Fig. 7. Normalized execution time with NoFLA as base for normalization.

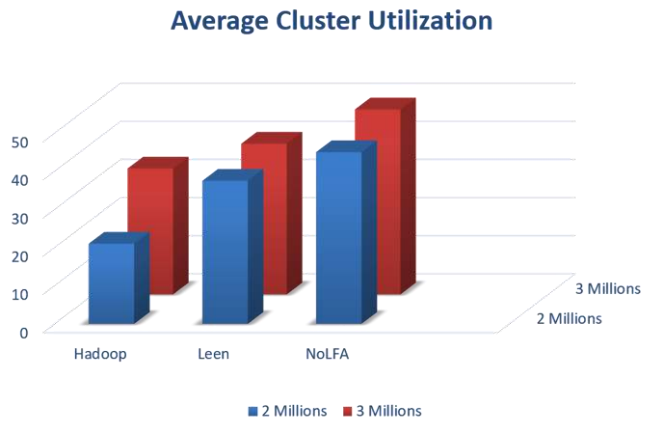

Fig. 8. Average performance gain of Hadoop, Leen, and NoLFA. Results are normalized according the number of key-value pairs processed by each scheme accordingly.

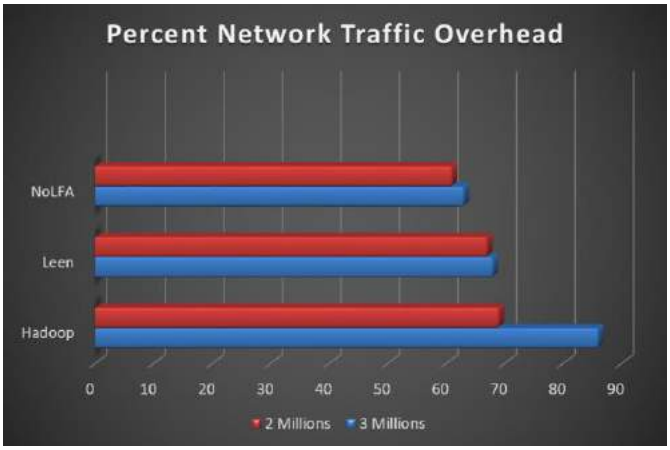

Fig. 9. Percent Network Traffic Overhead vs Numbers of Key-value pairs. The upper red bars shows the value for 2 million key-value pairs while the lower blue bar show it for 3 million key-value pairs to process.

for Leen with the number of key-value pairs set to 2 million and 3 million; and for NoLFA with the key-value pairs set to 2 million and 3 million, accordingly. X-axis shows the average percent network traffic overhead caused in each instance of experiment. Y-axis shows different partitioning techniques used in the model experiments. The top bottom blue bar shows the case when the number of key-values pairs to be processed is 3 million, whereas the top red bar represent the case of 2 million key-value pairs to be processed, accordingly. The results shows that the network overhead increases as the amount of data need to be processed increases in each instance of experiment.

With the above reasoning, we claim that it is clear that capacity awareness play an important role in selection of the partitioning different keys to different nodes in the cluster, and has a positive influence on the overall performance and near optimal utilization of the cluster.

VI. Related Work

Previous work aiming to improve the performance of MapReduce system achieved the desired goal through various approaches including reduction of network cramming by inserting partial data awareness into the shuffle phase, skew mitigation, replica awareness, and network awareness.

Authors in [18] proposed two schemes of pre-fetching and pre-shuffling for communal MapReduce environments. Pre-fetching use data locality and assign tasks to nearest node to the data block, whereas pre-shuffling reduce network overhead of slouching the key-value pairs. Our scheme NoLFA decouple the mapper and reducer tasks and scan over the keys frequency table generated upon execution of map phase and cross reference it with the capacity table created after executing the sample jobs on the nodes in the cluster to achieve the goal of partial balanced reduce tasks throughout the cluster. ShuffleWatcher [19] proposed a multi-tenant Hadoop scheduler that tries to curtail the network traffic in shuffle phase while maintaining the particular fairness constraints of the system. The working principle of the ShuffleWatcher is on the basis of the following three steps. First, it limit the intra-job map shuffle according to the network traffic load. Second, it auspiciously apportion the map tasks to localize the intermediate data. Finally, it exploit the confined intermediate data and delayed shuffle to reduce the network traffic in shuffle phase by favorably scheduling reduce tasks in nodes crofting the intermediate data. Unlike ShuffleWatcher, NoLFA take the capacity information of each node in the cluster whereas distributing the tasks which is very helpful in case of the heterogeneity in the cluster. EC-Cache [20] introduced a load-balanced, low latency cluster cache via erasure coding to overawed the inadequacy of selective replication. It employs erasure coding through two principles. First, by splitting and erasure coding individual objects during writes. Second, late binding. These led to improving load-balancing in the system. Tang et al. proposed a sampling evaluation to solve the problems of partitioning skew and intermediate data locality for the reduce tasks called Minimum Transmission Cost Reduce Task Scheduler [21] (MTCRS). They used communication cost and waiting time of each reduce task as heuristic whereas deciding which task to assign to which node in the cluster. Their scheduling algorithm used Average Reservoir Sampling for the spawning of parameter sizes, and location of intermediate data partitions for their rummage-sale mathematical estimation model. On the other hand, NoLFA used Random Sampling.

Transferring data over the network is costly and causes performance degradation more severely in federated clusters. Kondikoppa et al. [22] introduced a network-aware scheduling algorithm for Hadoop system which work in federated clusters, improving the map tasks scheduling and

consequently tries to abate the network traffic overhead leading to improved performance gain. NoLFA has different approach of decoupling the map and reduce tasks and routinize the keys-capacity frequency table to achieve the specified goal. Locality Aware Reduce Scheduling (LARS) [23] abate data transfer in their proposed grid-enabled MapReduce framework. Due to heterogeneity awareness of nodes in the grid, the map data size varies leading to assigning map tasks associated with different data size to different worker nodes according to their computation power. The LARS algorithm will select the nodes with largest region size of the intermediate data to be the destination for the reduce tasks. NoLFA achieve the desired goal with the frequency-capacity table.

Another concern is the partitioning skew that ascends due to an unstable distribution of map output across nodes, causing a massive size of data input for some reduce tasks while lesser for others. Centre-of-Gravity (CoG) [24] reduce scheduling add locality and skew cognizance to the scheduler. They allocates the reduce tasks to nodes nearer to nodes creating the intermediate data for that listed reduce tasks. SkewReduce [25] was proposed with the intention to dazed the computation skew in MapReduce systems where the partition run time depends on the data values as well as input size. It uses a user defined cost function based optimizer to regulate the partitioning parameterization of input data to curtail the computational skew. NoLFA only consider the case where the computational time of an input partition depends upon the input data size rather than both. LEEN [13] attenuates the partitioning skew and minimalize the transfer of data using network through load balancing of the data distribution among the nodes in the cluster. It also improve the data locality of MapReduce tasks in the process. Unlike LEEN, NoLFA work in heterogeneous environment as well through our capacity awareness algorithm. Chen el al. [26] proposed Dynamic Smart Speculative technique to alleviate the problems with default speculation implementation like skew, indecorous phase percentage configuration and asynchronous twitch of certain tasks with the cost of degradation of performance for batch jobs. Whereas FP-Hadoop [27] introduces a new phase called intermediate reduce (IR) to parallelize the reduce task to efficiently tackle the reduce data skew problem. IR process the blocks of intermediate data in parallel. NoLFA has a different approach of decoupling the mappers and reducers tasks as introduced in our previous work [28].

VII. Conclusion

Hadoop affords simplified implementation of MapReduce framework, but its design stances challenges to attain best performance in application execution due to tightly coupled shuffle, obstinate scheduling and partitioning skew. In this paper, we developed an algorithm which takes node capabilities as heuristics to achieve better trade-off between locality and fairness in the Hadoop MapReduce system. It effectively improves the data locality and by comparing with the default Hadoop's partitioning algorithm and Leen partitioning algorithm, on the average our approach achieves better performance gain and outperform both of the previously mentioned partitioning schemes.

Acknowledgment

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (No. 2017R1A2B4010395).

References

[1] M. James, M. Chui, B. Brown, J. Bughin, R. Dobbs, C. Roxburgh, and A. H. Byers, "Big data: The next frontier for innovation, competition, and productivity," *McKinsey Glob. Inst.*, no. June, p. 156, 2011.

[2] P. Mell and T. Grance, "The NIST definition of cloud computing," *NIST Spec. Publ.*, vol. 145, p. 7, 2011.

[3] J. Dean and S. Ghemawat, "MapReduce," *Commun. ACM*, vol. 51, no. 1, p. 107, 2008.

[4] A. Matsunaga, M. Tsugawa, and J. Fortes, "CloudBLAST: Combining MapReduce and Virtualization on Distributed Resources for Bioinformatics Applications," *2008 IEEE Fourth Int. Conf. eScience*, pp. 222–229, 2008.

[5] J. Ekanayake, H. Li, B. Zhang, T. Gunarathne, S. Bae, J. Qiu, and G. Fox, "Twister : A Runtime for Iterative MapReduce," *HPDC '10 Proc. 19th ACM Int. Symp. High Perform. Distrib. Comput.*, pp. 810–818, 2010.

[6] G. Mackey, S. Sehrish, J. Bent, J. Lopez, S. Habib, and J. Wang, "Introducing map-reduce to high end computing," *2008 3rd Petascale Data Storage Work.*, pp. 1–6, 2008.

[7] C.-T. Chu, S. K. Kim, Y.-A. Lin, Y. Yu, G. Bradski, A. Y. Ng, and K. Olukotun, "Map-Reduce for Machine Learning on Multicore," *Adv. Neural Inf. Process. Syst. 19*, pp. 281–288, 2007.

[8] Apache!, "Apache™ Hadoop!" [Online]. Available: http://hadoop.apache.org/. [Accessed: 19-Dec-2016].

[9] Amazon!, "Amazon Elastic Compute Cloud (EC2)." [Online]. Available: http://aws.amazon.com/ec2/. [Accessed: 19-Dec-2016].

[10] Yahoo!, "Yahoo Developer Network." [Online]. Available: https://developer.yahoo.com/blogs/hadoop/yahoo-launches-world-largest-hadoop-production-application-398.html. [Accessed: 1-Jan-2017].

[11] R. Jain, D. Chiu, and W. Hawe, "A Quantitative Measure of Fairness and Discrimination for Resource Allocation in Shared Computer Systems," *DEC technical report TR301*, vol. cs.NI/9809, no. DEC-TR-301. pp. 1–38, 1998.

[12] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The Hadoop Distributed File System," *2010 IEEE 26th Symp. Mass Storage Syst. Technol.*, pp. 1–10, 2010.

[13] S. Ibrahim, H. Jin, L. Lu, S. Wu, B. He, and L. Qi, "LEEN: Locality/fairness-aware key partitioning for MapReduce in the cloud," *Proc. - 2nd IEEE Int. Conf. Cloud Comput. Technol. Sci. CloudCom 2010*, no. 2, pp. 17–24, 2010.

[14] T. Condie, N. Conway, P. Alvaro, J. M. Hellerstein, K. Elmeleegy, and R. Sears, "MapReduce online," *Nsdi'10*, pp. 21–21, 2010.

[15] M. Zaharia, A. Konwinski, A. Joseph, R. Katz, and I. Stoica, "Improving MapReduce Performance in Heterogeneous Environments." *Osdi*, pp. 29–42, 2008.

[16] "Amazon EC2 Instance Types." [Online]. Available: https://aws.amazon.com/ec2/instance-types/. [Accessed: 3-Jan-2017].

[17] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage, "Hey, You, Get Off of My Cloud: Exploring Information Leakage in Third-Party Compute Clouds," *Proc. 16th ACM Conf. Comput. Commun. Secur*, pp. 199–212, 2009.

[18] S. Seo, I. Jang, K. Woo, I. Kim, J.-S. Kim, and S. Maeng, "HPMR: Prefetching and pre-shuffling in shared MapReduce computation environment," *2009 IEEE Int. Conf. Clust. Comput. Work*, pp. 1–8, 2009.

[19] F. Ahmad, S. T. Chakradhar, A. Raghunathan, and T. N. Vijaykumar, "ShuffleWatcher: Shuffle-aware Scheduling in Multi-tenant MapReduce Clusters," *2014 USENIX Annu. Tech. Conf. (USENIX ATC 14)*, pp. 1–13, 2014.

[20] K. V Rashmi, M. Chowdhury, J. Kosaian, I. Stoica, K. Ramchandran, and I. Osdi, "EC-Cache : Load-Balanced , Low-Latency Cluster Caching with Online Erasure Coding This paper is included in the Proceedings of the R / W," *Osdi*, 2016.

[21] X. Tang, L. Wang, and Z. Geng, "A Reduce Task Scheduler for MapReduce with Minimum Transmission Cost Based on Sampling Evaluation," vol. 8, no. 1, pp. 1–10, 2015.
[22] P. Kondikoppa, C.-H. Chiu, C. Cui, L. Xue, and S.-J. Park, "Network-aware Scheduling of Mapreduce Framework on Distributed Clusters over High Speed Networks," pp. 39–44, 2012.
[23] Y. L. Su, P. C. Chen, J. B. Chang, and C. K. Shieh, "Variable-sized map and locality-aware reduce on public-resource grids," *Futur. Gener. Comput. Syst.*, vol. 27, no. 6, pp. 843–849, 2011.
[24] M. Hammoud, M. S. Rehman, and M. F. Sakr, "Center-of-gravity reduce task scheduling to lower MapReduce network traffic," *Proc. - 2012 IEEE 5th Int. Conf. Cloud Comput. CLOUD 2012*, pp. 49–58, 2012.
[25] Y. Kwon, M. Balazinska, B. Howe, and J. Rolia, "Skew-Resistant Parallel Processing of Feature-Extracting Scientific User-Defined Functions," 2010.
[26] Q. Chen, C. Liu, and Z. Xiao, "Improving MapReduce performance using smart speculative execution strategy," *IEEE Trans. Comput.*, vol. 63, no. 4, pp. 954–967, 2014.
[27] M. Liroz-Gistau, R. Akbarinia, D. Agrawal, and P. Valduriez, "FP-Hadoop: Efficient processing of skewed MapReduce jobs," *Inf. Syst.*, vol. 60, pp. 69–84, 2016.
[28] M. Hanif and C. Lee, "An Efficient Key Partitioning Scheme for Heterogeneous MapReduce Clusters," *Proc. 18th IEEE Int. Conf. Adv. Commun. Technol.*, 2016.

Muhammad Hanif was born in Pakistan. He received his B.S. degrees in computer and software engineering from University of Engineering and Technology (UET), Peshawar, Pakistan in 2012. He is currently pursuing his MS leading to PhD degree in Computer Software Engineering at Hanyang University, Seoul, South Korea. His current research interest includes Cloud & Distributed Computing, Big Data Analytic Engines, Stream Processing Frameworks, and Distributed Scheduling.

Choonhwa Lee was born in South Korea. He has been with the Division of Computer Science and Engineering at Hanyang University, Seoul, South Korea since 2004, and currently as Professor. He received his B.S. and M.S. degrees in computer engineering from Seoul National University (SNU), South Korea, in 1990 and 1992, respectively, and his Ph.D. degree in computer engineering from the University of Florida, Gainesville, in 2003. He worked as senior research engineer at LGIC Ltd from 1992 to 1998. He is a member of IEEE since 2004. His research interests include cloud computing, peer-to-peer and mobile networking and computing, and services computing technology.