International Journal of Research In Science & Engineering e-ISSN: 2394-8299
Volume: 3 Issue: 5 Sep.-Oct. 2017 p-ISSN: 2394-8280

JOB SCHEDULING SCHEME USING MAPREDUCE JOB

Miss Jadhav Usha Dattatraya[1],Prof V,R Chirchi[2]

PG Student, PG Department of CSIT , MBES Engg College,ushajadhav535@yahoo.com

Assistant Professor , PG Deparment of CSIT,MBES Engg College ,vr.chirchi@gmail.com

ABSTRACT

MapReduce is a framework using which it can write applications to process huge amounts of data, in parallel, on large clusters of commodity hardware in a reliable manner, MapReduce algorithm contains two important tasks, namely Map and Reduce., the hybrid job driven scheduling(JOSS) scheme i.e JOSS is used job level scheduling ,Map task level scheduling and reduce task level scheduling ,JOSS classifies MapReduce jobs based on job scale and job type and designs an appropriate scheduling polices to schedule each class of jobs , JOSS has two variations i.e JOSS-T & JOSS-J are proposed two achieve speeding up task assignment and further increasing the VPS-locality respectively ,the five of MapReduce benchmarks used to create two different MapReduce workload for comparing JOSS-T and JOSS-J with three know scheduling algorithm supported by Hadoop .the results show that the two variations outperform the tested algorithm in terms of map-data locality and reduce data locality & Comparison processing time of JOSS with TTA and JOSS with JTA ,the main aim of is to improve data locality for both map task , reduce task ,avoid job starvation ,improve job execution performance.

Keywords:-MapReduce , Hadoop , Map–task Scheduling , Reduce -task scheduling , MapReduce , MapReduce Workload.

---

1. INTRODUCTION

The MapReduce algorithm contains two important tasks, namely Map and Reduce. Map takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs). Secondly, reduce task, which takes the output from a map as an input and combines those data tuples into a smaller set of tuples , as the sequence of the name MapReduce implies, the reduce task is always performed after the map job. the advantage of MapReduce is that it is easy to scale data processing over multiple computing nodes, under the MapReduce model, the data processing primitives are called mappers and reducers, In this paper concentrated on a virtual MapReduce cluster , MapReduce cluster can only be classified as three levels as VPS-locality means that a map task and its input data are co-located at the same VPS,Cen-locality means that a map task and its input are within the same datacenter, but not at the same VPS, off-Cen, means that a map task and its input are located at different datacenters.,the hybrid job-driven scheduling scheme (JoSS for short) by providing scheduling in three levels: job, map task, and reduce task. JoSS classifies MapReduce jobs into either large or small jobs based on each job's input size to the average datacenter scale of the virtual MapReduce cluster, and further classifies small MapReduce jobs into either map-heavy or reduce-heavy based on the ratio between each job's reduce-input size and the job's map-input size. then JoSS uses a particular scheduling policy to schedule each class of jobs such that the corresponding network traffic generated during job execution can be reduced, and the corresponding job performance can be improved, the two variations of JoSS, named JoSS-T and JoSS-J, to guarantee a fast task assignment and to further increase the VPS-locality, respectively ,JoSS-T and JoSS-J is used to five MapReduce benchmarks to create two different MapReduce workloads for evaluating & comparing Joss-T & Joss-J .

Objectives:-

- JoSS Schedule Map- Reduce jobs in a virtual MapReduce cluster by addressing both map-data locality & reduce-data locality .
- The jobs classified into map-heavy and reduce heavy jobs , large job & designing the corresponding policies to schedule each class of jobs.
- JoSS increases data locality and improves job performance, avoids job starvation and improves job performance& the two variations of JoSS has respectively achieve two goals: speeding up task assignment and further increasing the VPS-locality .

2. LITERATURE REVIEW

Chao Tian et.al[1] it give a new view of the MapReduce model, and classify the MapReduce workloads into three categories based on their CPU and I/O utilization. With workload classification, we design a new dynamic MapReduce workload predict mechanism, MR-Predict, which detects the workload type on the fly this propose a Triple-Queue Scheduler based on the MR-Predict mechanism. The Triple-Queue scheduler could improve the usage of both CPU and disk I/O resources under heterogeneous workloads. And it could improve the Hadoop throughput by about 30% under heterogeneous Workloads.

Faraz Ahmad et .al[2]in this there are a total of 13 benchmarks, out of which Tera-Sort, word-Count, and Grep are from Hadoop distribution, the rest of the benchmarks were developed in-house and are currently not part of the Hadoop distribution. the three benchmarks from Hadoop distribution are also slightly modified to take number of reduce tasks as input from the user as well as to generate final time completion statistics of jobs.

Zhenhua Guo et.al[3] has investigate data locality in depth for data parallel systems among which GFS/MapReduce is representative and thus our main research target its advantage has the goodness of data locality is a significant advantage of data parallel systems over traditional HPC system also good data locality reduces cross switch network traffic one of the bottlenecks in data intensive computing it build a mathematical model of scheduling in MapReduce. **Jeffrey Dean and Sanjay Ghemawat[4]** has reviwed overall flow of MapReduce operation implementation in this MapReduce operation implementation has several advantage has the MapReduce programming model has been successfully used at google for many different purposes it attribute this success to several reasons first, the model is easy to use, even for programmers without experience with parallel and distributed systems, since it hides the details of parallelization, fault tolerance, locality optimization, and load balancing &, a large variety of problems are easily expressible as MapReduce computations.

Jin et.al[5] proposed the Balance Reduce algorithm which produces an initial task allocation for all map tasks allocation for all map task of a job and then takes network state and cluster workload into consideration to interactively adjust the task allocation to reduce job turnaround time.

Chen He et.al[6] has a new matchmaking algorithm is used to improve the data locality rate and the average response time of MapReduce cluster ,the matchmaking algorithm carried out experiments to compare not only MapReduce scheduling algorithm but also with an existing data locality enhancement technique and the experimental results demonstrate that matchmaking algorithm can often obtain the highest average response time for map tasks & to evaluate the matchmaking scheduling algorithm it compare Hadoop default FIFO scheduler & delay scheduling algorithm in this two metrics i.e map tasks data locality rate and average response time are used for evolution also avoid wasting computing resources the matchmaking algorithm will assign the node a non local task and the matchmaking algorithm achieves the high data locality rate & high cluster utilization .

Zaharia et.al[7] have developed a delay scheduling algorithm to impove the data locality rate to Hadoop cluster for this the delay scheduling algorithm a jobs execution is postponed to wait for a slave node that contains the jobs input data here the delay time D is a key parameter by default it is set 1.5 times the slave nodes heartbeat interval also to obtain the best performance for delay scheduling algorithm it have to choose an appropriate D value.

Moussa Ehsan,Radu Sion[8] introduce LiPS, a new cost-efficient data and task co-scheduler for MapReduce in a cloud environment & the advantages of LiPS is LiPS allows flexible control of job makespans , multi-resource management and fairness by using linear programming to simultaneously schedule data and tasks LiPS helps to achieve minimized dollar cost globally & LiPS suggest giveing priority to data placement in cloud task scheduling .

Jorda Polo et .al [9] In this address the problem by introducing a new task scheduler for a MapReduce framework that allows performance-driven management of MapReduce tasks the proposed task scheduler dynamically predicts the performance of concurrent MapReduce jobs and adjusts the resource allocation for the jobs. It allows applications to meet their performance objectives without over provisioning of physical resources.

3. JOB SCHEDULING SCHEME ALGORITHMS

The Job Scheduling scheme is used the task scheduler algorithm, TTA algorithm and JTA algorithm using this algorithms the scheduling algorithm JOSS-T and JOSS-J is implemented , JOSS-T is made combination of task scheduler and TTA this is Fast task assignment algorithm and JOSS-J is combination of task Scheduler and JTA this algorithm improves VPS locality of Map task and reduce task these all algorithm is used MapReduce technique.

3.1. The algorithm of the task scheduler :-

Input: J &input –data description ,Output : task –scheduling decision & the below task scheduler algorithm contains 37 execution steps.

Procedure:

1: Calculate a hash value for J's executable code and J's input data
2: type
3: Let H be a set of hash values previously generated by Joss;
4: **if** the hash value is not in H {
5: Append all map tasks of j to the end of $MQ_{FIFO;}$
6: Append all reduce tasks of J to the end of $RQ_{FIFO;}$
7: **else**{
8: **if** J is a small RH job{// used policy A
9: let cen_w be a datacenter having the least unprocessed
10: tasks among $cen_1, cen_{2....}$ $Cen_{k;}$
11: Append all map tasks of J to the end of $MQ_{w,0;}$
12: Append all reduce tasks of j to the end of $RQ_{w,0}$
13: **else** {
14: let L_c be a set of all unique input blocks of J held by cen_c
15: where c= 1,2,…k;
16: a=m:/*m is the number of map tasks of J */
17: **while** a>0{/* i.e. not all map task of j are scheduled.*/
18: Let L_d is the first largest set among $L_1, L_{2,....}L_k$;
19 : Let $[L_d]$ be the size of $L_{d;}$
20: Let Cen_d be the related datacenter;
21: **if** J is small MH jobs {// use policy B
22: Append $[L_d]$ map tasks of J to the end of $MQ_{d,0;}$}
23: **else** {/* i.e J is a large job , so use policy C */
24: Let p be the total number of Map task queues in cen_d;
25 : Generates a new map –task queue $MQ_{d,p+1}$;
26: Append $[L_d]$ map tasks of j to the end of $MQ_{d,p+1}$;}
27: for c=1to k{
28: Delete a block from L_c if the block is in L_d;}
29: a=a-$[L_d]$;}
30: Let cen_c be a datacenter holding the largest number of unique input blocks of J;
31: Unique input blocks of J
32: **if** J is a small MH job{// Use policy B
33: Append all reduce task of J to the end of $RQ_{e,o}$;
34: **else** {/* i.e J is a large job so use policy C*/
35: Let q be the total number of reduce –task queues in cen_e;
36: Generate a new reduce task queue $RQ_{e,q+1}$;
37:Append all reduce tasks of J to the end of $RQ_{e,q+1}$;}}}

3.2. The algorithm task –driven task assigner(TTA) :-

TTA algorithm below Contains 22 execution steps & Input :an idle slot of $VPS_{c,l}$,Output: a task assigned to $VPS_{c,}l$.

Procedure:

1: Let Imap and Ired be two indexes with same initial value 0;
2: **while** $VPS_{c,l}$ has an idle slot{
3: Let N_{map} be the total number of Map-task queues in $cen_{c;}$
4: Let N_{red} be the total number of Reduce-task queues in cen_c;
5: **If** the slot is a map slot{
6: **If** MQ_{FIFO} is not empty{
7: Use FIFO to assign a map task from MQ_{FIFO} to $VPS_{c,l}$;
8: Remove the task from MQ_{FIFO}:}
9: **else**{
10: $I_{map}=I_{map}$ mod $(N_{map}+1)$;
11: Assign the map task from $MQ_{c,Imap}$ to $VPS_{c,l}$;
12: Remove the task from $MQ_{c,Imap}$;
13: I_{map}++}}
14: **else** {/* i.e the idle slot is a reduce slot:*/
15: **if** RQ_{FIFO} is m\not empty{
16: Assign the first reduce task from $RQ_{FIFO\ to}$ $VPS_{c,}$l;
17: Remove the task from RQ_{FIFO};}

18: **Else** {
19: I_{red} = Ired mod(N_{red}+1);
20 :Assign the first reduce task from RQ_c,I_{red} to VPS_c,l ;
21: Remove the task from $RQ_{c,Ired;}$
22: Ired++;}}}

3.3. The algorithm Job –driven task assigner(JTA) :-

The JTA algorithm below Contains 22 excution steps & Input :an idle slot of VPSc,l , Output: a task assigned to VPSc,l .

Procedure:

1: Let Imap and Ired be two indexes with same initial value 0;
2: **while** VPSc,l has an idle slot{
3: Let Nmap be the total number of Map-task queues in cen_c;;
4: Let Nred be the total number of Reduce-task queues in cen_c;
5: **If** the slot is a map slot{
6: **If** MQFIFO is not empty{
7: Use FIFO to assign a map task from MQFIFO to VPSc,l;
8: Remove the task from MQFIFO:}
9: **else**{
10: Imap=Imap mod (Nmap+1);
11: Use FIFO to assign the map task from $MQc_{,Ima\ p}$to VPSc,l ;
12: Remove the task from $MQc_{,Imap}$;
13: Imap++}}
14: **else** {/* i.e the idle slot is a reduce slot:*/
15: if RQ FIFO is m\not empty{
16: Assign the first reduce task from RQFIFO to VPSc,l;
17: Remove the task from RQFIFO;}
18: **Else** {
19: Ired = Ired mod(Nred+1);
20: Assign the first reduce task from RQc,Ired to VPSc,l ;
21: Remove the task from RQc,Ired;
22: Ired++;}}}

4. EVALUATION

In this section the job scheduling scheme algorithms implemented as the algorithm of the task scheduler, the algorithms of task -driven task assigner(TTA) and the job –driven task assigner(JTA) using these two algorithm implemented the algorithm as the schedule the job JOSS with TTA and schedule the job JOSS with JTA , to combine the the algorithms task scheduler and TTA the JOSS-T algorithm form and to combine the the algorithms task scheduler and JTA the JOSS-J algorithm form this is executed using the the three jobs as Word Count Job computes the occurrence frequency of each word in a document & Word Count is a large job therefore it uses the scheduling Policy C, Inverted Index job inverted index takes a list of documents as input and generates word o document indexing, this job is a small Reduce heavy job and it uses the policy A , Sort job this job Sorts the data in the input files in a dictionary order these three jobs are the MapReduce benchmarks ,Jobs used MapReduce Scheduling Polices , Polices A ,Policy B, Policy C as Policy A is designed for a small RH job ,Policy B is designed for a small Map-Heavy (MH) job ,Policy C is designed for a large job, the three types of jobs like word count, inverted index, sort job using these job implemented the schedule jobs with JOSS –TTA and JOSS-JTA in this algorithm MapReduce programming operation is performed using Map Task and Reduces task and schedule jobs according to job type by using scheduling policy, after implementing the the JOSS with TTA algorithm and JOSS with JTA then compare the Processing Time of JOSS with TTA and JOSS With JTA scheduling algorithm this shows the total time taken by Mapper and reducer to process the job is represented as processing time in milliseconds the result graph of Comparison Processing time JOSS with TTA and JOSS with JTA is as in below Fig 1 and to create a small workload and Mixed workload the is used five Mapreduce Benchmark as Word-Count(WC),Sequence-Count(SC), Inverted–Index(II), Grep, Permu are used & WC counts the occurrence of each word in data files, SC generates a count of all unique sets of three consecutive words in data files as input and generates word to file indexing ,Grep searches for a pattern in data files and Permu generates the permutation for three consecutive DNA sequences in DNA data files, the both workload is used the Map data locality is divited into VPS- locality rate is the ratio of total number of the Map task that can achieve the VPS locality to the the total number of the map task in the workload and Cen-locality rate is the ratio of total number of the Map task that can achieve the Cen-

locality to the locality to the the total number of the map task in the workload and off-cen rate is the 1- (VPS-locality rate + Cen- locality rate) for the value ranges from 0 to 1 the Reduce data locality rate is the percentage of input data that a reducer can obtain from its local datacenter for this value ranges form 0 to 1 , this MapRedue benchmark is used to evaluate the performance of JOSS-T and JOSS-J algorithm the Evaluation performance graph result figures as below.

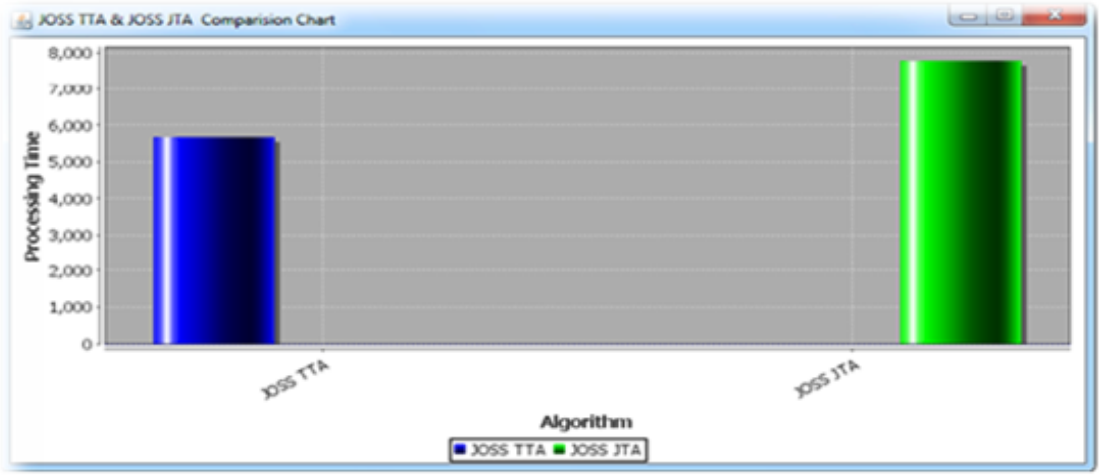

Fig 1. Processing time in (milliseconds) of JOSS TTA & JOSS JTA Comparison Chart .

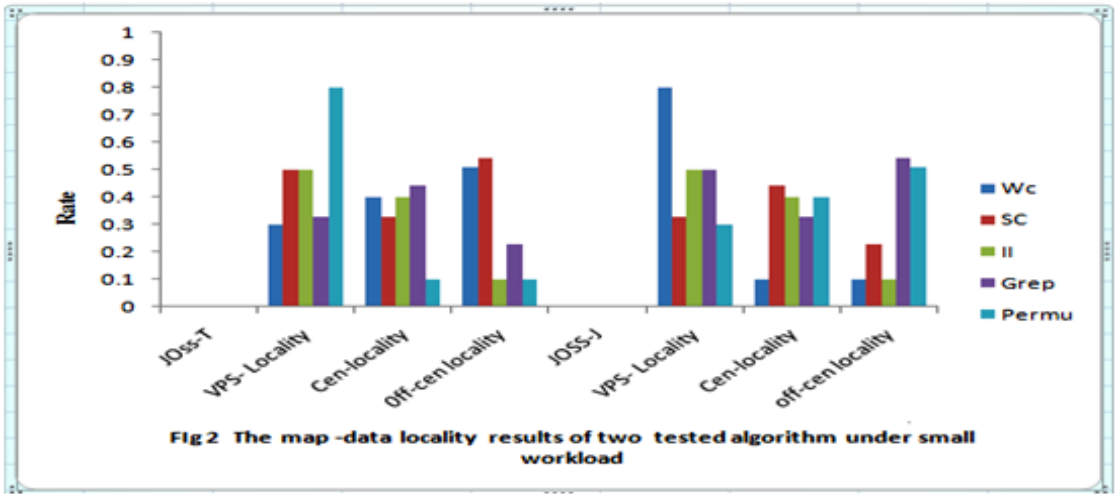

Fig 2 The map -data locality results of two tested algorithm under small workload

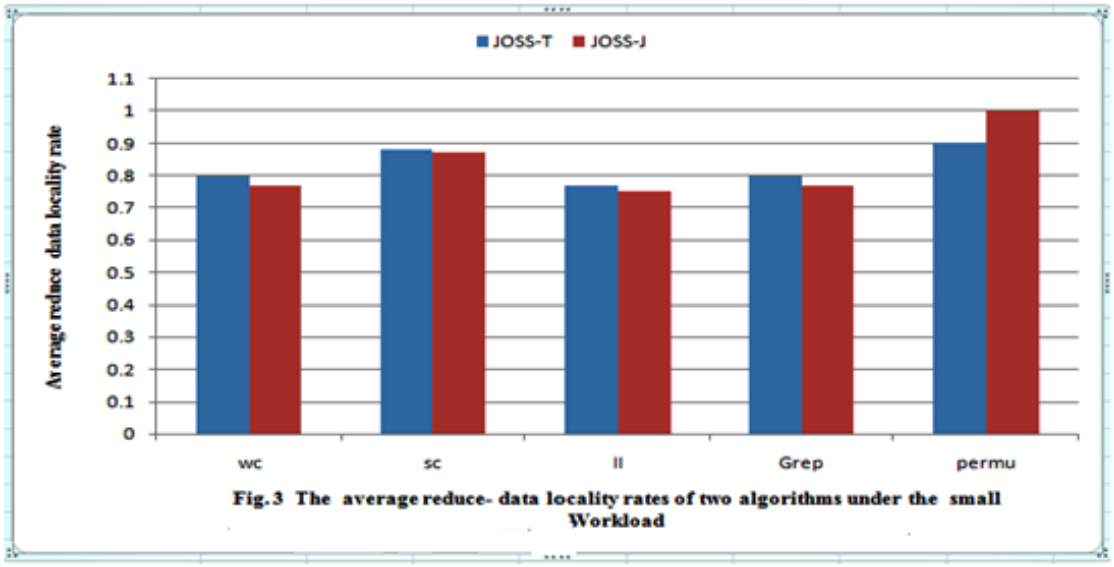

Fig. 3 The average reduce- data locality rates of two algorithms under the small Workload

e-ISSN: 2394-8299
p-ISSN: 2394-8280

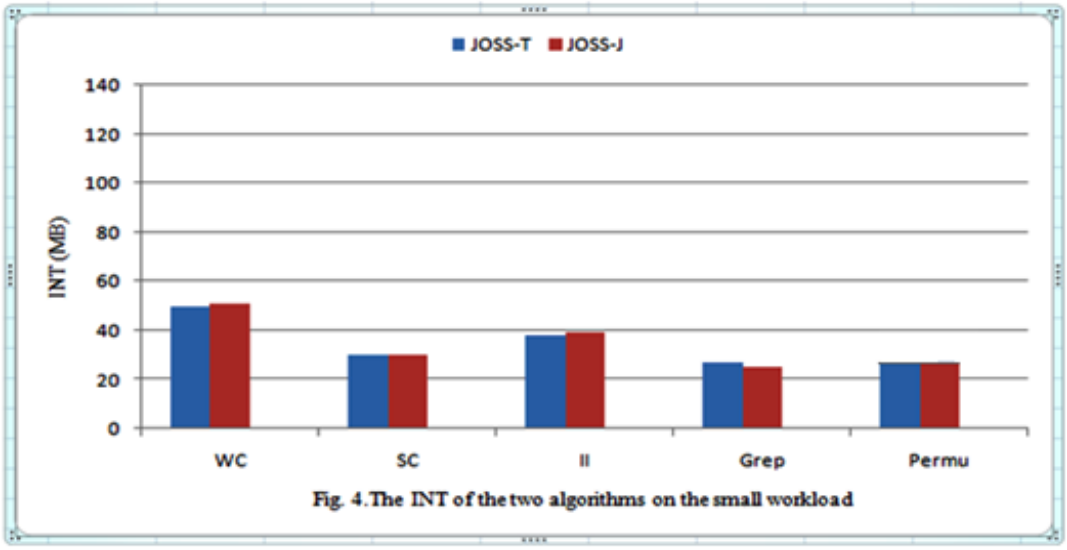

Fig. 4.The INT of the two algorithms on the small workload

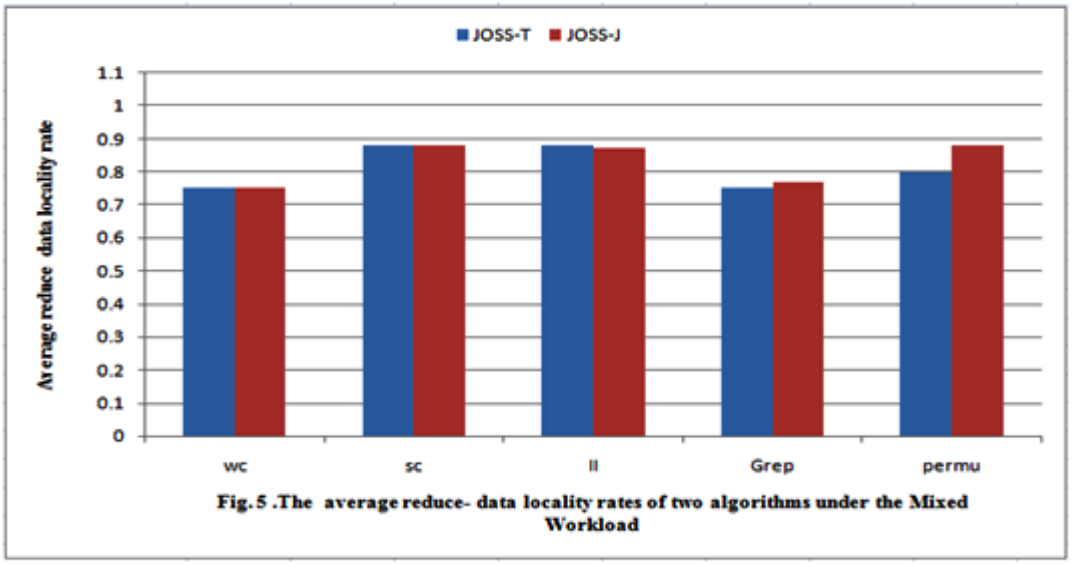

Fig. 5 .The average reduce- data locality rates of two algorithms under the Mixed Workload

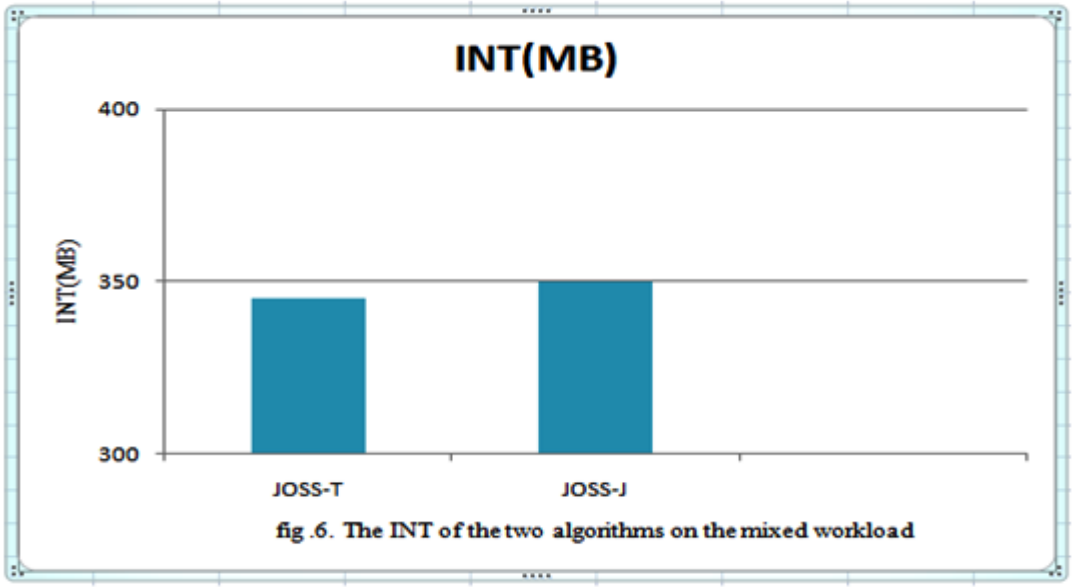

fig .6. The INT of the two algorithms on the mixed workload

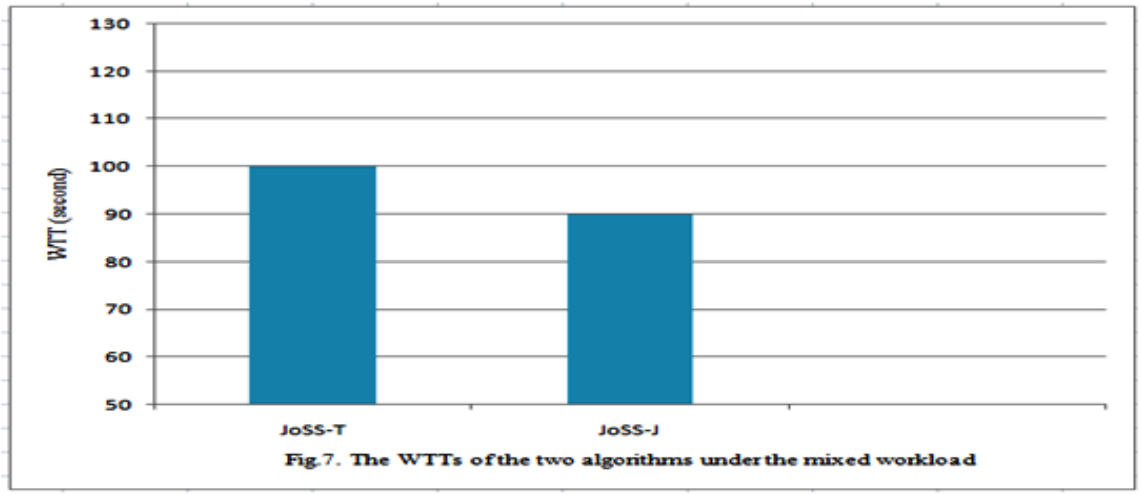

Fig.7. The WTTs of the two algorithms under the mixed workload

5. CONCLUSION

In this paper job scheduling algorithms is implemented as task scheduling algorithm and JOSS with TTA algorithm and JOSS with JTA algorithm ,the JOSS-TTA algorithm is schedule job for fast task assignment for this reason it has require minimum processing time for scheduling job as compare to JOSS with JTA algorithm and the scheduling algorithm JOSS With JTA improves VPS data locality rate of map task and reduce task ,it require large processing time for job scheduling as compared to JOSS with TTA algorithm & JOSS is used job level scheduling ,Map task level scheduling and reduce task level scheduling ,JOSS classifies Mapreduce jobs based on job scale and job type and designs an appropriate scheduling polices to schedule each class of jobs the scheduling policy A is used for small Reduce Heavy (RH) Job, Policy B is used small Map Heavy(MH) job and policy C is used to large job therefore job of scheduling scheme(JOSS) increases data locality and improves job performance, avoids job starvation and improves job performance & the five MapReduce Benchmark is used create Small workload and Mixed workload for this evaluating Performance of JOSS-T and JOSS-J Scheduling algorithm and job scheduling is used MapReduce job using MapReduce technique of Hadoop (FIFO) algorithm.

REFERENCES

[1] C. Tian, H. Zhou, Y. He, and L. Zha, "A dynamic mapreduce scheduler for heterogeneous workloads," in Proc. IEEE 8th Int. Conf. Grid Cooperative Compute., 2009, pp. 218–224 .
[2] F. Ahmad, S. Lee, M. Thottethodi, and T. N. Vijaykumar. (2012). PUMA: Purdue MapReduce benchmarks suite. ECE Tech. Rep., Purdue Univ. [Online]. Available: http://docs.lib.purdue.edu/ ecetr/437.
[3] Z. Guo, G. Fox, and M. Zhou, "Investigation of data locality in mapreduce," in Proc. 12th IEEE/ACM Int. Symp. uster, Cloud Grid.Comput., 2012, pp. 419–426.
[4] J. Dean, S. Ghemawat, " MapReduce: simplified data processing on large clusters", Communications of the ACM, 2008, Vol. 51, No. 1, pp. 107-113.
[5] J. Jin, J. Luo, A. Song, F. Dong, and R. Xiong, "BAR: An efficient data locality driven task scheduling algorithm for cloud computing," in Proc. 11th IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput., 2011, pp. 295–304.
[6] C. He, Y. Lu, and D. Swanson, "Matchmaking: A new mapreduce scheduling technique," in Proc. IEEE 3rd Int. Conf. Cloud Comput. Technol. Sci., Nov. 2011, pp. 40–47.
[7] M. Zaharia et al. "Delay scheduling: A simple technique for achieving locality and fairness in cluster scheduling" In EuroSys,2010.
[8] M. Ehsan, and R. Sion, "LiPS: A cost-efficient data and task co-scheduler for MapReduce," in Proc. IEEE 27th Int. Symp.Parallel Distrib. Process. Workshops PhD Forum, 2013, pp. 2230–2233.
[9] J. Polo, D. Carrera, Y. Becerra, J. Torres, E. Ayguad_e, M. Steinder, and I. Whalley, "Performance-driven task co-scheduling for mapreduce environments," in Proc. IEEE Netw. Oper. Manage. Symp. 2010, pp. 373–380.