Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 46 (2015) 627 – 634

Procedia Computer Science

International Conference on Information and Communication Technologies (ICICT 2014)

A Two Pass Scheduling Policy based Resource allocation for MapReduce

Ebin Deni Raj and Dhinesh Babu L.D *

School of Information Technology and Engineering,VIT University,Vellore-632014,India

Abstract

Big Data and parallel computing are used extensively for processing large quantities of data, structured, semi structured or totally unstructured. MapReduce and Hadoop are used for the parallel data processing of these kinds of data. Various scheduling policies are used for MapReduce scheduling which is discussed in detail and a new scheduling technique Two Phase Scheduling Policy (TPSP) based resource allocation for MapReduce is implemented and the efficiency is verified.

Peer-review under responsibility of organizing committee of the International Conference on Information and Communication Technologies (ICICT 2014)

Keywords: *Big Data; Hadoop; MapReduce; scheduling policies*

1. Introduction

Big Data has become the buzz word after cloud computing and grid computing and this new technology works together with distributed data processing systems to create useful insights from gigantic amount of data[1]. Many parallel data processing systems evolved over the past decade, out of which Hadoop is the one with much prospects ahead. The word Mapreduce originates from the higher order functions which accept other functions as arguments. MapReduce can be defined as the distributed processing across multiple machines and servers. It is an innovative

* Corresponding author. Tel.: +91-9486939119;
E-mail address: :lddhineshbabu@vit.ac.in

Peer-review under responsibility of organizing committee of the International Conference on Information and Communication Technologies (ICICT 2014)
doi:10.1016/j.procs.2015.02.110

programming model exclusively for data processing[2]. The greatest advantage of MapReduce is that it reduces the execution time of complex tasks considerably. The process consist of two phases namely Map and Reduce. In other words it can be said that it consist of two user defined functions – Map function and Reduce function. MapReduce helps in reducing the complexities of distributed file systems[3]. The basic idea of this technique is sectionalisation of large data into small data chunks. The MapReduce framework is best suited for applications and systems having little or no communication between nodes. This concept is widely known as shared nothing architecture. Apart from Google, MapReduce is widely used by Facebook, MySpace and Twitter for data analysis and social computing activities.

1.1 Comparison of MapReduce and RDBMS

In the last few years the Data growth has become so exponential that it grew from terabytes to petabytes and even to zeta bytes[4]. The RDBMS approach and MapReduce approach differ from each other. RDBMS operates on structured data such as definite rows and columns or on structured XML data[5]. Unstructured data such as image, voice, and video which do not have internal structure cannot be processed by RDBMS. MapReduce has the capacity to process different kinds of unstructured data consisting of variety of text, image and many more[6]. The interpretation of data happens during processing of data. Unlike RDBMS which possesses a static schema, MapReduce has a dynamic schema in which the schema changes continuously[7]. Hive is an apache project which was incorporated in hadoop with the goal of integrating SQL skills to manage huge volumes of data .The programming model of MapReduce is defined in such a way that it is linearly scalable across a large number of machines with low implementation cost. In RDBMS the data needs to be normalized and cleaned before processing whereas MapReduce processes the data without doing so. MapReduce has lower integrity when compared with RDBMS and the pros of RDBMS can be incorporated into MapReduce to make the framework more efficient.

1.2 Comparison of MapReduce with Divide and Conquer

The basic idea is similar to divide and conquer but in a distributed and highly scaled proportion. A single machine cannot be used to compute and solve very large datasets as the dataset grows uncontrollably. To increase the efficiency of the process, multiple machines are employed in which divide and conquer method faces so many stumbling blocks such as coordination, reliability and failed process machine. Hadoop framework will help in solving these stumbling blocks and help in executing parallel processing with less effort. Hadoop Distributed File Systems (HDFS) helps in scaling out the input to multiple machines. HDFS also helps in data locality optimization.

MapReduce programs have its base in functional programming paradigm and the main objective of the programmer is to define and create map function and reduce function. One of the highlights of functional programming paradigm is that it accepts other functions as function arguments. The primitive data types in MapReduce programs are key-value pairs which allows programmers to concentrate on business needs and keeps him/her free from the distributed file systems complexities. The Map task produces a set of key value pairs from the data chunks which is the input argument to the reduce task. MapReduce uses a shared nothing model which means that each node in the architecture is self sufficient, independent and hence, does not require intra communication. This no-shared model helps to achieve scalability without much complexity and we can cite the advantages of MapReduce as high fault tolerance, comfortable programming structure, automatic distribution of workload and high elasticity.

1.3 Overview of Hadoop

Hadoop is an open source project implementation of Google's MapReduce architecture. The files will be of very large sizes in the range of Terabytes or Petabytes and is built on top of low cost hardware. Data access in HDFS is by bucketing or streaming method. HDFS basically follows Google file system and the advantage is greater scalability and very high availability. It is both hardware and software tolerant and has moderately good fault tolerance as it does automatic re-replication of data on failed nodes. Hadoop addresses the two big challenges put forward by Big Data; firstly, how to store large volumes of data and secondly, how to work with copious data. Hadoop Distributed File System comprise of a client server architecture consisting of a name node (also called as master node) and a set of data nodes. The name node has a job tracker which keeps monitoring the task assigned to the data node. The data node contains a task tracker which constantly supervises the task given by the name node. The Hadoop environment

setup is neatly depicted in Fig 1. The replica process is handled by name node. HDFS also has a secondary node which can be configured in some other system other than the one in which name node resides. But in case of failure of a name node the secondary node cannot replace the name node.

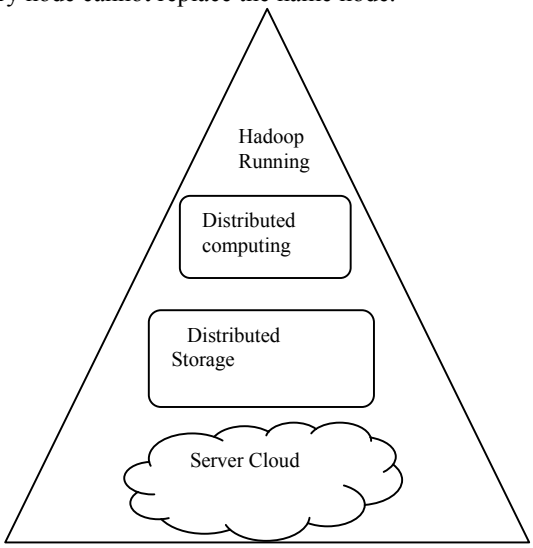

Fig 1: Hadoop Environment

The duty of secondary name node is to perform frequent checks on the name node and to store images of the name node. In case of a failure of name node, the most recent image will be uploaded into the name node by the secondary node. HDFS is introduced with the motive that it is easier to move the computation rather than moving the data. MapReduce has been implemented in various forms such as big table, MongoDB and couchdb. Hadoop is just another implementation of MapReduce. Some of the high level languages used in hadoop are Crunch, Pig and Hive. The predictive analytics used in Hadoop consists of Rhadoop, Rhive and RMahout.

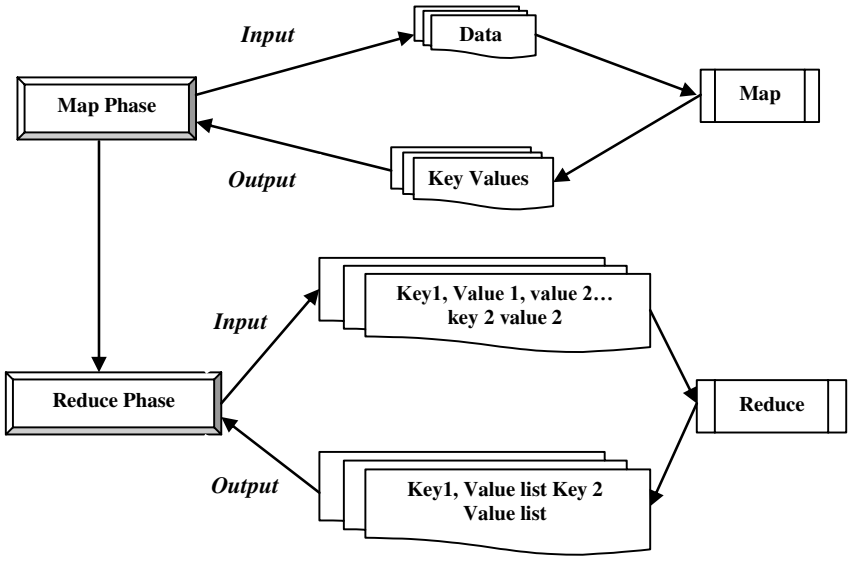

Fig 2: Working of MapReduce

1.4 Hadoop-YARN

MapReduce 2.0 or Yarn has many upgrades from the previous versions of Hadoop[8]. The previous versions of

Hadoop had disadvantages such as maximum cluster size and maximum concurrent tasks being 4000 nodes and 40000 nodes respectively. In YARN architecture each data node has a node manager (NM). The job tracker has two daemons which will assist in cluster resource management and scheduling/monitoring. A global resource manager resides in the name node. Each Data node contains containers. The containers are supervised by node managers. There is an application master which coordinates an application job. The application master will be managing the lifecycle of the applications running in the cluster and it has the privilege of talking directly to the resource manager. Each application master has a set of containers spread over different data nodes under its supervision. MapReduce jobs are managed by the application master. The node managers make sure that an application does not over use the resources.

2. MapReduce scheduling

Most of the MapReduce scheduling algorithm is based on the assumption that nodes are homogeneous. This assumption poses a drawback in scheduling algorithms. The selection of data nodes are done in random and the duration of MapReduce tasks vary from few seconds to few days[9]. The mean duration of map tasks and reduce tasks varies from job to job and task to task depending on the number of worker nodes. The allocation strategies in cloud includes apportioning of resources and time slices in a variety of modes[10,11].

2.1Task scheduling

MapReduce follows a pull scheduling scheme in which the slaves (Task Tracker) gets the job from master node on request. In Map Task scheduling the slaves send periodic heartbeat (to check whether the node is alive) to the master node. The slaves are also referred to as slots and if a slot is free, the master responds with a task that is in closest locality to the slave node. While in reduce task scheduling the master sends the next yet to be run task to the slave irrespective of its locality. Thus Map task considers locality of nodes while reduce task does not consider the locality.

2.2 Job scheduling

In MapReduce framework each application is considered as a job. A job comprises of many map and reduce tasks. MapReduce framework have many type of schedulers mainly First In First out (FIFO) scheduler and Fair scheduler. FIFO scheduler, as the name suggests, processes jobs in the order of submission while Fair scheduler aims to provide a fair share of the cluster capacity over time. The problem with FIFO scheduler is that it reduces data locality since it follows strict FIFO job order and this scheduling technique cannot provide fair opportunity for all task nodes. The Hadoop scheduler assigns the task with the help of a queue.

2.3 Comparison of Resource allocation policies

The MapReduce framework consists of two scheduling techniques namely First Come First Serve (FCFS) scheduling policy, Fair scheduler and capacity scheduler. The thirst for more efficiency has led into developing new scheduling techniques for both job scheduling and task scheduling. The initial release of Hadoop included only two resource allocation policies namely fair scheduler and FCFS. Later versions included the capacity scheduler in the scheduling policy of MapReduce. The drawback of FCFS scheduling policy is low resource utilization. There are many varieties of fair scheduling such as naive fair scheduling, and fair scheduling with delay scheduling. Table 1 shows the different policies proposed in different papers.

The first come most resource allocation policy gives priority to the task that starts first. The Laggard Tasks most scheduling policy makes use of a concept known as resource stealing. The longest time left scheduling policy checks each job's and task's estimated execution time and reallocates the resources accordingly. The resource allocation policy *speculative task most* is designed to avoid slow tasks by allotting multiple nodes to process the same data. The load balancing techniques used in cloud computing can be mimicked to a certain extent in MapReduce as well[12].

Table 1: Different MapReduce resource allocation policies

Strategy	Description
First Come First Serve Scheduling	Allot the resource according to the input arrival
First Come most	Task that starts first is given the resource
Fair share scheduling	Each task is ensured a reasonable share of the resource
Capacity Scheduling	A combination of fair share scheduling and FCFS scheduling
Shortest time left most	The task that will complete in the most nearest time will be given the resource
Longest time left most	The task that will be completed last will be given the resource
Speculative task Most	They are given the remaining resource
Laggard Tasks Most	The straggler tasks are given the remaining resources

In this paper, we compare the FCFS and capacity scheduler with the proposed algorithm two pass scheduling policy based resource allocation. The FCFS and capacity schedulers are still the default scheduling policies in Hadoop systems including the latest version -Hadoop Yarn.

3. Two Pass Scheduling Policy based Resource allocation

The algorithms and scheduling policies discussed in the previous section has been thoroughly analysed and a new technique named Two pass scheduling policy based scheduling technique is being formulated. The MapReduce simulation requires workload and workers. The layout and nature of workload is discussed in detail in the following sections.

3.1 MapReduce workload generation

The real world MapReduce workload consists of a variety of files including text, log files, scripts and different file formats. The procedure for creating a MapReduce workload for simulating MapReduce has already been discussed by various researchers. A number of metrics such as wait time, run time, response time, and number of running jobs play a vital role in MapReduce workload generation[13].

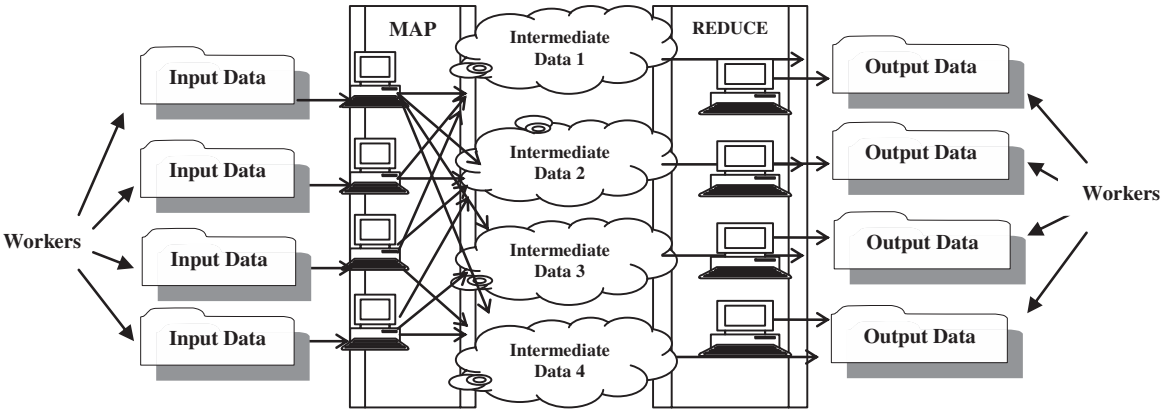

Fig 3. Matlab simulation of MapReduce

In this paper, we are using a simulated work load of textual data which consists of numbers, alphabets and special characters. We make use of six matlab workers in the parallel computing toolbox of matlab and a generic cluster was created which will mimic the MapReduce framework. Workload for this algorithm ranges from one GB to five GB of data. The data consists of only text files.

3.2 Matlab and MapReduce

There are various simulators which mimic the MapReduce framework. Some of the simulators which are used extensively in MapReduce research are MRPerf, MRSim and Mumak. The parallel computing toolbox and

distributing computing server can be utilized in creating a MapReduce environment which consists of worker nodes which acts as mappers and reducers. Figure 3 clearly illustrates the simulated environment in matlab. This environment is created on top of a generic cluster having twelve worker nodes.

3.3 Algorithm

The proposed algorithm two pass scheduling policy based resource allocation is as shown below along with Table 2 which describes the acronyms used in the algorithm.

Table 2: Explanation of Acronyms used

Variable Name	Explanation
UN	set of unscheduled tasks
FMS	set of free idle map slots
TS	A matrix of temporary list which copies the unscheduled tasks
C	|UN| X |FMS| matrix
LD_i	The maximum capacity of the container
UNM	Minimum execution time of unscheduled tasks
ER_t	Expected Running time
R_t	Running time

Pass -1

1) The scheduling queue is stored as a list and a copy of it is saved as TS
2) The minimum execution time is calculated from the size of file for each task(sent along with heartbeat)(UNM).
3) The possible execution time and FMS list are concatenated (which is named as matrix C).
4) UNM and FMS are sorted in descending order.
5) The first task from the list is taken and compared and assigned to FMS_i.
6) Delete the task UNM_i from the list.
7) Repeat the steps 5 and 6 until the UNM_i list is empty.

Pass-2

1) The schedule list TS is taken.
2) The worker node with maximum load is selected
    a. Check the ready time with UNM_i.
    b. If $ER_t < R_t$ and UNM_i not equal to first node in UNM list
        i. Reassign the task remaining in the node to i-1th worker node.
    Else
        ii. Assign the job to the same node and keep it unchanged.
        iii. Delete the node from TS list.
        iv. Move to the next node.
3) Repeat step 2 until TS list is empty.

The algorithm consists of two phases namely pass-1 and pass-2. The initial phase develops a scheduling queue list and another copy of the same is stored as a temporary queue list. The minimum execution time of the task is calculated from the file size of the workload. The parameters for file size checking are set manually according to the type of file that is being handled. In this implementation, we work only with text files and the file size is calculated by a matlab user defined function.

The number of free matlab workers is found and is added on to a list called Free Map Slot list (FMS). The minimum execution time list and free map slot list are concatenated on to a single matrix. The first task (which will be the biggest in size) will be compared to the FMS list. If free slots are available, the task will be sent to that worker for execution. Once the task is sent to the worker the UNM list is updated by deleting that task from the list. This process is continued till the UNM list is empty. The empty UNM list will trigger the next phase pass-2 of the algorithm.

In pass-2 of the algorithm, the Temporary list which was created in the phase1 stage of the algorithm is taken into consideration. From the temporary list, the worker with maximum workload is taken and heartbeat is checked for

getting information. The expected running time which was calculated in phase1 is compared with the current running time and the task, if required, is reassigned to the $(i-1)^{th}$ worker node. If the reassigning is not required then the job is kept in the same worker node and the node is deleted from the temporary list.
The algorithm is tested with various workloads and different number of workers and is compared with two algorithms namely FCFS and capacity scheduling.

4. Results and Discussion

The TPSP algorithm was implemented using matlab with six workers acting as mapper and reducer. The algorithm is compared to existing algorithms FCFS and capacity scheduling. The workload and number of workers are varied in different combinations to test the efficiency of the TPSP algorithm

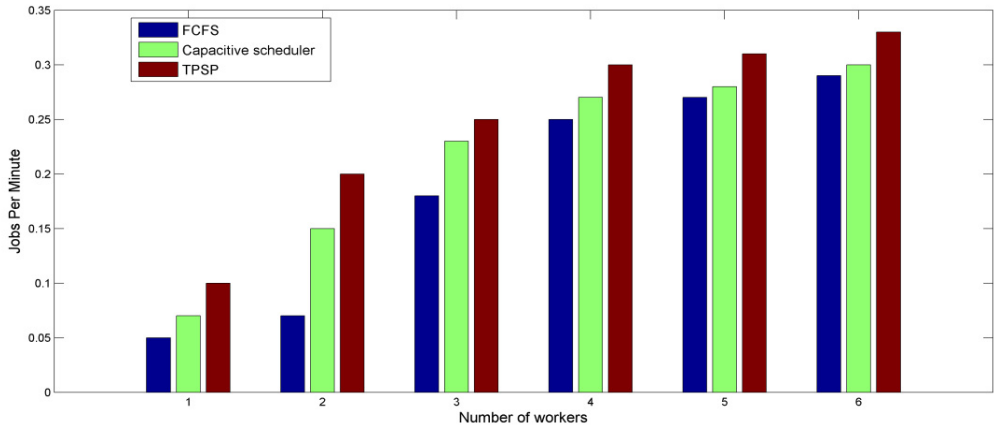

Fig 4: Throughput of the TPSP algorithm compared with FCFS and capacitive

Figure 4 depicts the throughput of the algorithm compared with the default algorithms namely Capacity scheduling and FCFS. Throughput can be defined as the number of jobs processed and completed per minute by the scheduler. The throughput was tested with a fixed workload and varying number of workers. The execution time of six

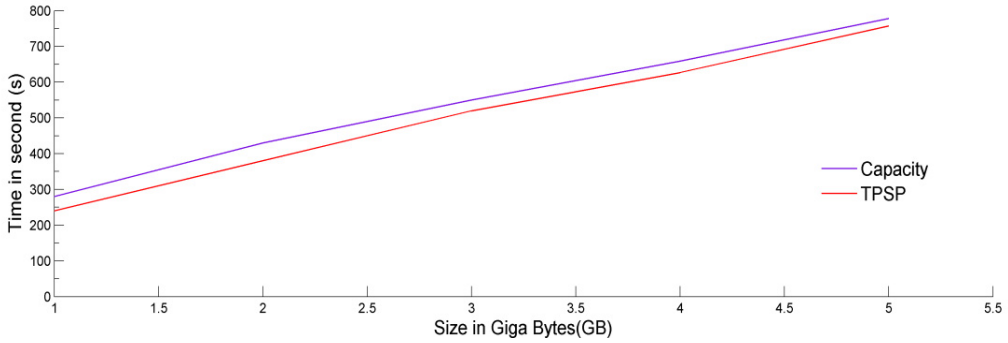

Fig 5: Comparison of execution time of six workers with varying workload

workers with varying workloads ranging from one GB to five GB is tested using TPSP and capacity scheduling algorithm and plotted as in Figure 5. The graph shows that TPSP is way ahead in execution speed when compared with capacity scheduling. The algorithm is executed with varying number of workers under a constant workload to obtain Figure 6. It shows the execution time compared with FCFS and capacity scheduler. The number of workers was changed from one to six. The algorithm is executed with varying number of workers under a constant workload to obtain Figure 6. It shows the execution time compared with FCFS and capacity scheduler.

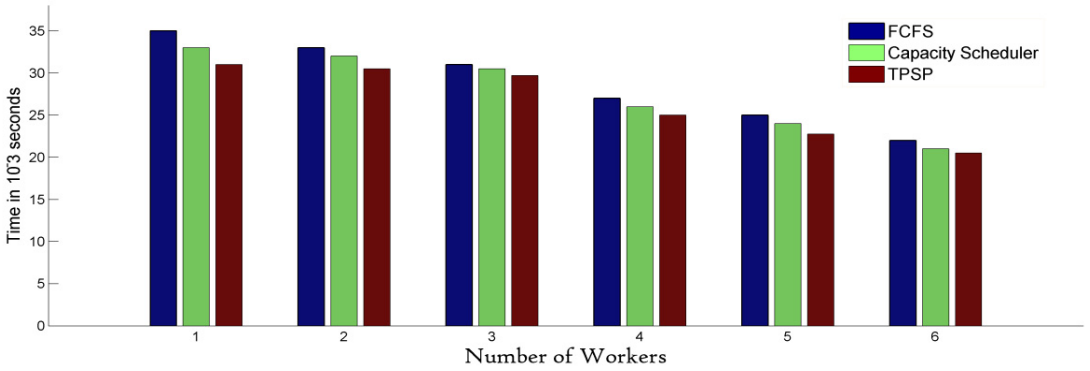

Fig 6: Comparison of Execution time with varying number of workers and a fixed workload

The number of workers was changed from one to six. The results depict the efficiency of TSTP algorithm which was tested for job throughput, execution time and varying workloads.

5. Conclusion

Focusing on MapReduce scheduling, this paper, through a survey of different MapReduce scheduling policies proposes a new technique which is more efficient than the default schedulers in hadoop. Big Data is definitely the future of parallel data processing and MapReduce framework will be an added advantage for processing gigantic amount of data within a short period of time. The efficiency of parallel data processing relies heavily on the type of scheduler that the parallel system is using. The efficiency of the algorithm has been proved by comparing it with different workloads, varying number of workers and with other algorithms. The algorithm which has been implemented and tested in a simulated environment, can be implemented in real hadoop systems as a future project.

References

1. Ebin Deni Raj, L.D.Dhinesh Babu, Ezendu. Ariwa, M.Nirmala .and P.V Krishna. *Forecasting the Trends in Cloud Computing and its Impact on Future IT Business* .In Green Technology Applications for Enterprise and Academic Innovation *on* pages 14-32, IGI Global Publishers, 2014
2. Lin, Jimmy, and Chris Dyer. *Data-intensive text processing with MapReduce*. In *Synthesis Lectures on Human Language Technologies* 3, pages 1-177.2010.
3. Dean, Jeffrey, and Sanjay Ghemawat. *MapReduce: simplified data processing on large clusters*. In *Communications of the ACM* 51. no. 1, pages 107-113.2008.
4. Odlyzko, Andrew M. *Internet traffic growth: Sources and implications.* In *ITCom* , pages. 1-15. International Society for Optics and Photonics, 2003.
5. Bourret, Ronald. *XML and Databases*. 1999.
6. Abadi, Daniel J. *Data Management in the Cloud: Limitations and Opportunities* .In *IEEE Data Eng. Bull.* 32, no. 1 pages 3-12 2009.
7. Olston, Christopher, Benjamin Reed, Utkarsh Srivastava, Ravi Kumar, and Andrew Tomkins. *Pig latin: a not-so-foreign language for data processing*. In*Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1099-1110. ACM, 2008.
8. http://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html. Accessed :2014-08-10
9. Guo, Zhenhua, Geoffrey Fox, Mo Zhou, and Yang Ruan. *Improving Resource Utilization in MapReduce* . In *Cluster Computing (CLUSTER), 2012 IEEE International Conference on*, pages. 402-410. IEEE, 2012.
10. Dhinesh Babu, L. D., Angappa Gunasekaran, and P. Venkata Krishna. *A decision-based pre-emptive fair scheduling strategy to process cloud computing work-flows for sustainable enterprise management*. In International Journal of Business Information Systems 16, no. 4 pages: 409-430 . Inderscience publications.2014.
11. Babu, LD Dhinesh, and P. Venkata Krishna. *Versatile time-cost algorithm (VTCA) for scheduling non-preemptive tasks of time critical workflows in cloud computing systems*. In International Journal of Communication Networks and Distributed Systems 11, no. 4 pages: 390-411 .Inderscience publications .2013.
12. L.D.D Babu and P.V Krishna,. *Honey bee behavior inspired load balancing of tasks in cloud computing environments*. In Applied Soft Computing 13,pages 2292-2303 .2013.
13. http://www.pds.ewi.tudelft.nl/~iosup/msc/2012_msc-thesis_tud_ThomasDeRuiter.pdf. Accessed :2014-07-14