ISSN: 0975-766X
CODEN: IJPTFI
*Available Online through* *Research Article*
www.ijptonline.com

# ADAPTIVE CORE ALLOCATION FOR MAPREDUCE CLUSTERS IN CLOUD COMPUTING

**M.Lavanya[1], T.Abinaya[2], R.Reka[2], S.Saravanan[1]**
[1]Assistant Professor, School of Computing, SASTRA University, TamilNadu, India.
[2]B.TechICT, School of Computing, SASTRA University, TamilNadu, India.
*Email: m_lavanyass@ict.sastra.edu*

*Received on 20-08-2016* *Accepted on 25-09-2016*

**Abstract**

Map Reduce is the heart of Hadoop, which is increasingly adopted as a programming paradigm by data intensive applications for distributed processing of large-scale data using a cluster of machines. Fixed slot-based resource management model used in Hadoop Map Reduce implementation suffers from performance degradation due to its indecorous resource allocation. Dynamic Map Reduce already exists to overcome the drawbacks of the traditional map reduce technique. The proposed paper introduces a technique called Adaptive Core Allocation for map reduce clusters to further optimize and improve slot utilization of existing Dynamic Hadoop Slot Allocation in Dynamic MapReduce. This will also lead to an effective utilization of processor. Experimental analysis shows the feasibility of the proposed algorithm which indicates the amount of time saved in order to process all tasks of an incoming jobs and it is compared with traditional map reduce techniques. Algorithm has been generated to prove the work.

**Keyword:** MapReduce, Fixed slot –based model, Adaptive Core Allocation, Dynamic MapReduce Corresponding Author: M.Lavanya (m_lavanyass@ict.sastra.edu)

## I. Introduction

Cloud system is a computing paradigm model. Now a day, one of the best-known cloud platforms for big data is Apache Hadoop. To implement Map Reduce, open source software is available named Hadoop. Hadoop Map Reduce is a software configuration for distributed and parallel processing of huge datasets in a cluster of computers. Map Reduce has been utilized by enterprises such as Facebook and Yahoo to support sequential processing of multiple jobs submitted by multiple users. There are many works regarding the optimization of Map Reduce and several summons to improve slot utilization and system performance.

The foremost challenge is that the CPU cores are abstracted into map core and reduce core which are preconfigured

by administrator. In this the cores are underutilized due to the assumption that map slots are allocated to mapping tasks and reduce slots are allocated to reducing tasks. This assumption adversely affects the system performance and slot utilization. To overcome the above mentioned summon, we proposed an optimization techniques viz, Adaptive Core Allocation (ACA) for map reduce clusters in cloud computing. This technique will improve slot utilization and system performance.

**II. Related Work**

*Optimization Of Scheduling And Resource Allocation:* The job sequence optimization for MapReduce workload is discussed in [2], [3].They have presented a two-staged MapReduce hybrid flow shop with multiprocessor tasks. In their model, the different job submission orders will result in varied system performance and cluster usage. However, there is a need to know the execution time of map and reduce tasks in advance. So it is only suitable for independent jobs. In comparison, our Adaptive Core Allocation technique is not restricted to such constraints and can be used for any type of map and reduce tasks. A new version of Hadoop with totally different architecture is YARN [1].It eliminates the concept of slot by introducing a „container" where both map and reduce tasks can run. But it provides insignificant performance in the case of multiple jobs. However, for multiple jobs, our ACA outperforms YARN [1].Polo et al. [7] proposed a resource aware scheduling technique for Map Reduce multi job workloads. Their aim is to improve resource utilization by extending the notion of conventional „task slot" to „job slot". This job slot confines the usage of typed tasks to the respective typed slots. Whereas our ACA, dynamically allocates unused map or reduce slots to overloaded reduce or map tasks.

*Optimizing Map Reduce on Cloud Computing*: There are numerous works on optimizing the Map Reduce onCloud Computing. The key works focus on the deadline and budget optimization [4], [5], and [6].They offered algorithms and cost models to revamp the task scheduling and resource allocation management for Map Reduce workloads for each metric. However, their works are on top of Hadoop framework, which is associated with coarse-grained optimization. In contrast, our project is a fine-grained optimization for Hadoop. Thus, we can merge existing work and our proposed techniques to further optimize the deadline and budget in Cloud Computing proposes an Method called *Dynamic Hadoop Slot Allocation(DHSA*),to relax the restrictions of slot allocation and allocate in both a slots to provide the dynamic approach, still at some instant it is similar to "container" concept in YARN. Mapping and allocation of advanced tasks and best effort tasks are discussed in [8] includes energy consumption concept. Slot allocation done here based on the type of the tasks. Partition scheduling of multiprocessors are discussed in paper[10]

which concentrates on global queue concept which has push based technique model of HADOOP.

**III. Proposed Work**

The basic compute units are the map and reduce slots, which are the abstracted compute resources (CPU cores). The incoming application job is initially splitted into independent tasks. These tasks are initially loaded into map queue. The task scheduler assigns the tasks whose map status is false to those mapped units which is free at that instant of time. Once the mapping of a task is finished, its map status is set to true and then it is loaded into the reduce queue. Similarly, the tasks inside reduce queue are scheduled to a free reducing unit when the map status is true and the reduce status is false. When the map and reduce work of a particular task is over, it is put into the completed queue. However, this existing Map Reduce design suffers from underutilization of the compute resources. This happens because map cores and reduce cores processes map tasks and reduce tasks respectively. Due to this, it could be observed that at different point of time there may be idle map or reduce cores as the job proceed from map phase to reduce phase.

This paper proposes an Adaptive Core Allocation technique for Map Reduce clusters aiming to utilize the free map and reduce cores processing time for those overloaded map or reduce queues. That is, at any point of time ,if there exists a free core, it acquire the task from the overloaded map or reduce queue and performs mapping or reducing function, irrespective of the core‟s designated function. By adopting our proposed technique, the performance of Map Reduce cluster is improved by optimizing the core utilization.

**Algorithm**

**When a job is received from a compute node *n*:**

1:*compute its number of independent tasks,each tasks processing time.*

2:*Set the mapStatus and reduceStatus of each task to false initially*

3:*And insert the map tasks into mapQueue*

*/* Case 1:**mapCore is free,mapQueue has pending map tasks** */*

4: if (*IsMapCoreFree==trueand mapStatus ==false*) then

5:*Set IsMapCoreFree to false*

6:*Process the map task for the estimated amount of processing time*

7:*Set mapStatus to true and IsMapCoreFree to true*

8:*Submit the completed map task to reduceQueue*

9:end if

*/*Case 2*:***mapCore as reduceCore when***

***mapCore is free,mapQueue has no map tasks ,reduceQueue has pending reduce tasks****/*

10: if(*IsMapCoreFree==trueand mapQueue==null)* then

11: *if(reduceQueue not null and reduceSatus==false) then*

12: *Set IsMapCoreFree to false*

13: *Process the reduce task for the estimated amount of processing time*

14: *Set reduceStatus to true and IsMapCoreFree to true*

15: *Submit the completed reduce task to completedQueue*

16: end if

*/*Case 3:****reduce Core is free,reduceQueue has pending reduce tasks****/*

17: if (*IsReduceCoreFree==trueand reduceStatus ==false*) then

18: *Set IsReduceCoreFree to false*

19:*Process the reduce task for the estimated amount of processing time* 20:*Set reduceStatus to true and Is Reduce Core Free to true*

21: *Submit the completed reduce task to completedQueue*

22: end if

*/*Case 4*: ***reduce Core as mapCore when***

***reduceCore is free,reduceQueue has no reduce tasks ,mapQueue has pending map tasks****/*

10: if(*IsReduceCoreFree==trueand reduceQueue==null)* then

11: *if(mapQueue not null and mapSatus==false) then*

12: *Set IsReduceCoreFree to false*

13: *Process the map task for the estimated amount of processing time*

14: *Set mapStatus to true and IsReduceCoreFree to true*

15: *Submit the completed map task to reduceQueue*

16: end if

17: Collect the completed tasks of a job and return the result of the job as a whole to the console.

**IV. Experimental Analysis:** To illustrate the proposed method, the cluster of quad-core processor resources are

chosen in which two cores are abstracted as map cores and the other two cores are abstracted as reduce cores. The incoming job is splitted into number of independent tasks. Next the map status and reduce status of the tasks are initialized to false.

Then we allotted those independent tasks to map/reduce cores by checking the status of the tasks and also based on the freeness of cores. These tasks will be processed by map/reduce cores according to the predetermined processing time of the particular tasks. Once the map and reduce status of the tasks are set to true by the cores after completing the mapping and reducing work, finally it is submitted to completed queue. Comparison analysis table(1) shows the overall time in milliseconds taken by the traditional map reduce and adaptive core allocation to process all independent tasks. The time difference in the table indicates the amount of time saved by the proposed ACA method in order to process all tasks. This shows the proposed ACA method works better than the existing map reduce techniques.

Table 1. Comparison analysis between Traditional Map Reduce and Adaptive Core method, It is clear that ACA algorithm increase the system performance of CLOUD environment which reduces the processing time of overall job. Analysis performed with 50,100,500 and 1000 number of tasks and the overall processing time is reduced by our proposed algorithm which also improves the efficient processor utilization.

| **No of Tasks** | **50** | **100** | **500** | **1000** |
|---|---|---|---|---|
| **DMR [8]** | 8577 ms | 14843 ms | 76250 ms | 147714 ms |
| **ACA** | 5566 ms | 11741 ms | 72460 ms | 115066 ms |
| **Time Difference** | 3011 ms | 3102 ms | 3790 ms | 3648 ms |
| **Units saved** | **30%** | **31%** | **37%** | **36%** |

## V. Conclusion

This paper proposes Adaptive Core Allocation framework aiming to further optimize and improve core utilization of existing mapreduce techniques. The underutilization of cores are totally evaded by breaking the implicit assumption of map tasks can run only on map cores and reduce tasks can run only on reduce cores. Instead, based on the freeness of cores, we modify it as a map tasks can run on reduce cores or a reduce tasks can run on map cores. This results in reduction of processing time of jobs and avoids the underutilization of cores.

The experimental results show that our proposed adaptive core allocation can improve the performance of the Hadoop system significantly. This work can be extended to improve the speculative task allocation of the map reduce cluster.

**References**


1. Apache Hadoop NextGen MapReduce (YARN). (2014).
2. Available: http://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-ite/YARN.html.
3. C. O_guz and M. F. Ercan, 1997, Scheduling multiprocessor tasks in a two-stage flow-shop environment, in Proc. 21st Int. Conf. Comput.Ind. Eng., 269–272.
4. S. J. Tang, B. S. Lee, R. Fan, and B. S. He, 2014, Dynamic job ordering and slot configurations for MapReduce workloads, Tech. Rep.NTU/SCE-2014-4, SCE, NTU, Singapore.
5. S. J. Tang, B. S. Lee, B. S. He, and H. K. Liu, 2014, Long-term resource fairness: Towards economic fairness on pay-as-you-use computing systems, in Proc. 28th ACM Int.Conf. Supercomput.. 251–260.
6. J. Polo, and Y. Becerra, D. Carrera, 2013, Deadline-based MapReduce workload management, IEEE Trans. Netw. Serv. Manage., Vol. 10, No. 2, 231–244.
7. M. A. Rodriguez, and R. Buyya, 2014, Deadline based resource provisioning and scheduling algorithm for scientific work flows clouds,IEEE Trans. Cloud Comput., Vol. 99.
8. J. Polo, C. Castillo, D. Carrera, Y. Becerra, I. Whalley, M. Steinder, J. Torres, and E. Ayguade, 2011, Resource-aware adaptive scheduling for mapreduce clusters, in Proc. 12th ACM/IFIP/USENIX International Conf. Middleware, 187–207.
9. Shanjiang Tang; Bu-Sung Lee, Bingsheng He, 2014, Dynamic MR: A Dynamic Slot Allocation Optimization Framework for Map Reduce Clusters, Cloud Computing, IEEE Transactions on , Vol.2, No.3, 333-347.
10. Sindhuja, P. Synthiya, V. Lavanya, M, Vaithiyanathan, V, 2013, Unsurpassed Resource Superintendence Technique in Cloud Computing Environment, International Journal of Applied Engineering Research, Vol. 8,No 20, 2733.
11. Lavanya M, Dr.V.Vaithiyanathan, S. Saravanan and D.Muthu, 2013, Improved Partitioned Queue Scheduling in Multiprocessor Soft Real Time Systems, International Journal of Applied Engineering Research, Vol. 8, No 20, 2621.