Enrichment of time efficiency for MapReduce Workloads through Job ordering

Tanmayi Nagale & Rahul Kapse

1, 2 Department of Information Technology, Pillai HOC College of Engineering and Technology, Rasayani

Abstract: In today’s world the quantity of data being generated is growing exponentially. Some of these data are structured, semi-structured or unstructured. This poses a great challenge when these data are to be analyzed because conventional data processing techniques are not suited to handling such data. Map Reduce is a programming model and an associated implementation for processing and generating large data sets. A Map Reduce workload generally contains a set of jobs, each of which consists of multiple map tasks followed by multiple reduce tasks. This system proposes of algorithms to optimize the makespan and the total completion time for an offline MapReduce workload. Our algorithms focus on the job ordering optimization for a MapReduce workload and we can perform optimization of Makespan and total completion for a MapReduce workload. Our work is focuses on resolving the time efficiency problems as well as memory utilization problem. By using MK_TCT_JR algorithm produced the result that are up to, 90 \% better than MK_JR. Our algorithm will improve the system performance in terms of makepan and total completion time.

Keywords: MapReduce, Hadoop, scheduling algorithm, job ordering, makespan, total completion time.

1. Introduction

MapReduce is a used for processing and generating large data sets. Users specify a map function that processes a key/value pair to generate a set of intermediate key/value pairs, and a reduce function that merges all intermediate values associated with the same intermediate key. Workload is referred to as the size of the data for a job. The workload may vary within a job between the different tasks. For example, if we consider a job which consist large data input, such as input every work from a file, and small data output, such as the count of the words. Hadoop, an open source implementation of MapReduce, has been deployed in large clusters containing thousands of machines by companies such as Amazon and Facebook. In those cluster and data center environments, MapReduce and Hadoop are used to support batch processing for jobs submitted from multiple users (i.e., MapReduce workloads). This work tries to improve the performance of MapReduce workloads [1]

There are two key performance metrics i.e. Makespan and total completion time (TCT). Makespan is defined as the time period since the start of the first job until the completion of the last job for a set of jobs. It considers the computation time of jobs and is often used to measure the performance and utilization efficiency of a system. In contrast, total completion time is referred to as the sum of completed time periods for all jobs since the start of the first job. In this works, we aim to optimize these two metrics [1].

When large data sets are given to operate on them for each or every single set of jobs the makespan and total completion time should be able to minimize and optimize the job reduction issues using MapReduce algorithm. This including the total completion time for all datasets should be optimized. To improve the workload distribution among the multiple machines MapReduce Framework is used. To avoid more memory consumption using efficient technique, MapReduce can be used.

2. Literature Review

Shanjiang Tang, Bu-Sung Lee, and Bingsheng has proposed job ordering algorithms that optimize the makespan and total completion time, we also show that they are stable, i.e., the optimized orders produced by job ordering algorithms do not change even if some MapReduce servers fail at execution time.

They also proposed enumeration algorithms for map/reduce slot configuration optimization, we also show that there is a proportional relationship for the optimized map/reduce slot configurations for any two different sizes of total slots. It is important to address the time efficiency problem of the proposed enumeration algorithms for a large-size number of total slots [9].

Wenhong Tian, Guozhong Li, Wutong Yang Rajkumar Buyya has proposed HScheduler: an optimal approach to minimize the makespan of multiple MapReduce jobs. This scheduler works on offline and online jobs [10].

Shanjiang Tang, Bu-Sung Lee, and Bingsheng proposed MOrder: Flexible Job Ordering technique which is used to dynamically optimize the job order
for online MapReduce workloads. MROrder is designed to be flexible for different optimization metrics, e.g., makespan and total completion time. This work improved the system performance by up to 31% for makespan and 176% for total completion time [8].

Shanjiang Tang, Bu-Sung Lee, Bingsheng proposed DynamicMR: A Dynamic Slot Allocation Optimization Framework for improving the performance for a single job but at the expense of the cluster efficiency. They proposed Speculative Execution Performance Balancing technique for balancing the performance tradeoff between a single job and a batch of jobs. They proposed a new technique called Slot PreScheduling that can improve the data locality but with no impact on fairness. Finally, integrating these two techniques, new technique is implemented called DynamicMR that can improve the performance of MapReduce workloads substantially [11].

Abhishek Verma, Ludmila Cherkasova, and Roy H. Campbell have introduced a simple abstraction where each MapReduce job is represented as a pair of map and reduce stage duration. The Johnson algorithm was designed for building an optimal job schedule. This framework evaluates the performance benefits of the constructed schedule through an extensive set of simulations over a variety of realistic workloads. It is achieved up to 10% - 25% of makespan improvements by simply processing the jobs in the right order. They have designed a novel heuristic, called BalancedPools, which improves Johnson’s schedule results (up to 15%-38%), exactly in the situations when it produces suboptimal makespan. Overall, they observed up to 50% in the makespan improvements with the new BalancedPools algorithm [7,12].

G. J. Kyparisis and C. Koukamas considered a scheduling problem in two-stage hybrid flow shop, where the first stage consists of two machines formed an open shop and the other stage has only one machine. The main objective is to minimize the makespan, i.e., the maximum completion time of all jobs. They first show the problem is NP-hard in the strong sense, then we present two heuristics to solve the problem. Computational experiments show that the combined algorithm of the two heuristics performs well on randomly generated problem instances [2].

3. Problem Formulation

In MapReduce Workloads, the job ordering and slot utilization is a challenging task due to fairness and resource requirements. For sequencing of jobs the different schedulers are used. But time efficiency and memory utilization problems are still exists. There are two sub problems of optimization in job ordering module.

1) Find an ordering to execute n number of jobs in so that makespan is minimized under map and reduce slot configuration.
2) Find an ordering to execute jobs that can optimized makespan and total completion time under map and reduce slot configuration.

5. Methodology

5.1 Job ordering optimization Module

1) Assign Workload such as large input file.
2) System will take mapper and reducers depending on the file size.
3) User will assign the jobs such as anagram job, LineCount jobs, CharCount job, WordCount jobs, etc.
4) System will display map phase and reduce phase time as well as job ordering.
5) System will display graph which depicts the improvement in Makespan and total completion time.

The job ordering optimization uses Greedy algorithm based on Johnson’s Rule for sequencing of jobs. System uses Bi-criteria optimization of makespan and total completion time for optimized job ordering and slot configuration. The proposed job ordering algorithm can optimize makespan and total completion time i.e. optimized orders produced by job ordering algorithm do not change even if some MapReduce servers fails execution time.
5. Results and Discussion

From the above study, we have found the following results. The following table shows that required time for two algorithms.

**Table 1. Required time for MK_JR and MK_TCT_JR**

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Minimized Makespan</th>
<th>Minimized Total completion time</th>
</tr>
</thead>
<tbody>
<tr>
<td>MK_JR</td>
<td>1749.2 ms</td>
<td>1924.0 ms</td>
</tr>
<tr>
<td>MK_TCT_JR</td>
<td>3745.6 ms</td>
<td>3194.0002 ms</td>
</tr>
</tbody>
</table>

**Figure 2: Makespan Speedup**

**Figure 3: Total Completion Time Speedup**

In our proposed system we consider large input file as a workload. We take 4 benchmarks such as Word Count, Char Count, Line Count, Anagram jobs. After uploading the file into HDFS and selection of jobs, the MK_JR algorithm will order the jobs. We use another algorithm such as MK_TCT_JR which will speed up the execution of our system. We can compute the make span and total completion time using a simple program we call Mr-Estimator, which simulates the execution of a set of jobs under an ordering. The figure 2 and 3 shows that, larger speedup indicates the better performance it is for the designated job order.

6. Conclusion

Our work is focuses on resolving the time efficiency problems as well as memory utilization problem. Our algorithm will improve the system performance in terms of makepan and total completion. By using MK_TCT_JR algorithm produced the result that are up to, 90 % better than MK_JR. Our algorithm will improve the system performance in terms of makepan and total completion time.

7. Acknowledgements

We thank all the anonymous reviewers and editors for their valuable comments and suggestions to improve the quality of this manuscript.

8. References


