Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X
IMPACT FACTOR: 6.017

IJCSMC, Vol. 7, Issue. 7, July 2018, pg.83 – 87

BigData Processing using First Come First Served (FCFS) Algorithm

M Vamshi Krishna
M.Tech, Department of Computer Science and Engineering, JNTUH, HYDERABAD, INDIA

Abstract: ***MapReduce framework has become the de facto standard for large scale data-intensive applications. It shows the experiment results on a cluster of computers. It also discusses which the right tool for the jobs is by analyzing the characteristics and performance of the paradigms.*** *MapReduce – is a software framework that allows developers to write programs that process massive amounts of unstructured data in parallel across a distributed cluster of processors or stand-alone computers.* ***MapReduce analysis process management system newly proposed in this paper is so designed as to perform MapReduce job in a processes management application to utilize a BPM (BigData Process Management) engine. We evaluate with different data set(BigData) understand that FCFS is the most suitable algorithm for analysis of data being efficient execution of data set and store the data after analysis.***

INTRODUCTION

MapReduce is a programming model and an associated implementation for processing and generating large datasets that is amenable to a broad variety of real world tasks. It is designed to make it possible to process an intelligent data analysis by controlling the conditions between diversified MapReduce jobs. We often happen to meet problems requiring heavy computations or data-intensive processing. Hence, on one hand, we try to develop efficient algorithms for the problems. On the other hand, with the advances of hardware and parallel and distributed computing technology, we are interested in exploiting high performance computing resources to handle them. Efficient data refinement and transmission are also possible by utilizing Process Management in order to perform any MapReduce job scheduled. Qualitative pros and cons of each framework are known, but

quantitative performances indexes help get a good picture of which framework to use for the applications.

A variety of software framework has been developed to take advantage of hardware capability and to effectively develop parallel and distributed applications. With the plentiful frameworks of parallel and distributed computing, it would be of great help to have performance comparison studies for the frameworks we may consider.

This paper is concerned with performance studies of various process frameworks like FCFS, SJF, Round Robin and Priority Scheduling algorithms. The comparative studies have been conducted for mane set of processes the all-pairs-shortest-path problem and a join problem for large data sets. MapReduce is recognized as the standard framework intended for big data processing. For each problem, the parallel programs have been developed in terms of the three models, and their performance has been observed.

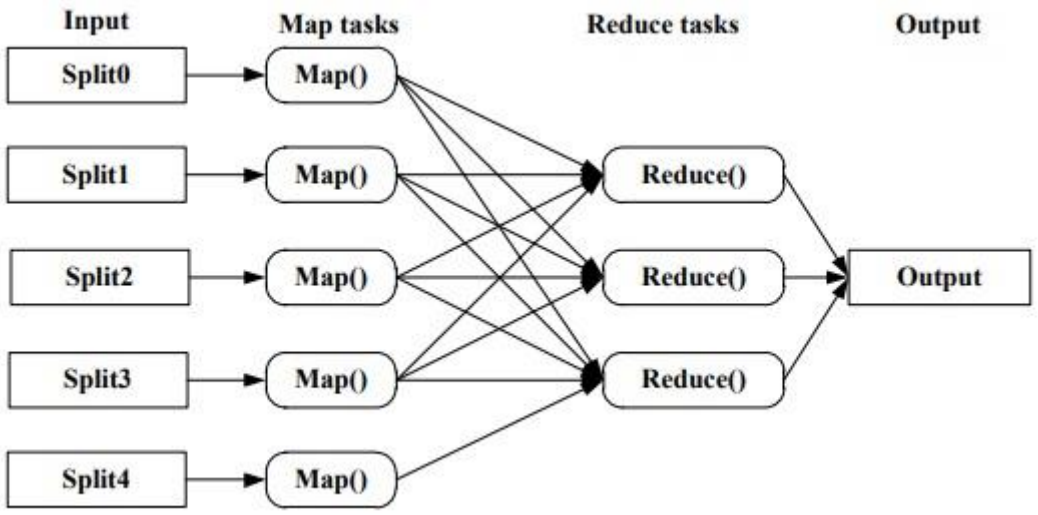
The remainder of the paper is organized as follows:

Section 1: by taking the inputs as processes in the queue form.

Section 2: briefly reviews all the papers and applies the Map-function to the queue of processes. **Section 3:** Then the mapped processes are together formed as sub-processes presents the selected programming frameworks in more detail.

Section 4: Those combined sub-processes are applied by Reduce- function so that it would be reduced by similarities of processes.

Section 5: Finally the FCFS algorithm is applied to newly formed queue.



RELATED WORK

MapReduce is a programming paradigm to use Hadoop which is recognized as a representative big data processing framework [11]. Hadoop clusters consist of up to thousands of commodity computers and provide a distributed file system called HDFS which can accommodate big volume of data in a fault-tolerant way. The clusters become the computing resource to facilitate big data processing.

MapReduce organizes an application into a pair (or a sequence of pairs) of Map and Reduce functions. It assumes that input for the functions comes from HDFS file(s) and output is saved into HDFS files. Data files consist of records, each of which can be treated as a key-value pair. Input data is partitioned and processed by Map processes, and their processing results are shaped into key-value pairs and shuffled into Reduce tasks according to key. Map processes are independent of each other and thus they can be executed in parallel without collaboration among them. Reduce processes play role of aggregating the values with the same key. MapReduce runtime launches Map and Reduce processes with consideration of data locality. The programmers do not have to consider data partitioning, process creation, and synchronization. The same Map and Reduce functions are executed across machines. Hence, MapReduce paradigm can be regarded as a kind of SPMD model.

MapReduce paradigm is a good choice for big data processing because MapReduce handles data record by record without loading whole data into memory and in addition the program is executed in parallel over a cluster [20]. It is very convenient to develop big data handling programs using MapReduce because Hadoop provides everything needed for distributed and parallel processing behind the scene which program does not need to know. MapReduce allows programmers with no experience in parallel and distributed systems to easily utilize the resources of a large distributed system. Typical MapReduce computation processes many terabytes of data on hundreds or thousands of machines. Programmers find the system easy to use, and more than 100,000 MapReduce jobs are executed on Google‟s clusters every day [2].

Conceptually the map and reduce functions supplied by a user have associated types as follows-

$$map\ (k_1, v_1) \rightarrow list\ (k_2, v_2)$$

$$reduce\ (k_2, list\ (v_2)) \rightarrow list\ (v_2)$$

First-In-First-Serve:

FCFS also termed as First-In-First-Serve i.e. allocate the CPU in order in which the process arrive. When the CPU is free, it is allowed to process, which is occupying the front of the queue. Once this process goes into running state, its PCB is removed from the queue. This algorithm is non-preemptive. It is a non preemptive scheduling algorithm. Here processes are kept in a queue and are executed one by one. Whenever a new process arrives it is kept at the end of the queue. So, this is the last process in the queue. Now, when a new process arrives after this process, the new process becomes the last process in the queue. When a process which is running is blocked the next process in the queue is run and the blocked process (when it is ready) is placed at the end of the queue. FCFS is usually used in batch systems. The advantage of FCFS algorithm is that it is very easy to implement. The disadvantage of FCFS algorithm is that since it is non preemptive, CPU usage can be wasted in some situations.

CONCLUSION

The processed data is stored in the form BigData can be performed a scheduling algorithm which is most efficient and effective. For the future work various data set of different data size can be used for performance analyzing and assessment. Hence dynamic big data resulted its performance on varying data set size and the availability of resources and computation capability. Hence dynamic big data after the data analysis the data is stored in encrypted form in HDFS. Dynamism associated with big data can have huge influence on scheduling algorithm with different size of data set. For future work other scheduling algorithm can be applied to the dynamic data and size of the input data size can also be enhanced.

References

1. D. E. Culler, J. P. Singh, and A. Gupta, Parallel Computer Architecture: A Hardware/software Approach, Gulf Professional, 1999.
2. H. Lee-Kwang, K. A. Seong, and K. M. Lee, "Hierarchical partition of nonstructured concurrent systems," IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics, vol. 27, no. 1, pp. 105–108, 1997.View at Publisher · View at Google Scholar · View at Scopus
3. S. W. Lee, J. T. Kim, H. Wang et al., "Architecture of RETE network hardware accelerator for real-time context-aware system," Lecture Notes in Computer Science, vol. 4251, pp. 401–408, 2006. View at Google Scholar · View at Scopus
4. S. W. Lee, J. T. Kim, B. K. Sohn, K. M. Lee, J. W. Jeon, and S. Lee, Real-Time System-on-a-Chip Architecture for Rule-Based Context-Aware Computing, vol. 3681 of Lecture Notes in Computer Science, 2005.
5. J. Diaz, C. Muñoz-Caro, and A. Niño, "A survey of parallel programming models and tools in the multi and many-core era," IEEE Transactions on Parallel and Distributed Systems, vol. 23, no. 8, pp. 1369–1386, 2012. View at Publisher · View at Google Scholar · View at Scopus

6. S. C. Ravela, "Comparison of shared memory based parallel programming models," Tech. Rep. MSC-2010-01, Blekinge Institute of Technology, 2010. View at Google Scholar
7. OpenMP Architecture Review Board, "OpenMP Application Program Interface," 2008, http://www.openmp.org/mp-documents/spec30.pdf.
8. W. Gropp, S. Huss-Lederman, A. Lumsdaine et al., MPI: The Complete Reference, the MPI-2 Extensions, vol. 2, The MIT Press, 1998.
9. J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," Communications of the ACM, vol. 51, no. 1, pp. 107–113, 2008. View at Publisher · View at Google Scholar · View at Scopus
10. B. Barney, Introduction to Parallel Computing, Lawrence Livermore National Laboratory, White,
11. T.: Hadoop: The Definitive Guide. O"Reilly, Sebastopol (2009)
12. Ghemawat, S., Gobioff, H., Leung, S.: The Google File System. In: Symposium on Operating Systems Principles, pp. 29--43 (2003)
13. Hadoop Distributed file system, http://hadoop.apache.org
14. 6. Wil, M. P., Arthur, H. M., Mathias, W.: Business Process Management: A survey. In: Lecture Notes in Computer Science, LNCS, vol. 2678, pp. 1--12 (2003)
15. M. Macedonia, "The GPU enters computing's mainstream," Computer, vol. 36, no. 10, pp. 106–108, 2003.View at Publisher · View at Google Scholar · View at Scopus
16. A. Alexandrov, S. Ewen, M. Heimel et al., "MapReduce and PACT—comparing data parallel programming models," in Proceedings of the 14th Conference on Database Systems for Business, Technology, and Web (BTW '11), pp. 25–44, 2011.
17. M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Feterly, "Dryad: distributed data-parallel programs from sequential building blocks," ACM SIGOPS Operating Systems Review, vol. 41, no. 3, pp. 59–72, 2007. View at Google Scholar

