ISSN 2278-9723
DOI:10.31838/ijccts/08.01.02

Research Article

The Use of Deep Data Locality towards a Hadoop Performance Analysis Framework

WICKRAMASINGHE.K
Information Institute of Technology, Colombo, Sri Lanka
E-mail: Wickram@gmail.com

Received: 25.10.19, Revised: 27.11.19, Accepted: 15.01.20

ABSTRACT

In big data systems, one of the base models in the recent past entails Hadoop, which holds that it is cheaper to move computation, compared to the decision to move data. Indeed, using Hadoop comes with an increase in the HDFS system's data locality, translating into improved system performance. Also, in bid data systems, there is a reduction in the network traffic between or among the selected nodes due to machine data-local increase. Currently, however, a mathematical performance framework for Hadoop data locality is yet to be established. In this study, a framework for analyzing Hadoop performance was proposed based on data locality, seeking to ensure that the MapReduce procedure is analyzed in the entirety. The objective was to discern the extent to which the framework for analyzing Hadoop performance could be applied towards an improvement in the Hadoop system performance, especially through the making of a deep data locality. In the findings, it was established that when three tests in the form of a physical test, a cloud test, and a simulation base test are applied, the deep data locality approach yields a significant improvement in the Hadoop performance. Particularly, the use of the deep data locality technique led to a 34 percent improvement in the Hadoop system. Thus, it was concluded that the superiority of the proposed approach arises from its ability to yield a reduction in the HDFS data movement.

Keywords: deep data locality, HDFS, data locality, Hadoop, MapReduce

INTRODUCTION

In the contemporary world, there is a significant increase in the volume of global data [1]. Hence, there is growing demand for big data analytics that could be deemed more efficient. During the bid data processing, therefore, one of the platforms of frameworks that have played an essential role is the case of Apache Hadoop. Indeed, this model relies on the phenomenon in which the aim is to move computation, an option deemed cheaper compared to the movement of data [2, 3]. In traditional systems of data processing, the arrangements hold that there is the movement of the target data to the servers for purposes of processing. However, some studies affirm that this approach ends up creating data transfer bottlenecks [4, 5].

A practical illustration demonstrating the limitation of the traditional systems is that in which a speed of 100 MB/s was used during the coping of 1 TB data on some hard drive, which requires about three hours before completing [6]. Even in situations where there is the division of such data in terms of 100 hard drives, the decision to copy 10 GB to a selected server would require close to 100 seconds [7, 8]. To address such a bottleneck, Hadoop can be seen to send codes to the target servers, ensuring that the data transfer overhead is either removed or reduced [9, 10]. The latter process translates into a state of data locality, aimed at enhancing the Hadoop performance significantly.

The eventuality is that most studies have strived to improve on the data locality concept using approaches such as cluster management, system configuration, and scheduling. However, a certain links still misses in these approaches, coming in the form of the failure to maintain high data locality in the entire MapReduce (MR) execution process. This study proposed a DDL (deep data locality) framework aimed at extending the superior performance of data locality to all MR stages.

METHODOLOGY

As mentioned above, a novel approach was proposed to add to the superior performance of all MR stages' locality. The motivation lay in the criticality of minimizing the shuffle's overhead. Indeed, the proposed DDL approach sought to deviate from the map-only locality approach. It is also notable that two types of DDL were investigated in relation to the ability to extend the locality to all MR stages. One of them entailed the block-based DDL aimed at reducing data transfer and reducing the RLM in multicore processors. Importantly, when the multicore processors are employed, they imply that all cores share common local disks; suggesting that between the cores, there is no data transfer overhead. Another DDL method involved the key-based DDL technique, which functioned by pre-arranging elements of data within the input data based on their keys. In so doing, only data elements meant for the given reducers would be assigned to the same mappers; preventing data transfer. The data element pre-arrangement would also be accomplished before the map stage. The following figure demonstrates the correlation between the key-based DDL, the block-based DDL, and the SDL (traditional map-only locality) relationship.

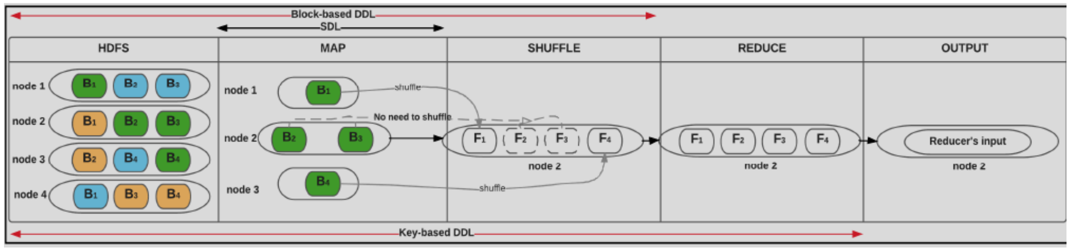


Figure 1: Locality relationships (key-based DDL, block-based DDL, and SDL)

RESULTS AND DISCUSSION

This experimental study involved the testing of the performance of Hadoop in three environmental scenarios. The objective lay in the comparison of the performance between traditional data locality and DDL. Also, there was a simulation to discern the proposed model's performance in different conditions. The collection of the experimental data was achieved using a small testbed in hardware.

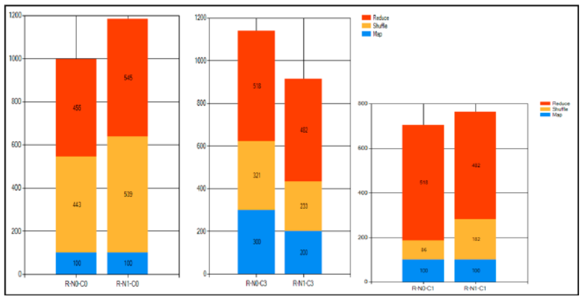
Initially, there was the testing of the performance of Hadoop on cloud. The three sets of data on the focus included two 120 GB data, two 60 GB data, and two 30 GB data. To achieve outcome reliability and validity, the execution of the test occurred ten times before averaging the outcomes. From the results, the use of the block-based DDL approach yielded a significant reduction in shuffle time, ranging between 18 percent and 30 percent.

Table 1: Experimental results between DDL and Hadoop Default performance

	Map (s)	Shuffle (s)	Total (min)		Map (s)	Shuffle (s)	Total (min)		Map (s)	Shuffle (s)	Total (min)
Default-Job1-30G	26	463	27	Default-Job1-60G	25	929	71	Default-Job1-120G	28	2345	165
Default-Job2-30G	30	431	27	Default-Job2-60G	27	906	76	Default-Job2-120G	32	1968	165
LNBPP-Job1-30G	22	292	21	LNBPP-Job1-60G	24	721	59	LNBPP-Job1-120G	24	1475	150
LNBPP-Job2-30G	20	328	25	LNBPP-Job2-60G	22	771	62	LNBPP-Job2-120G	25	1647	150
Decrease Time (%)	25%	30.7%	14.8%	Decrease Time (%)	11.5%	18.7%	17.7%	Decrease Time (%)	18.3%	27.6%	9%

With RLM eliminated, there was also a notable improvement in performance in the map. For Block-based DDL, the decreased time reduced to 17.7 percent, a decrease by 9%.

The next phase involved testing the performance of Hadoop on a simulator. From the literature, some of the factors affecting the performance of Hadoop include key distribution on blocks, data size, and the composition of Hadoop cluster. In this case, a simulator was developed in a quest to determine the performance of the proposed framework (compared to other models) under different conditions. At this stage, this study established that upon using key-based DDL and block-based DDL in combination, the system's performance was superior to the performance scenario involving the default MR. The combined arrangement outperformed the default MR by 35%. Hence, the results were encouraging.



(a) Default (b) block-based DDL (c) block & key-DDL

Figure 2: Simulator results comparison

In a quest to verify the accuracy of the simulation and analytical models that were employed, this experimental investigation proceeded to test the performance of data locality relative to hardware implementation. Hence, there was the configuration of a small Hadoop cluster in which five machines were used, with four of them meant for slave nodes while one of them was meant for the master node. From the figure that follows, there was an improvement in performance with the addition of more data locality. Compared to the case of the default MR, there was an increase in speed by 21.9% when key-based DDL was employed. For the case of the block-based DDL, the increase in speed stood at 9.8%. At a point where there was a combination of key-based and block-based DDL schemes, the increase in the speed of performance stood at 34.4% compared to the case of the default MR.

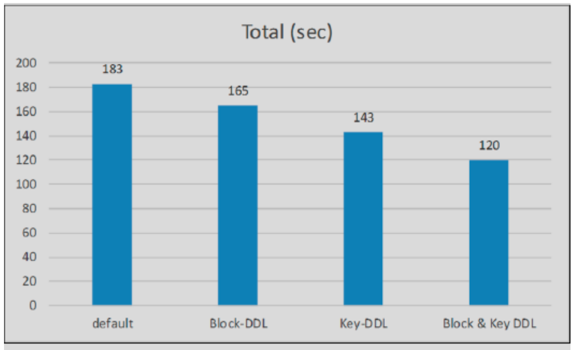


Figure 3: A comparison of the total time for the performance of different schemes

Table 2: Hadoop performance improvement for different schemes

	Block-DDL	Key-DDL	Block& Key-DDL
Default	9.8%	21.9%	34.4%
Block-DDL	N/A	13.3%	27.3%
Key-DDL	N/A	N/A	16.1%

CONCLUSION

This study focused on the performance analysis framework for Hadoop, as well as HDFS data locality concept. The objective of the study was to steer improvements in Hadoop system performance. The two DDL methods that were implemented included key-based DDL and block-based DDL. The implementation involved a combination of these two schemes on HDFS, a decision that saw an increase by over 34.4% in system performance compared to a scenario entailing the default MR. tested on the physical implement Hadoop system, Hadoop simulation, and a cloud, the two schemes increased the performance of Hadoop by 21.9% and 9.8% for the key-based DDL and the block-based DDL (more than the default MR) respectively. When the schemes were combined, compared to the default approach, there was an increase in Hadoop performance by 34.4%.

REFERENCES

1. Basak, A.; Brnster, I.; Ma, X.; Mengshoel, O.J. Accelerating Bayesian network parameter learning using Hadoop and MapReduce. In Proceedings of the 1st International Workshop on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications, Beijing, China, 12 August 2012; pp. 101–108.
2. Iwazume, M.; Iwase, T.; Tanaka, K.; Fujii, H.; Hijiya, M.; Haraguchi, H. Big Data in Memory: Benchimarking In Memory Database Using the Distributed Key-Value Store for Machine to Machine Communication. In Proceedings of the 15th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, Beijing, China, 30 June–2 July 2014; pp. 1–7.
3. Membrey, P.; Chan, K.C.C.; Demchenko, Y. A Disk Based Stream Oriented Approach for Storing Big Data. In Proceedings of the 2013 International Conference on Collaboration Technologies and Systems, San Diego, CA, USA, 20–24 May 2013; pp. 56–64.
4. Islam, N.S.; Wasi-ur-Rahman, M.; Lu, X.; Shankar, D.; Panda, D.K. Performance Characterization and Acceleration of In-Memory File Systems for Hadoop and Spark Applications on HPC Clusters. In Proceedings of the International Conference on Big Data, IEEE Big Data, Santa Clara, CA, USA, 29 October–1 November 2015; pp. 243–252.

5. Lee, S. Deep Data Locality on Apache Hadoop. Ph.D. Thesis, The University of Nevada, Las Vegas, NV, USA
6. Lee, S.; Jo, J.-Y.; Kim, Y. Survey of Data Locality in Apache Hadoop. In Proceedings of the 4th IEEE/ACIS International Conference on Big Data, Cloud Computing, and Data Science Engineering (BCD), Honolulu, HI, USA, 29–31 May 2019.
7. Lee, S.; Jo, J.-Y.; Kim, Y. Key based Deep Data Locality on Hadoop. In Proceedings of the IEEE International Conference on Big Data, Seattle,WA, USA, 10–13 December 2018; pp. 3889–3898.
8. Lee, S.; Jo, J.-Y.; Kim, Y. Performance Improvement of MapReduce Process by Promoting Deep Data Locality. In Proceedings of the 3rd IEEE International Conference on Data Science and Advanced Analytics (DSAA), Montreal, QC, Canada, 17–19 October 2016; pp. 292–301.
9. Elshater, Y.; Martin, P.; Rope, D.; McRoberts, M.; Statchuk, C.AStudy of Data Locality in YARN. In Proceedings of the IEEE International Congress on Big Data, Santa Clara, CA, USA, 29 October–1 November 2015; pp. 174–181.
10. Qin, P.; Dai, B.; Huang, B.; Xu, G. Bandwidth-Aware Scheduling with SDN in Hadoop: A New Trend for Big Data. IEEE Syst. J. 2015, 11, 1–8.
11. Prasad, D.S., Kabir, Z., Dash, A.L., Das, B.C.Prevalence and risk factors for metabolic syndrome in Asian Indians: A community study from urban Eastern India(2012) Journal of Cardiovascular Disease Research, 3 (3), pp. 204-211.
DOI: 10.4103/0975-3583.98895