Indian J.Sci.Res. 15 (1): 142-146, 2017
ISSN: 0976-2876 (Print)
ISSN: 2250-0138 (Online)

# PROFILING BASED REDUCE MEMORY PROVISIONING FOR IMPROVING THE PERFORMANCE IN HADOOP

**T. S. NISHA[a1] AND K. SATYANARAYAN REDDY[b]**

[a]Department of CSE, Cambridge Institute of Technology, VTU Belgaum, Bangalore-560036, India
[b]Department of ISE, Cambridge Institute of Technology, VTU Belgaum, Bangalore-560036, India

## ABSTRACT

**Hadoop Map Reduce framework has become a manageable, scalable and fault tolerant framework for processing big data. The number of Map and Reduce task run decides the performance of the big data computing. Usually the number of Map is decided is based on the number of data blocks available for processing, but there is no mechanism to decide the number of reduce jobs. Currently it is based on the user configuration. Like this many challenges exist in Hadoop and Hadoop needs to be optimized. In this survey paper, we propose a profiling based technique to find the optimum number of reduce slots and the amount of memory for reduce, so that when Hadoop is configured with these optimum settings, the job is able to complete successfully.**

**KEYWORDS:** Hadoop Map Reduce, Big Data, Data Blocks.

Hadoop is an open source implementation of Map Reduce framework for Big data processing. In the Map phase, the input data, in the form of data blocks, is processed by Map tasks to generate intermediate data. Each Map task processes one data block. After that, the intermediate data is handled by Reduce tasks of Reduce phase to deliver the final results. Reduce tasks bring the intermediate data chunks to memory to process it. Despite the Map phase, where the number of tasks is determined by the number of data blocks, in Reduce phase the number of tasks can be determined by user or cluster administrator. The ability to manage intermediate data, as well as determination of amount of Reduce tasks, significantly affects the performance. If the memory operations can be managed the performance of big data computing environment can be increased.

Most of previous works on Hadoop optimization has considered only storage or network regarding intermediate data or solely focused on ratio between slots and disregarded slots internal configuration. While saturation of storage IO operations or network bandwidth can lead to performance degradation of Reduce phase, the good news is that the MapReduce application would not be killed in such cases by the Hadoop framework and continues its execution although slowly. However, in case of out of memory error, the job is killed since the Reduce phase needs to bring large portions of intermediate data in to memory for processing. If there is not enough space left in memory, the Reduce tasks and consequently the Reduce phase will fail which leads to job termination by Hadoop. This is a major difference between shortages of memory vs. other resources such as disk/network IO or CPU in MapReduce applications and makes it a significant challenge to conquer. Albeit similar to other resources if the memory becomes the bottleneck, one will face performance degradation even if the job is not killed.

Out of memory is not the only reason for failures of MapReduce jobs; there are also other factors such as disk failure, out of disk, and socket timeouts that might also lead to failure. But such factors are induced by external effects example, by faults in case of disk failure and network timeouts, or lack of enough resources such as disk space. But the problem in deciding the number of reduce is internal to the operation of the application.

To meet the goal of optimizing the execution of Map Reduce applications in the presence of failures, while keeping the impact on the job completion time to the minimum, (D. Moise, T.-T.-L. Trieu, 2011) authors relied on a fault-tolerant, concurrency optimized data storage layer based on the BlobSeer data management service. They tested their approach with 2 distributed file systems: **HDFS** and their BlobSeer- based **BSFS**. In (G. Ruan, H. Zhang, and B. Plale,2013) authors analyzed the HPC platform. Their study examines two types of applications, a 3D-time series caries lesion assessment focusing on large scale medical image dataset, and a **HTRC** word counting task concerning large scale text analysis running on **XSEDE** resources which results demonstrate significant performance improvement in terms of storage space, data stage-in time, and job execution time. In (W. Yu, Y. Wang, X. Que, 2015) authors proposed a novel virtual shuffling strategy to enable efficient data movement and reduce I/O for MapReduce shuffling, thereby reducing power consumption and conserving energy. Their experimental results show that virtual shuffling significantly speeds up data movement in MapReduce

[1]Corresponding author

and achieves faster job execution. In (Y. Chen, A. Ganapathi, and R. H. Katz, 2010) authors developed a decision algorithm that helps MapReduce users identify when and where to use compression. This framework would also facilitate detailed exploration of several compression factors not examined in this work, such as a range of data compressibility, different compression codecs, resource contention between compression and the compute function of maps and reduces, to name a few (Adam Crume , Joe Buck , Carlos Maltzahn , Scott Brandt,2012) Authors proposed SciHadoop a slightly modified version of Hadoop. In Hadoop mappers send data to reducers in the form of key/value pairs. Authors proved that with preliminary designs of multiple lossless approaches to compressing intermediate data, one of which results in up to five orders of magnitude reduction the original key/value ratio.

In (Zhenhua Guo, Geoffrey Fox, Mo Zhou, 2012) authors investigate data locality in depth. They build a mathematical model of scheduling in MapReduce and theoretically analyze the impact on data locality of configuration factors, such as the numbers of nodes and tasks. Secondly, they found the default Hadoop scheduling is non-optimal .In addition, non-optimality of default Hadoop scheduling has been discussed and an optimal scheduling algorithm based on LSAP has been proposed to give the best data locality. Three scenarios – single-cluster, cross cluster and HPC-style setup, have been discussed and real Hadoop experiments were conducted. In (Qi Zhang, 2015) author introduced PRISM, a fine-grained resource-aware MapReduce scheduler that divides tasks into phases, where each phase has a constant resource usage profile and performs scheduling at the phase level. Through experiments using a real MapReduce cluster running a wide-range of workloads, they show PRISM delivers up to 18% improvement in resource utilization while allowing jobs to complete up to 1.3× faster than current Hadoop schedulers. In (M. Zaharia, A. Konwinski, A. D. Joseph, R. H. Katz, 2008) authors analyzed the problem of speculative execution in MapReduce. They designed a simple, robust scheduling algorithm, LATE, which uses estimated finish times to speculatively execute the tasks that hurt the response times the most. In (A. Verma, L. Cherkasova, and R. Campbell,2011) authors introduces a novel framework and technique to address this problem and to offer a new resource sizing and provisioning service in MapReduce environments. They validated the accuracy of models using a set of realistic applications. The predicted completion times of generated resource provisioning options are within 10% of the measured times in our 66-node Hadoop cluster. In (B. Nicolae, D. Moise, G. Antoniu,2010) authors substitute the original HDFS layer of Hadoop with a new, concurrency-optimized data storage layer based on the BlobSeer data management service. Thereby, the efficiency of Hadoop is significantly improved for data-intensive Map-Reduce applications. This work demonstrates that it is possible to enhance it by replacing the default Hadoop Distributed File System (HDFS) layer by another layer, built along different design principles introduced BlobSeer system, which is specifically optimized data accessed under heavy concurrency along with additional features such as efficient concurrent appends toward efficient, fine-grain access to massive, distributed, concurrent writes at random offsets and versioning.

In this paper, we analyze the challenges like this which are caused due internal operations and explore different solutions to solve them. We identify the pros and cons of each solution and finally propose a reduce memory optimization solution to execute the job successfully.

Our solution is based on the concept of linear modeling based profiling. The job is first executed with different size of data and the intermediate data generated is found. Based on this a linear modeling is done between the input data size and the intermediate data size, based on this, the number of reduce memory needed for a large input dataset is calculated prior and the number of reduce is provisioned accordingly. By this way, the number of reduce jobs and the reduce memory is provisioned prior and the job is able execute successfully with better CPU utilization.

We implement the proposed solution and measure the performance in terms of CPU Utilization with the default .Hadoop implementation and prove that our solution has better CPU utilization than default Hadoop implementation.

## PROPOSED SOLUTION

The architecture of the proposed solution is shown in Figure 1

Profiler takes the Job jar file as input and test run the jar file for various input size on the Hadoop environment and measure the size of intermediate data generated.

Optimizer module takes the input file which the Job jar wants to execute on as input and calculate the optimum number of reduce slots and the reduce

memory that will be consumed and sets it on Hadoop environment. So that when the job jar is executed for the input file, it will execute in the optimum configuration and execute successfully.

The implementation steps are shown in Figure 2.

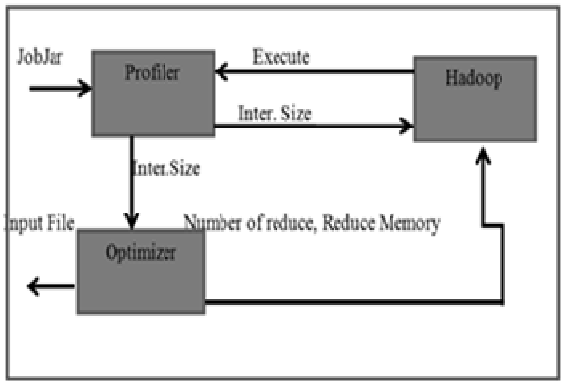


**Figure 1: Architecture of the proposed solution**

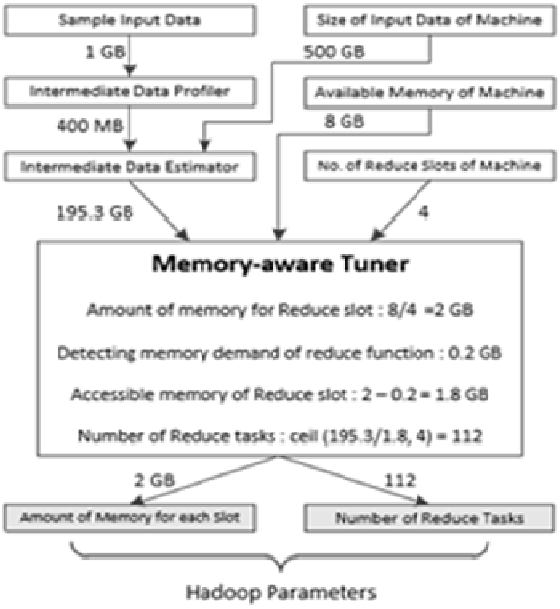


**Figure 2: Implementation Steps of the Proposed Solution**

## CLASS DIAGRAM

The class diagram are shown in Figure 3

## RESULTS

We implemented the proposed solution in Hadoop and measured two parameters

1. Execution time
2. CPU Utilization

By varying the input file size, we measured these 2 parameters and compared it with Hadoop without optimization and the result is shown in graph as Figure 4 & 5.

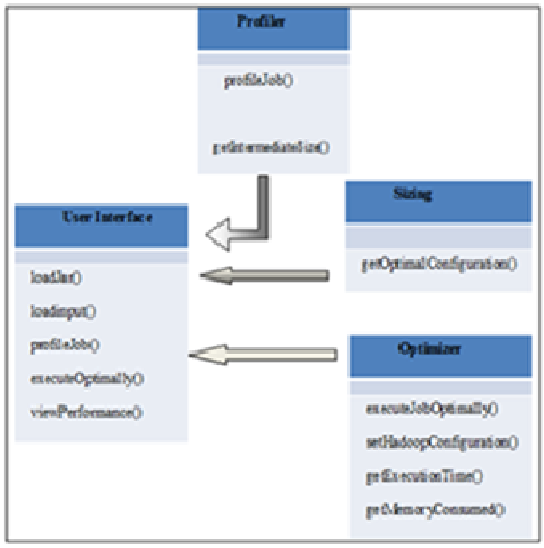


**Figure 3: Class Diagram of the Proposed Solution**

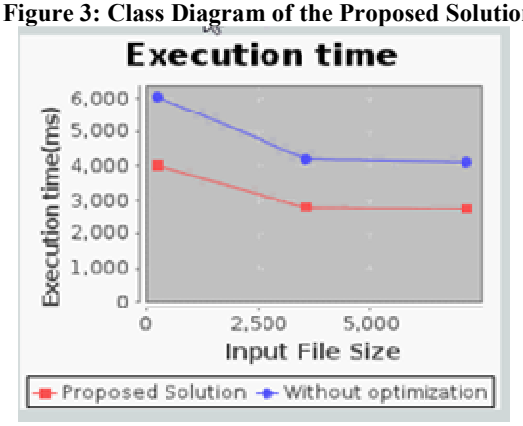


**Figure 4: Execution Time**

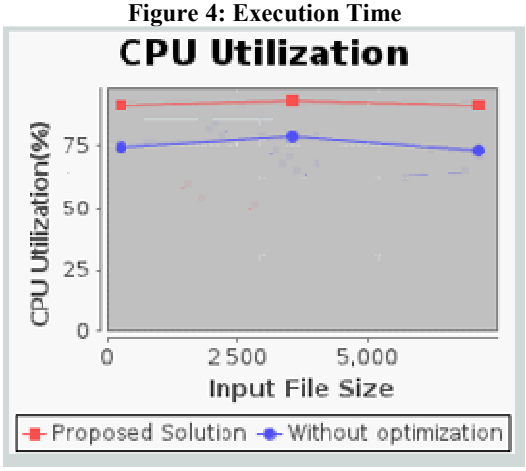


**Figure 5: CPU Utilization**

## APPENDIX

The Summary of Survey Paper listed below

| Paper | Advantages | Disadvantages |
|---|---|---|
| [1] | Able to handle Mapper failure and support fault tolerance | Not scalable |
| [2] | Job execution time is faster | Application specific only works well for mathematics and scientific applications |
| [3] | Achieves speed up by using shuffling also saves power consumption | In case of high volume of intermediate key values pairs the overhead is very high for shuffling |
| [4] | Uses compression codes to compress intermediate key value and optimizes memory consumption | Workload characteristics must be known priori in this approach. |
| [5] | Applied compression to reduce the volume of key – value pair | Execution time over head due to decompression is high |
| [6] | Achieves optimization based on locality | Not scalable for multi cluster setup |
| [7] | By applying phase wise scheduling speed up is achieved | The heuristics needed to split for phases is not well defined |
| [8] | Applied speculation to speed up the execution | Resource overhead is high |
| [9] | Resource are reserved and job execution is started | The efficiency of system is better only after long duration of profiling |

## CONCLUSION AND FUTURE ENHANCEMENT

We have implemented proposed Hadoop reduce memory optimization based on profiling and sizing. We executed the proposed solution for different volume of input file and showed that proposed system has reduced execution time when compared to non-optimized Hadoop. Since the system is able to provide dimension values well ahead of execution, the system administrator can also scale up the system if sufficient resources are available.

In the base paper estimation of intermediate is done based on one time profiling. But this is not valid for some tasks where the number of intermediate generated is different for different volume of data. To solve this problem, linear modeling is done for different volume of data and the intermediate size is found for each input data volume. After finding it, least square estimation is done to find the best fit for estimation of intermediate. Using this kind of modeling yielded best estimation of intermediate size.

## REFERENCES

Moise D., Trieu T.-T.-L., Boug´e L. and Antoniu G., 2011. "Optimizing intermediate data management in map reduces computations," in Proceedings of the first international workshop on cloud computing platforms, pp. 1–7. ACM.

Ruan G., Zhang H. and Plale B., 2013. "Exploiting map reduces and data compression for data-intensive applications," in Proceedings of the Conference on Extreme Science and Engineering Discovery Environment: Gateway to Discovery. pp. 1–8, ACM.

Yu W., Wang Y., Que X. and Xu C., 2015. "Virtual shuffling for efficient data movement in map reduce," IEEE Transactions on Computers, **64**(2):556–568.

Chen Y., Ganapathi A. and Katz R.H., 2010. "To compress or not to compress-compute vs. io tradeoffs for map reduce energy efficiency," in Proceedings of the first ACM SIGCOMM workshop on Green networking. pp. 23–28, ACM.

Adam C., Joe B., Carlos M. and Scott B., 2012. Compressing Intermediate Keys between Mappers and Reducers in SciHadoop, Proceedings of the 2012 SC Companion: High Performance Computing, Networking Storage and Analysis, pp.7-12. [doi>10.1109/SC.Companion.2012.12]

Zhenhua G., Geoffrey F. and Mo Z., 2012. Investigation of Data Locality in MapReduce,

Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012), pp. 419-426. [doi>10.1109/CCGrid.2012.42].

Qi Z., 2015. "PRISM: Fine-Grained Resource-Aware Scheduling for MapReduce", pp.4-8, IEEE Transactions on Cloud Computing.

Zaharia M., Konwinski A., Joseph A. D., Katz R. H. and Stoica I., 2008. Improving map reduce performance in heterogeneous environments. In USENIX Symposium on Operating Systems Design and Implementation (OSDI), **8**:7-8.

Verma A., Cherkasova L. and Campbell R., 2011. Resource Provisioning Framework for Map Reduce Jobs with Performance Goals. ACM/IFIP/USENIX Middleware, pp. 165–186.

Nicolae B., Moise D., Antoniu G. and al. BlobSeer, 2010. Bringing high throughput under heavy concurrency to Hadoop Map/Reduce applications. In Procs of the 24th IPDPS 2010, In press.

Nisha T.S. and Reddy K.S., 2017. "A Survey On Optimization in Hadoop Big data Environment," in IJRDT, **7**(6).