Job Allocation Scheduler with locality for Efficient Deployment of MapReduce Applications in Heterogeneous Computing Environments

P. Ashwini Reddy[1], T. Swetha Reddy[2], B. Sandhya[3], P. Jyothsna[4]
[1] Asso.Professor, TKREC, Hyderabad, TS-India, Email:ashwinireddy90@gmail.com
[2] B.Tech C.S.E TKREC Hyderabad Email: thummalaswetha1996@gmail.com
[3] B.Tech C.S.E TKREC Hyderabad Email:sandysandya777@gmail.com
[4] B.Tech C.S.E TKREC Hyderabad Email: jyothsnapally110@gmail.com

1. ABSTRACT

Cloud computing has become increasingly popular model for delivering applications hosted in large data centers as subscription oriented services. Hadoop is a popular system supporting the MapReduce function, which plays a crucial role in cloud computing. The resources required for executing jobs in a large data center vary according to the job type. In Hadoop, jobs are scheduled by default on a first-come-first-served basis, which may unbalance resource utilization. This paper proposes a job scheduler called the *job allocation scheduler* (JAS), designed to balance resource utilization. For various job workloads, the JAS categorizes jobs and then assigns tasks to a *CPU-bound queue* or an *I/O-bound queue*. However, the JAS exhibited a locality problem, which was addressed by developing a modified JAS called the *job allocation scheduler with locality* (JASL). The JASL improved the use of nodes and the performance of hadoop in heterogeneous computing environments. Finally, two parameters were added to the JASL to detect inaccurate slot settings and create a dynamic job allocation scheduler with locality (DJASL). The DJASL exhibited superior performance than did the JAS, and data locality similar to that of the JASL.
.

Keywords: Hadoop, heterogeneous environments, heterogeneous workloads, MapReduce, scheduling

2. INTRODUCTION

BIG DATA

"Big data" is a term used to describe a collection of data sets with the following three characteristics:

i. Volume- Large amounts of data generated.
ii. Velocity-Frequency and speed of which data are generated, captured and shared
iii. Variety-Diversity of data types and formats from various sources.

The size and complexity of big data makes it difficult to use traditional database management and data processing tools. Data is being created in much shorter cycles from hours to milliseconds. There is also a trend underway to create larger databases by combining smaller data sets so that data correlations can be discovered.

Big data has become the new frontier of information management given the amount of data today's systems are generating and consuming. It has driven the need for technological infrastructure and tools that can capture, store, analyse and visualize vast amounts of disparate structured and unstructured data. These data are being generated at increasing volumes from data intensive technologies including, but not limited to, the use of the Internet for activities such as accesses to information, social networking, mobile computing and commerce. Corporations and governments have begun to recognize that there are unexploited opportunities to improve their enterprises that can be discovered from these data.

3. Proposed work

An approach for balancing resource utilization in Hadoop systems in heterogeneous computing environments such as clouds.

To overcome the limitations of current mapreduce application platforms, we first proposes a job scheduler called job allocation scheduler (JAS) for balancing resource utilization in heterogeneous computing environments. The JAS divides jobs into two classes (CPU and I/O bound) to test the capability of each TaskTracker According to the capacity ratio for job types to maximize resource utilization.
The proposes a modified JAS, called job allocation with locality(JASL).The JASL can record each node's execution time, and then compare the execution times of the local and non-local nodes too determine whether the task can be executed on non-local node.in addition , an enhased JASL , called dynamic job allocation scheduler with locality(DJASL), was developed by adding a dynamic function to the JASL.

4. System architecture

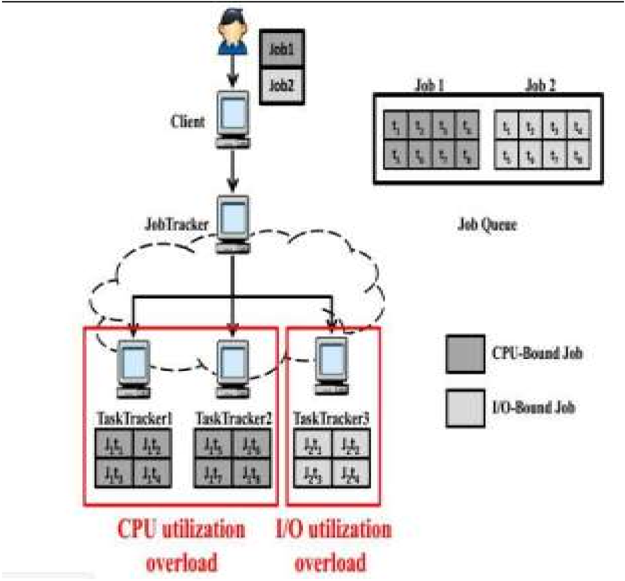
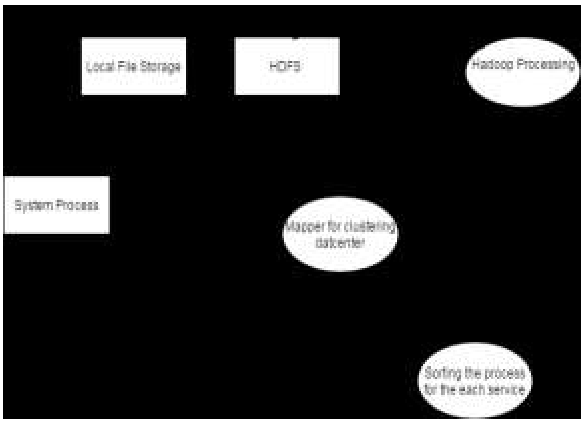


Fig: System Architecture

DATA FLOW DIAGRAM:

1. The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
2. The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.
3. DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.
4. DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.


Local File Storage
HDFS
Hadoop Processing
System Process
Mapper for clustering datcenter
Sorting the process for the each service

.

5. Algorithm

Job queue (set CPU-slot)

```
for Job in Job Queue do
    if Job has been completed and Job is CPU-bound then
        obtain the task information from TaskTrucker;
        compute the TaskTrucker capability according to TaskTruckerCPUCapability;
        $c_y := \frac{|M_y|}{e_y}$;
        for each TaskTrucker do
            $k_y :=$ (the number of CPU slots) $* \frac{c_y}{\sum_{j=1}^{m} c_j}$;
            record $k_y$ on TaskTruckerCPUTable;
            SetTaskTruckerCPUTable := 1;
        return TaskTruckerCPUslot according to TaskTruckerCPUTable;
        break;
```

It reads the execution of job time

Job queue(set I/O_slot)

```
for Job in Job Queue do
    if Job has been finished and Job is I/O-bound then
        obtain the task information from TaskTracker;
        compute the TaskTracker capability according to TaskTrackerIOCapability;
        $d_y := \frac{n_y}{f_y}$;
        for each TaskTracker do
            $i_y :=$ (the number of I/O slots) $* \frac{d_y}{\sum_{j=1}^{m} d_j}$;
            record $i_y$ onTaskTrackerIOTable;
            SetTaskTrackerIOTable := 1;
        return TaskTrackerIOslot according to TaskTrackerIOTable;
        break.
```

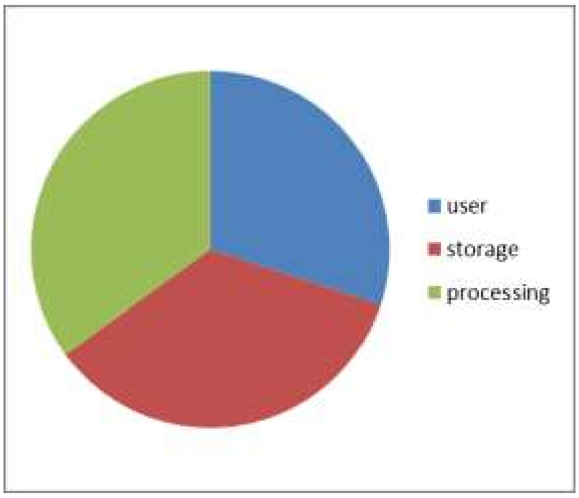
In this algorithm the input data's slots are given time for every task.

DJASL

```
When a batch of jobs are submitted into JobTracker:
add jobs into Job Queue;
Initialize SetTaskTrackerCPUTable := 0;
Initialize SetTaskTrackerIOTable := 0;
Initialize CPUcount := 0;
Initialize IOcount := 0;
Initialize LocalityBenifitTable := 0;
while receive Heartbeat by TaskTracker do
    TaskTrackerCPUslot := 0;
    TaskTrackerIOslot := 0;
    obtain TaskTrackerRunningCPUtask from Heartbeat information;
    obtain TaskTrackerRunningIOtask from Heartbeat information;
    AvailableCPUSlots := 0;
    AvailableIOSlots := 0;
    JOB_CLASSIFICATION_L(Heartbeat);
    if SetTaskTrackerCPUTable == 1 then
        obtain TaskTrackerCPUslot according to TaskTrackerCPUTable;
    else
        TaskTrackerCPUslot :=SET_CPU_SLOT(Job Queue);
    if SetTaskTrackerIOTable == 1 then
        obtain TaskTrackerIOslot according to TaskTrackerIOTable;
    else
        TaskTrackerIOslot :=SET_IO_SLOT(Job Queue);
    if TaskTrackerCPUslot == 0 then
        TaskTrackerCPUslot := default CPU slot;
    if TaskTrackerIOslot == 0 then
        TaskTrackerIOslot := default I/O slot;
    if TaskTracker.CPUusage > 90% of CPU usage then
        CPUcount += 1;
    if TaskTracker.IOusage > 35MB then
        IOcount += 1;
    if CPUcount >= 100 then
        reset CPU_slot;
    if IOcount >= 100 then
        reset I/O_slot;
    AvailableCPUSlots := TaskTrackerCPUslot − TaskTrackerRunningCPUtask;
    AvailableIOSlots := TaskTrackerIOslot − TaskTrackerRunningIOtask;
    CPU_TASK_ASSIGN_L(AvailableCPUSlot);
    IO_TASK_ASSIGN_L(AvailableIOSlot).
```

When the frequency is overload it can maintain the tasktracker.

6. Result analysis



7.Conclusion

The JAS provide highly efficient job scheduler for hadoop system .according to the job tracker first computes the capability of each task tracker and then sets the no.of CPU and I/Pslots accordingly

The DJASL also improved the data locality of the JAS by approximately 27% this scheduling alg is not only useful for hadoop system but also applicable to other cloud software systems as YARN and Aneka

8. Future Enhancement

In the future the characterization and performance comparisons of CPU and I/O-bound jobs will be provided.

CPU and I/O-bounds can be parallelized to balance resource utilization.

9. References

[1] Apache Hadoop. http://hadoop.apache.org/

[2] Apache Hadoop YARN. http://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html

[3] Hadoop's Capacity Scheduler. http://hadoop.apache.org/core/docs/current/capacity scheduler.html.

[4] Matei Zaharia, "The Hadoop Fair Scheduler" http://developer.yahoo.net/blogs/hadoop/FairSharePres.ppt

[5] Ahmad, F., Chakradhar, S. T., Raghunathan, A., and Vijaykumar, T. N., "Tarazu: optimizing mapreduce on heterogeneous clusters," In ACM SIGARCH Computer Architecture News, Vol. 40, No. 1, pp. 61–74, 2012.

[6] Atallah, M. J., Lock, C., Marinescu, D. C., Siegel, H. J., and Casavant, T. L., " Co-scheduling compute-intensive tasks on a network of workstations: model and algorithms," In Proceedings of the 11th International Conference on Distributed Computing Systems, pp. 344–352, 1991.

[7] Bezerra, A., Hernandez, P., Espinosa, A., Moure, J.C., "Job scheduling in Hadoop with Shared Input Policy and RAMDISK," Cluster Computing (CLUSTER), 2014 IEEE International Conference , pp. 355–363, 2014.

[8] Feitelson, D. G., and Rudolph, L., "Gang scheduling performance benefitsfor fine-grained synchronization," Journal of Parallel and Distributed Computing, Vol. 16, No.4 , pp. 306–318, 1992.

[9] Ghemawat, S., Gobioff, H., and Leung, S. T., "The Google file system," In ACM SIGOPS Operating Systems Review, Vol. 37, No. 5, pp. 29–43, 2003.

[10] Ghoshal, D., Ramakrishnan, L.,"Provisioning, Placement and Pipelining Strategies for Data-Intensive Applications in Cloud Environments," Cloud Engineering (IC2E), 2014 IEEE International Conference , pp. 325–330, 2014.

[11] Ghodsi, A., Zaharia, M., Shenker, S., and Stoica, I., "Choosy: max-min fair sharing for datacenter

jobs with constraints," In Proceedings of the 8th ACM European Conference on Computer Systems,
pp. 365–378, 2013.
[12] Hammoud, M., and Sakr, M. F. , "Locality-aware reduce task scheduling for MapReduce," In Proceedings
of IEEE Third International Conference on Cloud Computing Technology and Science (Cloud-
Com), pp. 570–576, 2011.
[13] Ibrahim, S., Jin, H., Lu, L., Wu, S., He, B., and Qi, L, "Leen: Locality/fairness-aware key partitioning
for mapreduce in the cloud," In Proceedings of IEEE Second International Conference on Cloud
Computing Technology and Science (CloudCom), pp. 17–24, 2010.
28
2168-7161 (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/ publications/rights/index.html for more information.
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2016.2552518, IEEE
Transactions on Cloud Computing
[14] Isard, M., Prabhakaran, V., Currey, J., Wieder, U., Talwar, K., and Goldberg, A., "Quincy: fair
scheduling for distributed computing clusters," In Proceedings of the ACM SIGOPS 22nd symposium
on Operating systems principles, pp. 261–276, 2009.
[15] J.K. Ousterhout, "Scheduling techniques for concurrent systems," in Proceedings of the third International
Conference on Distributed Computing Systems, pp. 22–30, 1982.
[16] Lee, W., Frank, M., Lee, V., Mackenzie, K., and Rudolph, L., "Implications of I/O for Gang Scheduled
Workloads," In Proceedings of Springer Berlin Heidelberg on Job Scheduling Strategies for Parallel
Processing, pp. 215–237, 1997.
[17] Lee, H., Lee, D., and Ramakrishna, R. S., "An Enhanced Grid Scheduling with Job Priority and
Equitable Interval Job Distribution," In Proceedings of the first International Conference on Grid
and Pervasive Computing, Lecture Notes in Computer Science, pp. 53–62, 2006.
[18] Page, A. J., and Naughton, T. J., "Dynamic task scheduling using genetic algorithms for heterogeneous
distributed computing," In Proceedings of 19th IEEE International on Parallel and Distributed
Processing Symposium, pp. 189a–189a, 2005.
[19] Reiss, C., Tumanov, A., Ganger, G. R., Katz, R. H., and Kozuch, M. A., "Heterogeneity and dynamicity
of clouds at scale: Google trace analysis," In Proceedings of the Third ACM Symposium on
Cloud Computing, pp. 7, 2012.
[20] Rosti, E., Serazzi, G., Smirni, E., and Squillante, M. S., "The Impact of I/O on Program Behavior
and Parallel Scheduling," In ACM SIGMETRICS Performance Evaluation Review, Vol. 26, No. 1,
pp. 56–65, 1998.
[21] Rosti, E., Serazzi, G., Smirni, E., and Squillante, M. S., "Models of Parallel Applications with Large
Computation and I/O Requirements," In IEEE Transactions on Software Engineering, Vol. 28, No.
3, pp. 286–307, 2002.
[22] Schwarzkopf, M., Konwinski, A., Abd-El-Malek, M., and Wilkes, J., "Omega: flexible, scalable schedulers

for large compute clusters," In Proceedings of the 8th ACM European Conference on Computer
Systems, pp. 351–364, 2013.
[23] Tian, C., Zhou, H., He, Y., and Zha, L., "A Dynamic MapReduce Scheduler for Heterogeneous Workloads,"
In Proceedings of the 8th IEEE International Conference on Grid and Cooperative Computing,
pp. 218–224, 2009.
[24] Tumanov, A., Cipar, J., Ganger, G. R., and Kozuch, M. A., "alsched: Algebraic scheduling of mixed
workloads in heterogeneous clouds," In Proceedings of the third ACM Symposium on Cloud Computing,
pp. 25, 2012.
[25] Joe Weinman,"Cloud Computing is NP-Complete" Working Paper, 2011.
[26] Wiseman, Y., and Feitelson, D. G., "Paired Gang Scheduling," In IEEE Transactions on Parallel and
Distributed System, Vol. 14, No. 6, pp. 581–592, 2003.
[27] Zaharia, M., Borthakur, D., Sen Sarma, J., Elmeleegy, K., Shenker, S., and Stoica, I., "Delay scheduling:
a simple technique for achieving locality and fairness in cluster scheduling," In Proceedings of
the 5th European conference on Computer systems, pp. 265–278, 2010.
[28] Zhang, X., Zhong, Z., Feng, S., Tu, B., and Fan, J., "Improving data locality of mapreduce by
scheduling in homogeneous computing environments," In Proceedings of the 9th IEEE International
Symposium on Parallel and Distributed Processing with Applications (ISPA), pp. 120–126, 2011.