Efficient load aware scheduler for map reduce applications in cloud environment

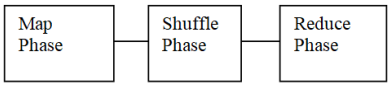
A.Sree Lakshmi[1], M.BalRaju[2], N.Subhash Chandra[3]
[1]Geethanjali college of Engineering and Technology, Hyderabad, Telangana, India,
[2]Krishna Murthy Institute of Technology, Hyderabad, Telangana, India,
[3]CVR college of Engineering , Hyderabad, Telangana, India

Most of the current day applications are compute and data intensive which laid a platform for invention of technologies like Hadoop. Hadoop uses a Map Reduce paradigm to solve the problem by using parallelism. Cloud computing environments have provided more flexibility in using Hadoop to solve Big Data problems without any investment on infrastructure procurement and maintenance and take the advantage of parallelism with required scalability. The current schedulers of Map Reduce tasks can be improved for virtual environments to reduce the cost of the services used in the cloud. This work proposes an efficient scheduler for Map reduce applications in cloud environment.

Keywords: Cloud Computing, Hadoop, Map Reduce, Virtual machine, HDFS, Scheduler

1. INTRODUCTION

Parallelism of tasks is one of the biggest achievements in the computing field. From the invention of MPI to current day Map Reduce framework there has been a different way of utilization of parallelism to improve the performance of the system. Multi core architectures have paved the way to utilize the parallelism to next level. Hadoop is the framework used to solve problems involving large amount of data using automatic parallelism[1] in the form of map and reduce tasks. Hadoop has a good advantage of moving computation towards data rather than normal strategy of moving data towards computation. The input data is copied into Hadoop Distributed File System(HDFS)[4] with a replication factor of 3 by default. The input data is divided into multiple chunks of fixed size(by default 64 MB in Hadoop1 and 128MB in Hadoop YARN. Hadoop execution life cycle has three phases Map Phase, Shuffle phase and reduce phase .



. Fig. 1: Phases of Map Reduce framework in Hadoop

The Hadoop framework automatically divides the given problem into multiple tasks and executes multiple map tasks parallely on multiple nodes. The intermediate results from map tasks are shuffled into different buckets which are hashed to different reducers. Each reducer processes the intermediate output from multiple map tasks and the output from the reducers tasks are written back into HDFS.

2. SCHEDULERS IN HADOOP YARN

Hadoop has built in three types of schedulers which are FIFO scheduler, Fair Scheduler, Capacity Scheduler [2]. Hadoop provides scheduler as a pluggable component which enables users to specify scheduler as per their requirements. The Resource Manager manages and allocates resources to applications in a cluster. Scheduler is a part of Resource manager that handles the scheduling policies of Hadoop cluster. User applications are placed in Queues which allows multiple users

to share the cluster. Hierarchical queues are used to decide about the sharing of resources in the cluster. FIFO Scheduler schedules the jobs in First in first out order of their submission time. Fair Scheduler shares the resources fairly to all the jobs running on cluster when only one job is present. It occupies entire cluster when only one job is present. As other jobs are submitted every job gets equal share of resources. Capacity scheduler allows sharing resources of large cluster giving each organization a minimum guarantee of capacity.

3. EXISTING WORK

Apart from Hadoop package available schedulers many researchers worked on designing different schedulers for Hadoop with different requirements. Delay scheduler, Dominant Fairness scheduler, Adaptive scheduler are some parts of works done for designing schedulers for Hadoop clusters [8]. Delay scheduler enables a task to wait for specified time until it gets a chance to execute on a node which has its data. Delay scheduler tries to optimize the time by reducing the time required to transfer the input data to the node as the node is non local to that input split. Dominant Fairness scheduler categorizes jobs as CPU dominant or mem ory dominant based on resource requirements of the job. Number of tasks allocated to run concurrently for each job is designed to satisfy the dominant resources of the jobs. Adaptive scheduler was designed to use the performance of jobs to adjust the amount of resources. It studies about the behavior of jobs and uses that information in deciding amount of resources that can be given for that job. Apart from these standardized works there are many researches on scheduling mechanism like HaSTE[19], EHadoop[9], SAMR[11], HScheduler[17] by different researchers . Different schedulers are being designed to optimize the runtime/cost of the map reduce jobs in heterogeneous environments, cloud environments [3, 5, 10, 11, 13, 14, 16].

4. PROPOSED WORK

Cloud Technologies have been commonly used now days due to its advantages like easy to use infrastructure, no requirement of skills to maintain and use infrastructure, quick deployment of required number of resources , scalability, Fault tolerant etc., As Hadoop is a framework which needs more number of nodes to run parallel to take advantage of its design. With cloud computing environments, now users need not incur costs to buy the required infrastructure and maintain it, but can use the services provided by cloud to establish a hadoop cluster very quickly and need not worry about the number of resources e.g Amazon EMR provides a very flexible way of establishing a Hadoop cluster and execute Hadoop jobs. It also provides Ganglia Resource monitoring tool which enables users to have a complete picture of their usage and run time statistics. Out previous work [1] discussed about the study of Amazon EMR in executing Hadoop jobs. Our work projected that run time of a task is affected by the performance of virtual machine as many number of virtual machines will be created in a particular node in cloud virtualized environment. Our proposed work is to design a Hadoop scheduler that suits the cloud environment, where the scheduling of map and reduce tasks is being done in consideration with the characteristics of the virtual machines and the task. This enables a proper assignment of tasks to different containers scheduled on virtual machines in such a way that the total execution time of all the tasks is minimized. Let M be the number of map tasks and R be the number of reduce tasks of a particular job.

No of mappers= file size/input split size.

If 3 machines are rented in the cloud, then the M map tasks are to be distributed to 3 machines in such a way that the total execution time is minimum. Let $t < m_i, H_i >$ denote the time taken by map tasks m_i on machine H_i. Let $n < m, H_i >$ denote number of map tasks scheduled on machine Hi. Then average time taken by $n < m, Hi >$ tasks on machine H_1

$$\sum_{i=1}^{n<m_i,H_1>} \frac{t < m_i, H_1 >}{n < m, H_1 >}$$

Similarly, average time taken by $n < m, Hi >$ are reducer tasks on machine H1 is

$$\sum_{i=1}^{n<r_i,H_1>} \frac{t < r_i, H_1 >}{n < r, H_1 >}$$

The goal is to minimize the overall execution time on 3 nodes

$$Min[\sum_{j=1}^{3}\{ \sum_{i=1}^{n<m_i,H_1>} \frac{t < m_i, H_1 >}{n < m, H_1 >} + \sum_{i=1}^{n<r_i,H_1>} \frac{t < r_i, H_1 >}{n < r, H_1 >}\}]$$

In cloud environment $t < mi, Hj >$ or $t < ri, Hj >$ also depends on the VM it is being deployed along with the job. When multiple jobs are executed on the cluster, If proper scheduling of tasks can be done while creating containers on virtual machines then an optimized execution time can be obtained. In cloud environment usually we pay on hourly basis, execution time plays a crucial role in the cost and deadline. To provide an efficient scheduling we categorize job as CPU intensive and IO intensive depending on the statistics of the execution of map tasks. Virtual machines can also be categorized as CPU heavy or IO heavy machines depending on the current CPU and IO space utilization of the host on which the virtual machine is deployed. As the cloud environment is virtualized environment, multiple virtual machines are placed on a single host.

When scheduling a task which is IO intensive if a virtual machine which is less IO heavy is choosen or for a task which is CPU intensive if a virtual machine with less CPU utilization is choosen in in consideration with data locality then the overall execution time of all the tasks of multiple jobs can be optimized. A host which has a virtual machine executing a CPU intensive task is optimal for other task which is IO intensive. This strategy would be optimal if the huge jobs which has many map tasks so that the knowledge gained by executing the initial map tasks enable to tag the tasks belonging to that job as either CPU intensive or IO intensive. This knowledge can be applied for scheduling the remaining tasks effectively.

To design a scheduler that can schedule the map reduce tasks based on job characteristics and VM characteristics in Hadoop framework does not increase time complexity. NodeManager can specify the average CPU usage and amount of IO of the task running on it in its heartbeat. ResourceManager can decide on scheduling the jobs on the nodes so that jobs of different usage can be scheduled on a node to make efficient utilization of the cloud resources.

Job are categorized to CPU-intensive, IO-intensive and every job is associated with a 2 digit tag which indicates the intensiveness of the job [15]. MSB indicates the CPU intensiveness and LSB denotes IO intensiveness. Tag with value 01 indicates a job which is IO-intensive and value 10 indicates a job which is CPU-intensive. Tag value of 00 indicates an equal weight of CPU intensiveness and IO- intensiveness.Similarly 2 bit tag is used for virtual machines to indicate the current load of CPU and IO of the host on which the virtual machine id deployed. Submission of cloudlets is decided with reference to the tag of jobs and virtual machines. Job can be assigned a tag by user if the behavior of the task is known or can be identified after few map tasks get executed as defined in algorithm3. When a virtual machine tag indicates as CPU heavy then a task which is less CPU-intensive and more IO-intensive would be appropriate choice. Load aware scheduler is designed to submit the jobs in consideration with the job characteristics and VM characteristics where the submit cloudlets is modified to schedule the cloudlets to VM as per their characteristics.

Algorithm1 LoadAwareScheduler()

```
1.  n= machines specified from request in simulation.properties
2.  create n Virtual Machines in the selected datacenter where the mtype is available
3.  while cloudletList.size()>0 do
4.     submitCLoudlet() //based on Vmtag and job tag: Algorithm2
       for each vm: vmList
5.        process the cloudlet submitted to VM
6.        update cloudlet processing
7.        check cloudlet completion
8.        cloudletReturn() //Algorithm3
9.        for each finished cloudlet add a new one from the waiting list
10.       update cloudlet execution statistics like execution time, finish time
11.    end for
12. end while
13. close datacenters
14. print job execution statistics
15. end
```

SubmitCloudlet() algorithm identifies an appropriate task for each VM and assigns them to it.The decision is based on the characteristics of job and the virtual machine.

Algorithm2 submitCloudlet()

```
1.  if cloudletList.size()=0
2.    exit;
3.  end if
4.  for each vm in VmList
5.     boolean done=false;
6.     for each cloudlet in cloudletList
7.        if(cloudlet instanceof ReduceTask)
8.           if(allMapTasksFinished(cloudlet)
9.              Send cloudlet to VM;
10.             break;
11.          else
12.             continue;
13.          end if
14.       end if
15.       if(cloudlet instanceof MapTask)
16.          id=cloudlet.getId();
17.          Request r=getRequestFromTaskId(id);
18.          if(r.job.tag & vm.tag =0)
19.             C=cloudlet;
20.             done=true;
21.             break;
22.          end if
23.       end if
24.    end for
25.    if(!done & CloudletList.size()!=0)
26.       C=cloudletList.get(0);
27.    end if
28.    Cloudlet.setVmId(Vm.getId());
29.    bind cloudlet C to VM;
30.    add cloudlet C to cloudletsubmittedList;
31.    remove cloudlet C from cloudletList;
32.    send event CLOUDLET_SUBMIT to datacenter;
33. end for
34. end
```

cloudletReturn() algorithm is being implemented to analyze the characteristic of Job so that it can be used in scheduling decisions of remaining map tasks of the job.

Algorithm3 cloudletReturn()

```
1.  id=cloudlet.getId();
2.  Request r=getRequestFromTaskId(id);
3.  if(cloudlet  instanceof MapTask)
4.      CPUT=t.getCPUUtilizationTime();
5.      TET=t.getTotalTimeOfExecution();
6.      CPUR=CPUT/TET;
7.      if CPUR is near to 1 then
8.          j.setTag(CPU_HEAVY);
9.      else if CPUR is near to 0 then
10.         j.setTag(IO_HEAVY);
11.     else
12.         j.setTag(NORM);
13.     endif
14. endif
15. end
```

5. EVALUATION

To simulate the proposed algorithm we used CloudSimEx simulator [6] which is an extension of CloudSim simulator in support of Map Reduce Simulation. CloudSim[7] supports server virtualization. It enables creation of multiple datacenters. VM allocation policies are used to assign a virtual machine to a host. VM allocation policies are defined for each data center. VM scheduler policies are used to allocate resources to VMs by each host. Cloudlet scheduler policies are defined for particular VM which manages the cloud application jobs internally. Cloudlet is any job/task/request to be executed.

CloudsimEx is completely event driven where all the simulation steps are handled by events placed in two different types of queues: Future Queue and Deferred Queue. Future Queue holds all the events when generated and later are moved to Deferred queue based on the source and destination of the events.

CloudSimEx is being modified to suit the Hadoop environment of map reduce programming. The following changes are being made to simulate CloudSimEx for the implementation of Load Aware Map Reduce scheduling algorithm

1. Cloud.yaml is changed to include only public data centers 2. Number of machine instances as per the user request instead of all machines available in datacenter 3. Allocation of Vmtype based on user request 4. To include locality info of the tasks input data 5. Decision Process in binding of cloudlet to Vm 6. Simulation.properties to include no of machines and machine types as requested by user.

Simulation.properties:

cloud.file=Cloud.yaml

experiment.files=test5.yaml

machines=<< No of machines used is 4,5,6,7,8>>

mtype=large-aws-us-east-1

jobs used :

MapReduce_9_2.yaml(10000millioninstructions)

MapReduce_15_2.yaml(100000million instructions)

MapReduce_30_2.yaml(100000million instructions)

MapReduce_50_1.yaml(10000million instructions)

The above proposed algorithm is implemented in CloudSimEx simulator and the results indicate that proper binding of cloudlet to VM based on their run time characteristics decreases the execution time. The execution time plays a key role in cloud environment as user needs to pay on hourly basis. The algorithm is tested with 2 test cases and each test case with number of machines as 4, 5, 6, 7, 8. In each test case three jobs are submitted and the total VM processing time and Cost of each VM is studied. The final Execution time of the cluster with n machines to complete both the jobs is considered as the maximum amount of time used by the VMs in the cluster and similarly the cost.

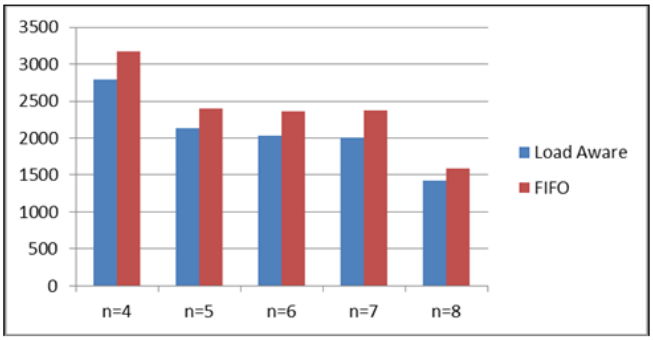
CASE 1:

test5.yaml

```
!!org.cloudbus.cloudsim.ex.mapreduce.Experiment
workloads:
 - !!org.cloudbus.cloudsim.ex.mapreduce.Workload
   [
    LoadAware, Public,
     {
       GOLD: 100.0,
       SILVER: 60.0,
       BRONZE: 0.0
     },
    [
     #[Submission Time, Deadline, Budget, Job, user class]
     !!org.cloudbus.cloudsim.ex.mapreduce.models.request.Request
     [200000, 120, 2.5, MapReduce_50_1.yaml, GOLD],
 !!org.cloudbus.cloudsim.ex.mapreduce.models.request.Request
     [200000, 120, 2.5, MapReduce_15_2.yaml, GOLD],
!!org.cloudbus.cloudsim.ex.mapreduce.models.request.Request
     [200000, 120, 2.5, MapReduce_9_2.yaml, GOLD]
   ]  ]
```

. $Table1 : Executiontimes for Mapreducejobs : 50_1, 15_2, 9_2$

No of machines	Algorithm	Job1 Finishing Time (sec)	Job2 Finishing time(Sec)	Job3 Finishing time(Sec)	Total Cloud Execution time(Secs)
4	Load Aware	2749.12	2692.62	2786.62	2786.62
	FIFO	227.62	3134.32	3176.02	3176.02
5	Load Aware	2098.27	2037.32	2128.27	2128.27
	FIFO	205.12	2372.32	2399.38	2399.38
6	Load Aware	2000.12	2007.32	2026.93	2026.93
	FIFO	177.62	2335.26	2363.76	2363.76
7	Load Aware	1985.12	1880.12	2007.62	2007.62
	FIFO	165.12	2322.69	2377.69	2377.69
8	Load Aware	1414.69	1352.32	1429.69	1429.69
	FIFO	152.62	1562.62	1589.38	1589.38



. $Fig.2 : Comparison of FIFO and Load Aware scheduling for Mapreduce jobs : 50_1, 15_2, 9_2$

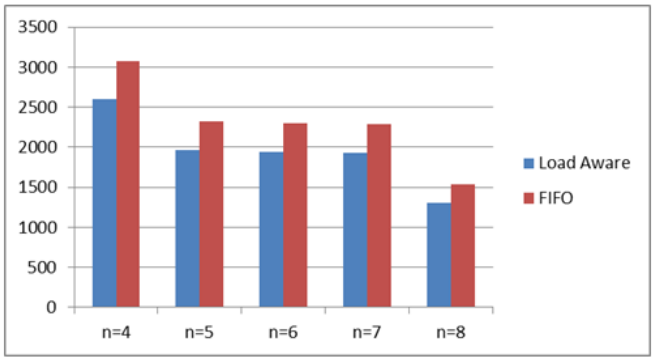
CASE 2:
test5.yaml

```
!!org.cloudbus.cloudsim.ex.mapreduce.Experiment
workloads:
 - !!org.cloudbus.cloudsim.ex.mapreduce.Workload
   [
    LoadAware, Public,
     {
       GOLD: 100.0,
       SILVER: 60.0,
       BRONZE: 0.0
     },
    [
     #[Submission Time, Deadline, Budget, Job, user class]
!!org.cloudbus.cloudsim.ex.mapreduce.models.request.Request
     [200000, 120, 2.5, MapReduce_30_2.yaml, GOLD]
     !!org.cloudbus.cloudsim.ex.mapreduce.models.request.Request
     [200000, 120, 2.5, MapReduce_15_2.yaml, GOLD],
 !!org.cloudbus.cloudsim.ex.mapreduce.models.request.Request
     [200000, 120, 2.5, MapReduce_50_1.yaml, GOLD]
    ]
   ]
```

.

. $Table 2 : Execution times for Map reduce jobs : 30_2, 15_2, 50_1$

No of machines	Algorithm	Job1 Finishing Time (sec)	Job2 Finishing time(Sec)	Job3Finishing time(Sec)	Total Cloud Execution time(Secs)
4	Load Aware	2542.32	2542	2602.03	2602.03
	FIFO	42.62	3016.97	3075.01	3075.01
5	Load Aware	1910.06	1915.83	1960.54	1960.54
	FIFO	31.61	2281.02	2325.74	2325.74
6	Load Aware	1902.32	1902.32	1939.69	1939.69
	FIFO	30.11	2256.24	2293.89	2293.89
7	Load Aware	1895.12	1880.12	1932.62	1932.62
	FIFO	30.09	2257.73	2287.73	2287.73
8	Load Aware	1277.32	1277.32	1307.01	1307.01
	FIFO	30.01	1510.48	1538.51	1538.51



. $Fig.3 : Comparison of FIFO and Load Aware scheduling for Map reduce jobs : 30_2, 15_2, 50_1$

The evaluation of the proposed Load Aware scheduling algorithm indicate an improvement of 14 to 17 percent in the makespan of the jobs submitted to Hadoop cluster on cloud. It is observed that though finish time of job1 is less in FIFO, the load aware algorithm schedules the task in such a way that the makespan is minimized so that the rent paid for cloud usage would be less. This algorithm is more advantages when large Hadoop jobs are submitted to Hadoop cluster on cloud .It would be only helpful in the situations where makespan of jobs is important so that cloud usage cost is minimized.

6. CONCLUSION

In this paper we evaluated the performance of map reduce scheduler in cloud environments considering the current load on the virtual machine and the usage characteristics of jobs under execution. This work helps in reducing the execution time of the map reduce jobs when executed in virtualized environment especially by reducing the cost to be paid for the cloud infrastructure usage. Map reduce framework is being currently used very commonly which enables parallel execution of map and reduce tasks. This algorithm enables the parallel execution of map reduce tasks to be optimized based on the virtual environment provided by the cloud provider to the user. This algorithm would be more helpful if the cloud users are charged on minutes basis instead of hourly basis.

7. REFERENCES

1. A.Sree Lakshmi , Dr.M.BalRaju , Dr.N.Subhash Chandra,Towards optimization of hadoop map reduce jobs on cloud IEEE International Conference on Computing, Analytics and Security Trends (CAST 2016), Dec 2016, ISBN: 978-1-5090-1338-8.

2. A.Sree Lakshmi, Dr.M.BalRaju, Dr.N.SubhashChandra : Scheduling of Parallel Applications Using Map Reduce On Cloud: A Literature Survey, International Journal of Computer Science and Information Technologies, Vol. 6 (1) , 2015, Page(s): 112-115.

3. Andrew Wylie, Wei Shi, Jean-Pierre Corriveau A Scheduling Algorithm for Hadoop MapReduceWorkflows with Budget Constraints in the Heterogeneous Cloud, 2016 IEEE International Parallel and Distributed Processing Symposium Workshops.

4. Hadoop Distributed File System:$http://hadoop.apache.org/docs/r1.2.1/hdfs_design.html$.

5. H. Kang, Y. Chen, J. L. Wong, R. Sion, and J. Wu, Enhancement of Xens scheduler for MapReduce workloads, in Proceedings of the 20th international symposium on High performance distributed computing, New York, NY, USA, 2011, pp. 251262.

6. http://nikgrozev.com/2014/06/08/cloudsim-and-cloudsimex-part-1/.

7. http://www.cloudbus.org/cloudsim/.

8. https://hadoop.apache.org/docs/r2.7.1/hadoop-yarn/hadoop-yarn-site/.

9. Lenar Yazdanov, Maxim Gorbunov, Christof Fetzer EHadoop: network I/O aware scheduler for elastic MapReduce cluster , 2015 IEEE 8th International Conference on Cloud Computing, DOI 10.1109/CLOUD.2015.113.

10. Moussa Ehsan, Karthiek Chandrasekaran, Yao Chen, Radu Sion , Cost-efficient tasks and data co-scheduling with affordhadoop , IEEE transactions on cloud computing, april 2016, , doi 10.1109/tcc.2017.2702661.

11. Quan Chen, Daqiang Zhang, Minyi Guo, Qianni Deng, and Song Guo. Samr: A self-adaptive mapreduce scheduling algorithm in heterogeneous environment. Computer and Information Technology, International Conference, 0:2736 2743, 2010.

12. Qutaibah Althebyan,Yaser Jararweh, Qussai Yaseen, Omar AlQudah and Mahmoud Al-Ayyoub Evaluating map reduce tasks scheduling algorithms over cloud computing infrastructure, Concurrency and Computation: practice and experience 2015; 27:56865699 ,31 July 2015 , Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/cpe.3595.

13. Rajdeep Das, Rohit Pratap Singh, Ripon Patgiri ,MAPREDUCE SCHEDULER: A 360-DEGREE VIEW, International journal of current engineering and scientific research (IJCESR), ,issn (print): 2393-8374, (online): 2394-0697, volume-3, issue-11, 2016.

14. Sewoog Kim, Dongwoo Kang and Jongmoo Choi I/O Characteristics and Implications of Big Data Processing on Virtualized Environments, Applied Mathematics and Information Sciences An International Journal, 9, No. 2L, 591-598 (2015).

15. Shyam Deshmukh, Dr. J. V. Aghav, Rohan Chakravarthy Job Classification for MapReduce Scheduler in Heterogeneous Environment, 2013 International Conference on Cloud and Ubiquitous Computing and Emerging Technologies.

16. S. Kim, D. Kang, J. Choi and J. Kim, Burstiness-aware I/O scheduler for MapReduce framework on virtualized environments", 2014 International Conference on Big Data and Smart Computing (BIGCOMP), 2014, Pages: 305- 308, doi:10.1109/BIGCOMP.2014..

17. W. Tian, G. Li, W. Yang and R. Buyya,"HScheduler: an optimal approach tominimize the makespan of multiple MapReduce jobs", The Journal of Supercomputing June 2016,Volume 72, Issue 6, pp 23762393, doi:10.1007/s11227-016-1737-4.

18. X. Wang, D. Shen, G. Yu, T. Nie and Y.Kou, "A ThroughputDriven Task Scheduler for Improving MapReduce Performance in Job-Intensive Environments", 2013 IEEE International Congress on Big Data, 2013, Pages: 211 -218.

19. Yi Yao,Jiayin Wang, Bo Sheng, Jason Lin, Ningfang Mi , HaSTE: Hadoop YARN Scheduling Based on Task-Dependency and Resource-Demand, 2014 IEEE International Conference on Cloud Computing , 978-1-4799-5063-8/14

A.Sree Lakshmi is a research scholar in the area of big data processing on cloud under Computer Science and Engineering of JNT University, Hyderabad, India. She is working as an Associate Professor in Computer Science and Engineering department of Geethanjali College of Engineering and Technology, Hyderabad. Her research interests are Distributed computing, Cloud Computing and Big Data Analytics.

Dr.M.BalRaju received M.Tech (CSE) from Osmania University and Ph.D from Jawaharlal Nehru Technological University, Hyderabad in 2010. Currently working as Professor and Principal in Swamy Vivekananda Institute of Technology, Hyderabad, India. His area of specialization includes Data Base, Data Mining, and Image Processing.

Dr. N. Subhash Chandra received his Ph.D in Computer Science and Engineering from JNT University, Hyderabad, India in 2010. He is a Professor of Computer Science and Engineering at the CVR College of Engineering, Hyderabad. His research interests are Image processing, Data Mining and Cloud Computing. He has published more than 50 research papers in journals and International conferences. He received best paper award in international conference.