing of the data splits orders and distribution during the application execution
- The partition factor selection according to the monitored efficiency

Number of processing nodes expectation and assessment for better efficiency. [36]

The algorithm determines (i) the number of proportions for dividing the initial workload; (ii) the data chunks scheduling strategy; and (iii) the processing node number.

There are two phases

Phase1: generating an initial partition of the data set. If the generation of a new partitions cost is high then it went to generate alternative partitions before the execution and chose the most suitable one during the execution.

Phase2: measuring and evaluating the performance for tuning.

Table 1 comparative study of load balance algorithm in MR
S:Static D:Dynamic Y:Yes N :No F:Fixed V:variable

Algorithm	type	Component	MR pahses	using saved data	Sampl-ing	data awareness	avaialbility	scalability	suitable application	hetrogenity	target time	data size	communicate
Adaptive MR using Situation-Aware Mappers	D	Adaptive Combine Sampling partitioning	1	Y	Y	Y	Y	Y	-	N	N	F	Y
Block-based for Entity	D	BDM Blocksplit	2	N	N	Y	N	Y	Entity resolution	N	N	V	N
Scalable Cardinality Estimates	D	TopCluster Global histogram Local histogram	1	Y	Y	N	N	Y	-	N	N	F	Y
shortest path	D	initially distribute fragments initially distribute fragments	1	N	-	N	N	Y	hetrogeni ous cluster	Y	N	F	N
Randomized Hydrodynamic	H	normal hadoop structure with MPI	1	N	N	N	N	Y	-	N	N	F	N
cogset	S	depends on data replication	1	Y	N	N	Y	N	Replicatio n	N	N	F	N
Resource-Awar	D	job slots resoutce awareness resoutce awareness	1	N	N	N	N	Y	-	Y	Y	F	N
mesh simplificatio	D	hashing key histogram partitioner	2	Y	Y	N	N	Y	-	-	N	F	N
Stream Based	D	machine learning	1	N	N	N	N	Y	network streaming	Y	N	F	N
locality-aware	D	USSOP VSMS LARS	1	N	N	N	N	Y	hetrogeni ous clusters	Y	N	V	N
Ant colony	D	shortest path random decisio	1	N	N	N	N	Y	cloud applicatio ns	Y	N	F	N
partition cost	D	fine partitioning	1	N	N	Y	N	Y	skewed data	N	Y	V	N
LB based on DLT	D	decision made	1	N	N	Y	N	Y		N	Y	V	N

4. COMPARATIVE ANALYSIS OF ALGORITHMS

Table 1 summarizes the algorithms that discussed in this paper, Cogset is the static algorithm discussed. It is based on the replication and choose the next partition to process. Adaptive MapReduce using situation-aware mapper has a good technique of communicating all the mappers with each other it uses ZooKeeper as DMDS to manage this asynchronous communication it also uses the high availability by using two or more zookeeper servers it uses sampling and histogram. In the block based balancing for ER MR it is based on dividing large blocks into sub-blocks and then treat the small blocks as normal MR. Load Balancing in MapReduce Based on Scalable Cardinality Estimates uses the TopCluster algorithm to estimate the approximate histogram as the exact histogram is expensive. Improving MapReduce Performance through Data Placement in Heterogeneous Hadoop Clusters addresses the heterogeneity in hardware by distributing the file fragments over the nodes and redistribution if new conditions occurred. Randomized Hy-

drodynamic Load Balancing uses the balancing of the HDFS space it uses a threshold from 0% to 100% the smaller the threshold the more balanced cluster and the longer time used for balancing. RAS is using the concept of job slot and it contains profile for hardware and the target time needed as optional parameter. Placement Algorithm and the Job Utility Calculator are considered the core of RAS they are estimated periodically. Mesh simplification algorithm uses two phases the first one is for 25% sampling and the other one is for real task performance. Dynamic Load Balancing Schemes in Stream Based Scenarios uses the machine learning. In Variable-sized map and locality-aware reduce on public-resource grids the map size is variable USSOP portal uses several nodes for the application and nominate one of them as master of the application. Idle nodes request the data from the master. Ussop has to balance the workload between all workers by adjusting the size of a task dynamically according to computing capability of the workers. It also uses locality-aware reduce scheduling (LARS). When a worker requests a reduce task, the master uses the LARS algorithm to choose an appropriate reduce task, and assign it to the work. In ACO, each node in the network was configured with Capacity Probability of being a destination. Pheromone is the probability and based on the value it finds the shortest bath. Partition cost model considers both skewed data distributions and complex reducer side algorithms. Two algorithms are used, fine partitioning and dynamic fragmentation. The reducer workload is distributed evenly to all nodes participating in the model. LB based on DLT monitors the processing time of data splits to specify their schedule order for future examinations, and to adapt the partition factor dynamically.

5. CONCLUSION AND FUTURE WORK

In this paper the load balancing techniques are studied in Map Reduce environment and made a comparison between them from some aspects. there are suggestions for enhancing some algorithms as future work to reduce the cost of load balance also it is suggested to implement the join algorithms using several bnchmarks[37,38] and conduct performance measurement. The suggested enhancement is to use data similarity to estimate the execution plan based on previous runs based on histogram and taking samples from runs it is possible to deduce the optimum execution plan for this run which reduce execution time.

6. REFERENCES

[1] A. Alexandrov, S. Ewen, M. Heimel, F. Hueske, O. Kao, V. Markl, et al., "MapReduce and PACT - Comparing Data Parallel Programming Models," 2010.

[2] R. Vernica, M. J. Carey, and C. Li, "Efficient parallel set-similarity joins using MapReduce," Proceedings of the 2010 international conference on Management of data, 2010.

[3] J. Cohen, "Graph Twiddling in a MapReduce World," 2009.

[4] "http://lucene.apache.org/mahout/."

[5] L. Kolb, A. Thor, and E. Rahm, "Load Balancing for MapReduce-based Entity Resolution."

[6] "hadoop", http://hadoop.apache.org."

[7] Ebada Sarhan, Atif Ghalwash,Mohamed Khafagy ,Queue Weighting Load-Balancing Technique for Database Replication in Dynamic Content Web Sites ",APPLIED COMPUTER SCIENCE (ACS'09) University of Genova, Genova, Italy, 2009, Pages 50-55

[8] R. Mishra and A. Jaiswal, "Ant colony Optimization: A Solution of Load balancing in Cloud," International journal of Web & Semantic Technology, vol. 3, pp. 33-50, 2012.

[9] Z. Sui and S. Pallickara, "A Survey of Load Balancing Techniques for Data Intensive Computing," 2011.

[10] S. Shivle, R. Castain, H. J. Siegel, A. A. Maciejewski, T. Banka, K. Chindam, et al., "Static mapping of subtasks in a heterogeneous ad hoc grid environment," Proc. of 13th HCW Workshop, IEEE Computer Society, 2004.

[11] Ebada Sarhan, Atif Ghalwash,Mohamed Khafagy,Agent-Based Replication for Scaling Back-end Databases of Dynamic Content Web Sites",ICCOMP'08 Proceedings of the 12th WSEAS international conference on Computers WSEAS,GREECE 2008 Pages 857-862

[12] S. V. Valvåg, "Cogset: A High-Performance MapReduce Engine," 2011.

[13] D. Escalante and A. J. Korty, "Cloud Services: Policy and Assessment," EDUCAUSE Review, vol. 46, 2011.

[14] R. Vernica, A. Balmin, K. S. Beyer, and V. Ercegovac, "Adaptive MapReduce using Situation-Aware Mappers," 2012.

[15] R. Baxter, P. Christen, and T. Churches., "A comparison of fast blocking methods for record linkage," Workshop Data Cleaning, Record Linkage, and Object Consolidation, 2003.

[16] H. K¨opcke, A. Thor, and E. Rahm, "Evaluation of entity resolution approaches on real-world match problems," PVLDB, vol. 3, 2010.

[17] L. Kolb, A. Thor, and E. Rahm, "Block-based Load Balancing for Entity Resolution with MapReduce," 2011.

[18] D. J. DeWitt, J. F. Naughton, D. A. Schneider, and S. Seshadri, "Practical Skew Handling in Parallel Joins," 1992.

[19] J. W. Stamos and H. C. Young, "A Symmetric Fragment and Replicate Algorithm for Distributed Joins,," IEEE TPDS, vol. 4, 1993.

[20] W. P. Yan and P.-A. Larson, "Eager Aggregation and Lazy Aggregation," vldb, 1995.

[21] B. Gufler, N. Augsten, A. Reiser, and A. Kemper, "Load Balancing in MapReduce Based on Scalable Cardinality Estimates."

[22] Ebada Sarhan, Atif Ghalwash, Mohamed Khafagy, Queue weighting load-balancing technique for database replication in dynamic content web sites, Proceedings of the 9th WSEAS International Conference on APPLIED COMPUTER SCIENCE 2009

[23] Khafagy, M.H. ; Feel, H.T.A.,Distributed Ontology Cloud Storage System" IEEE,Proceedings of the 2012 Second Symposium on Network Cloud Computing and Applications Pages48-52

[24] al Feel, H.T. ; Khafagy, M.H.OCSS: Ontology Cloud Storage System",IEEE Network Cloud Computing and Applications (NCCA), 2011 First International Symposium on Pages 9-13

[25] Haytham Al Feel, Mohamed Khafagy, Search content via Cloud Storage System. International Journal of Computer Science Issues (IJCSI)bVolume 8 Issue 6, 2011

[26] J. Xie, S. Yin, X. Ruan, Z. Ding, Y. Tian, J. Majors, et al., "Improving MapReduce Performance through Data Placement in Heterogeneous Hadoop Clusters," 2010.

[27] K. A. Venkatesh, K. Neelamegam, and R. Revathy, "Using MapReduce and load balancing on the cloud Hadoop MapReduce and virtualization improves node performance," 2010.

[28] J. Polo, D. Carrera, Y. Becerra, M. Steinder, and I. Whalley, "Performance-driven task co-scheduling for MapReduce environments," Network Operations and Management Symposium IEEE, pp. 373-380, 2010.

[29] J. a. Polo, C. Castillo, D. Carrera, Y. Becerra, I. Whalley, M. Steinder, et al., "Resource-Aware Adaptive Scheduling for MapReduce Clusters," 2011.

[30] S. C. Racha, "Load Balancing Map-Reduce Communications for Ecient Executions of Applications in a Cloud," 2012.

[31] G. T. Lakshmanan and R. Strom., "Biologically-inspired distributed middleware management for stream processing systems," ACM Middleware conference, 2008.

[32] Y.-L. Su, P.-C. Chen, J.-B. Chang, and C.-K. Shieh, "Variable-sized map and locality-aware reduce on public-resource grids," Future Generation Computer Systems, vol. 27, pp. 843-849, 2011.

[33] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters.," CACM, 2008.

[34] B. Gufler, N. Augsten, A. Reiser, and A. Kemper, "The Partition Cost Model for Load Balancing in MapReduce," 2012.

[35] Bharadwaj, R. V., T.G., and D. Ghose, "Scheduling Divisible Loads in Parallel and Distributed Systems," IEEE Computer Society Press, Los Alamitos, 1996.

[36] C. Rosas, A. Sikora, J. Jorba, A. Moreno, and E. César, "Improving Performance on Data-Intensive Applications Using a Load Balancing Methodology Based on Divisible Load Theory," International Journal of Parallel Programming, vol. 42, pp. 94-118, 2012.

[37] Ebada Sarhan, Atif Ghalwash, Mohamed Khafagy, Specification and implementation of dynamic web site benchmark in telecommunication area, Proceedings of the 12th WSEAS international conference on Computers 2008 Pages 863-86

[38] www.tpc.org