Phase and Heterogeneous Resource aware Scheduler for Map Reduce framework

S.Madhavi1, k.suganthi2

1Student, M.E (II year), Department of CSE, Arasu Engineering College, kumbakonam (TN).
madhavi.div@gmail.com

2Assistant Professor, Department of CSE, Arasu Engineering College, kumbakonam (TN).
pksuganthi@gmail.com

Abstract - Big data is the data sets with the sizes beyond the ability of commonly used software tools to capture, manage, and process data within a tolerable elapsed time. Therefore the big data analytics tools are emerged and one of the popular analytics tool is Map Reduce. Map Reduce is the process of splitting of each job into small map and reduce task and executes them in a parallel manner across number of machines, results in reduction of the running time of data-intensive jobs. For that, the task level scheduling is the most commonly used technique but it gives the sub-optimal job performance. This is because the task level scheduler cannot know to utilize the available resources in an effective way, to reduce the job execution time. Therefore to improve resource utilization, the resource efficient Map Reduce schedulers can be used in recent days. A fine grained resource aware Map Reduce scheduler, that divides the task into various phases, where each phase can have a constant resource usage profile (i.e. homogeneous resource requirement) and performs scheduling at the phase level. But in reality, the phases have highly varying resource requirements during their lifetime. Therefore due to the homogeneous resource allocation to each phase, it is not so efficient to utilize the resources. To address this problem, the proposed scheduler allocates heterogeneous resources to each phase based on their needs and scheduling at the phase level. Therefore it improves resource utilization and thereby reduces the job execution time.

Keywords - Big Data, Map Reduce, scheduling, homogeneous resource allocation, heterogeneous resource allocation.

I. INTRODUCTION

In today’s world large amount of data can be generated and at the same time, it should be needed the new forms of data processing that enable enhanced insight, decision making, cost-effective and process automation. Those data’s are typically called as the Big Data. The characteristics of Big Data can be commonly articulated in terms of 3v’s such as Data Volume, Data Velocity and Data Variety.

Map Reduce is a programming model for reducing large volumes of data into useful aggregated result. The core component of Map Reduce system is Job scheduler. And it is responsible to schedule the map and reduce task, spanning two or more jobs which is used to minimize the job running time and maximizes resource utilization. The scheduling plays a vital role in Map Reduce system because resource contention occurs when scheduling too many concurrently running tasks on a single machine and thereby it increases the job execution time, and at the same time it leads to poor performance when scheduling too few concurrently running tasks on a single machine. Job scheduling become simpler to process when considering the assumption that all map and reduce task having the homogeneous resource requirement. This assumption can be used in Hadoop Map Reduce version 1.x and it provides the simple slot-based resource allocation scheme, where the physical resources on every machine are captured by the number of identical slots that can be allotted to tasks. But in reality, resource consumption can be varied from job to job and task to task during the runtime. Therefore it is failed to reflect these job usage characters and leads to inefficient job scheduling with minimum resource utilization and takes long job execution time.

Based on this observation, a number latest proposals such as Resource-aware adaptive scheduling for Map Reduce clusters [2] and Hadoop Yarn also called Hadoop Map Reduce version 2 or Hadoop NextGen [3], have introduced resource-aware job schedulers to the Map Reduce framework. But these schedulers assume that the run-time resource consumption of task are stable over its life time and therefore provide constant resources to each task. However, this can be fail for many Map Reduce jobs. Because, the execution of Map Reduce task can be further split into various phases of data transfer, processing and storage [5]. A phase is the sub-procedure of task that has a distinct process and can be categorized by the uniform resource consumption over its life time [1]. PRISM: a Phase and Resource Information aware Scheduler for Map Reduce clusters that performs resource-aware scheduling at the level of task phases. This works whenever the job has executed repeatedly with the same input size. But it is failed whenever the input size is varied.

In this paper, Phase and Heterogeneous Resource aware Scheduler for Map Reduce framework has been introduced. This scheduler can perform heterogeneous resource-aware scheduling at the level of each individual phase (for example: map 1, map 2…map n) and at the same time it performs the phase level scheduling to execute the each task phases in order to increase...
the parallel processing. Therefore it improves resource utilization and phase level parallelism to reduce the job execution time.

II. BACKGROUND

The Phases involved in the Execution of a Typical Map Reduce Job are shown in figure 1. Each job can be split into map and reduce task. The Map task consists of two phases such as 1.Map, 2.Merge. The Reduce task consists of three phases such as 3.Shuffle, 4.Sort, 5.Reduce. The Map Reduce task gets the inputs from the HDFS (Hadoop Distributed File System) [4]. In HDFS the input files are split into various data blocks of size 64 MB or 128 MB. And those data blocks can be stored across various data nodes and it can be accessed by using a name node.

The working of phases and their order of execution of the each Map Reduce job is explained as follows:

1. Map: In this phase, the mapper fetches the input data from the HDFS and then performs the map function and stores the result on the buffer. Whenever the buffer becomes full then the content of the buffer is moved to the local disk.

2. Merge: In this phase, the mapper executes merge function to group the results based on their intermediate keys and store the results in many files so that the each file can be taken by a corresponding reducer.

3. Shuffle: In this phase, the reducer taken the input file from the local storage, where the map task result are stored in their earlier phase and it can fed into the storage buffer of the memory or the disk depends on their content size. And also performs local merge sort to reduce the runtime of the subsequent sort phase.

4. Sort: After completing the entire local merge sort, the overall sorting is performed to confirm that all collected results are in order.

5. Reduce: In this phase, the reducer performs the reduce function and the output is stored back to the HDFS (Hadoop Distributed File System).

Different phases can have the varying resource consumption characteristics [1]. The evidence is shown in [1], i.e. resource consumption of runtime task can change significantly across the phase [1]. And at the same time, the runtime resource consumption of each phase can also be varied whenever the different sized input is fetched for processing.

III. SYSTEM ARCHITECTURE

By analyzing the run-time resource usage of task it is well understood that, allocating the fixed size container to each task will results in inefficient scheduling and resource utilization [1]. Resources are wasted when allocating more than the current resource usage and at the same time the resource allocated to the task is less than the current resource usage then the bottleneck occurs and therefore slow down the task execution. This encourage to design a phase level scheduling that allocates the

© 2015 IJAIR. All Rights Reserved
heterogeneous resource to the each phases based on its resource requirements.

The clients send the Map Reduce job request to the resource manager. The resource manager consists of phase level scheduler, resource tracker and application manager. The phase level scheduler receives the heart beat message periodically from the node manager. And the resource tracker is responsible for managing the overall cluster resources. The application manager is responsible for all application master launched in the node manager. Then the application master launches the phases in the node manager and the task metadata can be fetched from the name node and forwards it to the node manager. And whenever the phases need to schedule then the phase level scheduler replies to that heart beat message send by the node manager and perform phase level scheduling. By getting the input data from data node using the metadata and then the phases are executed in the corresponding node manager assigned by the application master. And resources are allocated dynamically to each phase during the runtime based on the heterogeneous resource requirement of each phases. After completing the execution, the results are stored in the data node and the corresponding metadata is also generated.

![System Architecture](image)

**Fig 2: System Architecture.**

### IV. ALGORITHM

**A. Phase-Level Scheduling Algorithm:**

- **Step 1:** Receive the status message from machine N.
- **Step 2:** Get the resource utilization of the machine N.
- **Step 3:** Phase Selected $\leftarrow \{\phi\}$.
- **Step 4:** Candidate Phases $\leftarrow \{\phi\}$.
- **Step 5:** Repeat the above steps.
- **Step 6:** For each job $J \in$ jobs that has the tasks on N do.
- **Step 7:** For each schedulable phase $I \in J$ do.
- **Step 8:** Candidate Phases $\leftarrow$ Candidate Phases $\cup \{I\}$.
- **Step 9:** End for_loop.
- **Step 10:** End for_loop.
- **Step 11:** For each job $J \in$ top K jobs with the highest deficit N do.
- **Step 12:** If exist schedulable data local task then.
- **Step 13:** Candidate Phases $\leftarrow$ Candidate Phases $\cup \{\text{first phase of the local task } I\}$.
- **Step 14:** Else.
- **Step 15:** Candidate Phases $\leftarrow$ Candidate Phases $\{\text{first phase of the non-local task } I\}$.
- **Step 16:** End if.
- **Step 17:** End for_loop.
- **Step 18:** If Candidate Phases $\neq \phi$ then.
- **Step 19:** For $I \in$ Candidate Phases do.
- **Step 20:** If $I$ is not schedulable on N given the current utilization then.
- **Step 21:** Candidate Phases $\leftarrow$ Candidate Phases $\setminus \{I\}$.
- **Step 22:** Continue.
Step 23: End if
Step 24: Compute the utility $U(I, N) = U_{\text{fairness}}(I, N) + \alpha \cdot U_{\text{perf}}(I, N)$
Step 25: If $U(I, N) \leq 0$ then
Step 26: Candidate Phases $\leftarrow$ Candidate Phases $\setminus \{I\}$
Step 27: End if
Step 28: End for loop
Step 29: If Candidate Phases $\neq \emptyset$ then
Step 30: $I \leftarrow$ task with highest $U(I, N)$ in the Candidate Phases
Step 31: Phase Selected $\leftarrow$ Phase Selected $\cup \{I\}$
Step 32: Candidate Phases $\leftarrow$ Candidate Phases $\setminus \{I\}$
Step 33: Update the resource utilization of the machine $N$
Step 34: End if
Step 35: End if
Step 36: Until Candidate Phases $= \emptyset$
Step 37: Return Phase Selected

V. EXPERIMENTAL RESULT
The figure 3, figure 4 show the resource utilization of cluster during the execution of each batch for PRISM and proposed scheduler respectively. The horizontal axis represents the number of jobs and the vertical axis represents the running time in seconds.

Fig 3: Utilization of PRISM.

Fig 4: Utilization of proposed approach.
A. Performance Evaluation:
Analyzing the performance of scheduler’s by comparing the proposed scheduler to the existing scheduler (PRISM). The figure 5, will clearly shows the proposed scheduler works efficiently than the PRISM. I.e. the Memory and Disk utilization is increased in proposed scheduler. Therefore the running time of the proposed scheduler is significantly reduced than the existing scheduler (PRISM).

![Fig 5: Comparison of PRISM and Proposed approach.](image)

VI. RELATED WORK
The novel Hadoop Map Reduce implementation follows the simple slot based resource allocation scheme. And the runtime resource consumption of task cannot be considered here. Therefore several recent works reports the inefficiency due to that such simple allocation scheme, and the proposed solutions. For instance, J.Polo et al. proposed the RAS [2], Resource - aware adaptive scheduling for Map Reduce clusters that used the jobs specific slots for its scheduling. However, it still does the task level scheduling and therefore runtime task resource consumption cannot consider here. Later, Hadoop YARN [3] brings the major effort towards the resource aware scheduling in the Map Reduce clusters. And in Hadoop YARN, it is able to specified fixed size container to each task. The vital challenge is to defining the notion of the fairness whenever the multiple resource type is considered. A.Ghodsi et al. proposed the dominant resource fairness as the measure of fairness in the occurrences of multiple resource types and also provides the simple scheduling algorithm to accomplish nearly-optimal DRF. But it is also still focusing only on the task level scheduling rather on the phase level scheduling.

By using profiles to improve the Map Reduce job performance has receiving the significant attention in recent days [5]. For instance, A.Verma et al. [7] has develop the frameworks that profiles running time of task and at the same time the job profiles to achieving the deadline aware scheduling in Map Reduce clusters. H. Herodotou et al. recently proposed the Starfish: A self-tuning system for big data analytics [5], a job profiler which is used to collect the fine grained task resource usage characteristics and that provides fine-tuned job configuration parameters. T. Condie et al. [9] has developed the Map Reduce online, which is a framework used for stream based processing of Map Reduce jobs. In this model the partial output of each phases is to be sent directly to the subsequent phase, therefore execution of phases can be overlapped.

A.Rasmussen et al. [8] proposed the Themis MR, it is used to modify the Map Reduce phases to improving the I/O efficiency. But this solution cannot be concentrated on scheduling and not providing the resource aware Map Reduce processing.

Qi Zhang et al. [1] has proposed the phase level scheduling, that divides the task into various phases, where each phase can have a constant resource usage profile (i.e. homogeneous resource requirement) and performs scheduling at the phase level. But in reality the phases can have varying resource requirement during runtime. Therefore it leads to ineffective resource aware Map Reduce processing.

VII. CONCLUSION
Map Reduce is a programming model for reducing large volumes of data into useful aggregated result. However, the existing scheduler is only focusing on the task level scheduling. But the fact is when the task divided into phases then it is clearly understood that the resource consumption of each phases can be varied drastically. To address this problem, in the proposed approach, a phase and heterogeneous resource aware scheduler that coordinates the task execution at the phase level and allocating the resource dynamically during the runtime based on phases having the heterogeneous resource requirements. Therefore it improves resource utilization and thereby reduces the job execution time.

REFERENCES


ABOUT THE AUTHOR

S. Madhavi has received B.E degree in the department of Computer Science and Engineering from As-Salam College of Engineering and Technology in the year of 2014. Now she is pursuing M.E degree in the department of Computer Science and Engineering in Arasu Engineering College. Her area of interest is Big Data, Cloud Computing, Operating System, Image Processing and Network Security.

K. Suganthi has received B.Tech in the department of Information Technology from Anjalai Ammal Mahalingam Engineering College in the year 2005 and M.Tech in the department of Computer Science and Engineering from PRIST University in the year of 2014. And now she is working as an assistant professor in Arasu Engineering College and also have 8 year and 2 months of teaching experience in the same Institution. Her area of Interest is Mobile Computing, Network security, Wireless Network and Image Processing.