# Revisit Data Partitioning in Data-intensive workflows

Radita Liem*[§], Shadi Ibrahim[†]
* RWTH Aachen University, Germany
[†] Inria, Univ. Rennes, CNRS, IRISA, France

As data volumes increase (e.g., in 2020 alone, 64.2 ZettaByte of data was generated and replicated [1]), data-intensive workflows are now executed across multiple platforms from the Edge to Cloud and HPC, which is known as computing continuum. The intention behind this execution model is to exploit data locality and reduce data movement by executing data close to their sources while taking into consideration the computation capacities of different platforms.

Data-intensive workflows usually consist of multiple computation stages where data flows among them. For example, MapReduce jobs consist of two consecutive stages (Map and Reduce) and the output of the map stage is transferred and used as an input for the Reduce stage [2], [3]. The completion time of a stage strongly depends on the finishing time of the last task in this stage. Accordingly, much efforts have focused on reducing the variation in task execution times in the same stage by launching speculative copies of slow tasks [2] or evenly partitioning data across tasks to mitigate data skew [4], [5], [6], [7].

With the proliferation of data-intensive workflows such as IoT applications, machine learning applications, and deep learning applications; data skew remains a bottleneck in data analytic frameworks. Many research efforts have been dedicated to mitigating data skew by evenly distributing data across tasks while considering data locality [4], [6] and relying on high-speed networks [7], or by partitioning the data based on the computation capacities of the nodes [8]. However, most of these efforts focus on only two-stage applications and don't consider network and I/O heterogeneity within or across platforms, making them impractical when running multi-stage data-intensive workflows on heterogeneous and shared environments.

As shown in Figure 1, PageRank application exhibits a severe skew in partitioned data (i.e., shuffled data) in stage 3 and stage 4 of the application (bottom chart). This in turn results in a noticeable variation in task execution times. The variation is represented by the gap between the average and maximum task execution times (top chart). We can also observe that the variation is more obvious when the degree of parallelism is set to 24, 32, and 40. Here, it is important to note that, the data assigned to each task (i.e., task input) includes the partitioned data and the data retrieved from the storage memory (cache).

[§]Work done while at Inria, Univ. Rennes, CNRS, IRISA, France.

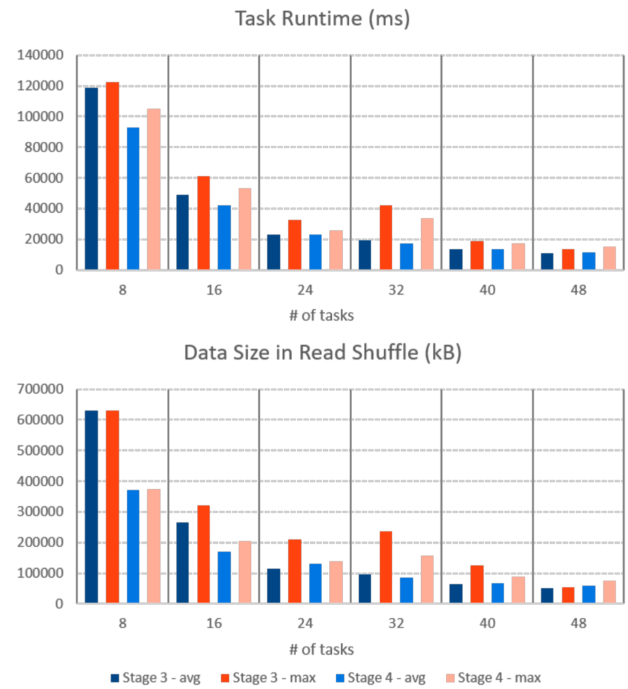


Fig. 1. The skew in partitioned data (i.e., data shuffled from previous stage and read by each task as a part of its input) and its impact on task execution times: We report the total size of shuffled data and task execution times in stages 3 and 4 when running PageRank application with 2.8 GB of data on Spark cluster of 4 nodes, deployed in Rennes site of Grid'5000. The number of stages is set to 6 stages.

In this work in progress, we will showcase a comprehensive analysis of the current state-of-the-art solutions for data skew mitigation in several environments. Our experiments and evaluation comprise several data-intensive workflows running on Spark using the Grid'5000 testbed [9]. The data-intensive workflows vary from a highly optimized WordCount application, an iterative application like PageRank, to an SQL-based decision support system benchmark, TPC-H with various sizes and configurations.

## ACKNOWLEDGMENTS

This work is supported by the ANR KerStream project (ANR-16-CE25-0014-01). Experiments presented in this paper were carried out using the Grid'5000 testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER, and several Universities as well as other organizations (see http://www.grid5000.fr/).

# REFERENCES


[1] David Reinsel, John Rydning, and John F Gantz. Worldwide global datasphere forecast, 2021–2025: The world keeps creating more data—now, what do we do with it all. *IDC Corporate USA*, 2021.

[2] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[3] Hai Jin, Shadi Ibrahim, Li Qi, Haijun Cao, Song Wu, and Xuanhua Shi. *The MapReduce Programming Model and Implementations*, chapter 14, pages 373–390. John Wiley Sons, Ltd, 2011.

[4] Shadi Ibrahim, Hai Jin, Lu Lu, Song Wu, Bingsheng He, and Li Qi. Leen: Locality/fairness-aware key partitioning for mapreduce in the cloud. In *2010 IEEE Second International Conference on Cloud Computing Technology and Science*, pages 17–24, 2010.

[5] Benjamin Gufler, Nikolaus Augsten, Angelika Reiser, and Alfons Kemper. Handling data skew in mapreduce. *Closer*, 11:574–583, 2011.

[6] Shadi Ibrahim, Hai Jin, Lu Lu, Bingsheng He, Gabriel Antoniu, and Song Wu. Handling partitioning skew in mapreduce using leen. *Peer-to-Peer Networking and Applications*, 6(4):409–424, 2013.

[7] Zeyu He, Zhifang Li, Xiaoshuang Peng, and Chuliang Weng. Ds 2: Handling data skew using data stealings over high-speed networks. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pages 1865–1870. IEEE, 2021.

[8] Qi Chen, Jinyu Yao, and Zhen Xiao. Libra: Lightweight data skew mitigation in mapreduce. *IEEE Transactions on parallel and distributed systems*, 26(9):2520–2533, 2014.

[9] Daniel Balouek, Alexandra Carpen Amarie, Ghislain Charrier, Frédéric Desprez, Emmanuel Jeannot, Emmanuel Jeanvoine, Adrien Lèbre, David Margery, Nicolas Niclausse, Lucas Nussbaum, Olivier Richard, Christian Pérez, Flavien Quesnel, Cyril Rohr, and Luc Sarzyniec. Adding virtualization capabilities to the Grid'5000 testbed. In *Cloud Computing and Services Science*, volume 367 of *Communications in Computer and Information Science*, pages 3–20. Springer International Publishing, 2013.