engineering proceedings

MDPI

Proceeding Paper

RGen: Data Generator for Benchmarking Big Data Workloads †

Rubén Pérez-Jove [1,*], Roberto R. Expósito [2] and Juan Touriño [2]

[1] Grupo RNASA-IMEDIR, CITIC, Universidade da Coruña, 15071 A Coruña, Spain
[2] Computer Architecture Group, CITIC, Universidade da Coruña, 15071 A Coruña, Spain; rreye@udc.es (R.R.E.); juan@udc.es (J.T.)
* Correspondence: ruben.perez.jove@udc.es
† Presented at the 4th XoveTIC Conference, A Coruña, Spain, 7–8 October 2021.

Abstract: This paper presents RGen, a parallel data generator for benchmarking Big Data workloads, which integrates existing features and new functionalities in a standalone tool. The main functionalities developed in this work were the generation of text and graphs that meet the characteristics defined by the 4 Vs of Big Data. On the one hand, the LDA model has been used for text generation, which extracts topics or themes covered in a series of documents. On the other hand, graph generation is based on the Kronecker model. The experimental evaluation carried out on a 16-node cluster has shown that RGen provides very good weak and strong scalability results. RGen is publicly available to download at https://github.com/rubenperez98/RGen, accessed on 30 September 2021.

Keywords: Data generator; MapReduce; HDFS; Apache Hadoop; Java; Big Data; Benchmarking

Citation: Pérez-Jove, R.; Expósito, R.R.; Touriño, J. RGen: Data Generator for Benchmarking Big Data Workloads. *Eng. Proc.* **2021**, *7*, 13. https://doi.org/10.3390/engproc2021007013

Academic Editors: Joaquim de Moura, Marco A. González, Javier Pereira and Manuel G. Penedo

Published: 2 October 2021

1. Introduction

One of the main problems that arise in those fields where huge amounts of data are managed is the necessity of having datasets that settle all the requirements in terms of volume, type and truthfulness. Overall, this kind of data can be extracted from preprocessed sources or generated synthetically. Specifically, the benchmark suites used to characterize the performance of Big Data frameworks and workloads generally rely on third-party tools for generating each type of input data that is needed, as there is no other option providing all of them. In this context, RGen has been developed as a parallel data generator for benchmarking Big Data workloads. RGen brings together a twofold task of integrating existing features and developing new functionalities in a standalone generator tool. The initial requirements for developing such tool are those specified by the data generation necessities of the Big Data Evaluator (BDEv) benchmark suite [1], which provides support for multiple representative Big Data workloads.

The main objective is the development of a parallel and scalable tool that gathers the necessary functionalities for BDEv without having to depend on third-party software to generate data for a great variety of workloads. Additionally, the performance evaluation and scalability of the data generator has been carried out both in a local environment and in a high-performance cluster. Different configurations have been evaluated considering both the number of nodes used and the amount of data to be generated in parallel.

2. Design and Implementation

RGen was developed under the MapReduce programming paradigm [2], more specifically on top of the Apache Hadoop framework [3], supporting the generation of data directly on the Hadoop Distributed File System (HDFS) [4].

The first step was the study of the state of the art regarding data generation topic. This research concluded with the choice of DataGen, the data generator tool integrated in the HiBench suite [5], as the base platform for our tool. The next step consisted in integrating some existing generation features not provided by DataGen from native classes of the Hadoop and Mahout frameworks.

The following phases were the development of two new generation methods, being the first one the text generation. To create new text that can preserve the characteristics of existing realistic data, the Latent Dirichlet Allocation (LDA) model [6] was selected, as it is one of the most widespread topic models. The implementation in RGen is able to generate text taking an LDA model as an input parameter, keeping the original characteristics of a pre-analyzed set of documents. Similarly to the previous method, the graph generation was tackled by using the Kronecker model [7], which allows keeping the most important characteristics of a set of nodes and vertexes and generating from such information new graphs that preserve its original constitution.

3. Experimental Evaluation

To analyze the scalability of the tool when generating data in a parallel way, multiple experiments were carried out, focused on evaluating the new features implemented in RGen: the text and graph generation based on the LDA and Kronecker models, respectively. Along with them, the experiments were also executed for random text generation and using PageRank for graph generation as baseline for comparison purposes.

Scalability is the capability of a parallel code to keep its performance when the computational resources and/or the problem size are increased. There are two ways of measuring this metric: (1) weak scalability, where the number of CPU cores is increased while keeping constant the workload per core (i.e., both the number of cores and the problem size are increased); and strong scalability, where the resources are increased while the total workload remains the same (i.e., the workload per core is reduced). Weak scalability tests the capability of addressing larger problems in the same time by increasing the resources in a proportional way. On the other hand, strong scalability focuses on minimizing the runtime needed for solving the same problem by adding more resources.

Table 1 shows the configuration of the experiments conducted to analyze weak and strong scalability. The experiments were executed on the Pluton cluster of the Computer Architecture Group, where each node provides 16 physical CPU cores, 64 GB of memory and 1 TB local disk intended for HDFS storage. Additionally, all the nodes are interconnected via InfiniBand FDR (56 Gbps). As can be seen in Table 1, the experiments were conducted varying the number of nodes from two up to 16.

Table 1. Configuration of the experiments carried out on a high-performance cluster.

#Nodes	Text Data Volume		Graph Data Volume	
	Weak Scalability	**Strong Scalability**	**Weak Scalability**	**Strong Scalability**
2	40 GB	320 GB	67 M nodes	536 M nodes
4	80 GB	320 GB	134 M nodes	536 M nodes
8	160 GB	320 GB	268 M nodes	536 M nodes
16	320 GB	320 GB	536 M nodes	536 M nodes

4. Results and Conclusions

Figures 1 and 2 show the results for text and graph generation, respectively. Each plot presents both weak and strong scalability for the new generation methods (single lines) and for those used as baseline (marked lines). The runtimes for weak scalability are shown in green lines against the left axis, while the red lines present the runtimes for strong scalability against the right axis. The first conclusion that can be drawn is that the new generation methods take more time to execute for the same experiment than those used as baseline. This is an expected behavior as the computational complexity of these methods for generating data based on the LDA and Kronecker models is significantly higher than generating text randomly or using PageRank for graph generation.

When analyzing these results further, it can be concluded that RGen provides good scalability overall. In the case of text generation (see Figure 1), almost constant runtimes are obtained for weak scalability, which means that RGen provides similar runtimes when

the number of resources and the workload are increased proportionally. Regarding strong scalability, it can be seen a significant reduction in runtime when generating text using the LDA model. This means that the same amount of text (320 GB) is generated much faster when increasing the computational resources. The results show almost linear strong scalability for LDA-based text generation, powered by combining MapReduce with HDFS (only *Map* tasks are executed in this case).

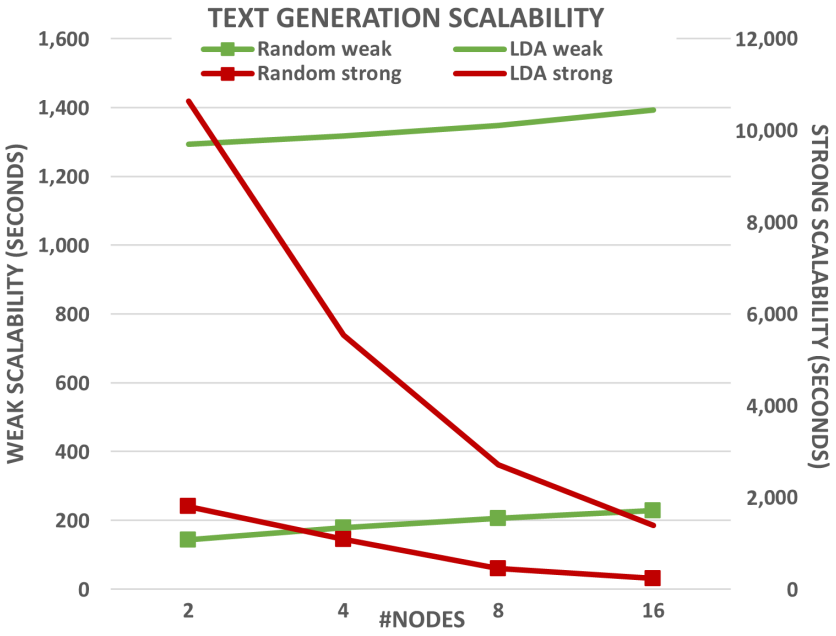

Figure 1. Scalability results for text.

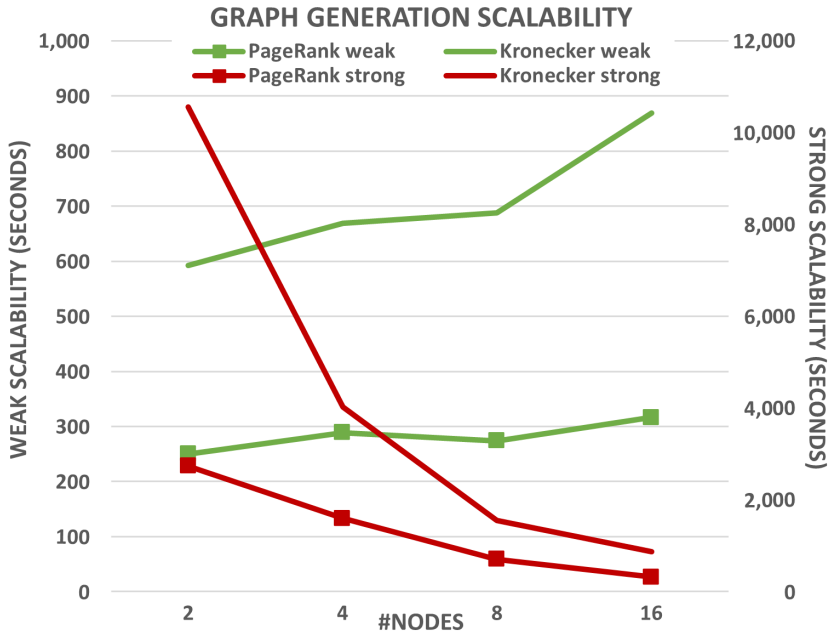

Figure 2. Scalability results for graphs.

The results for graph generation show a similar trend (see Figure 2). On the one hand, weak scalability presents a more irregular pattern for both data generation methods when compared to text generation. However, these results can be explained taking into account that the Kronecker method executes two MapReduce jobs instead of only one, and they also require to execute *Reduce* tasks. Both facts can hinder scalability as the cluster network

performance now plays a key role, especially when using 16 nodes. On the other hand, the strong scalability provided by the Kronecker model is even more noticeable than in the PageRank implementation.

Author Contributions: Conceptualization, R.P.-J., R.R.E. and J.T.; methodology, R.P.-J., R.R.E. and J.T.; implementation, R.P.-J.; validation, R.P.-J.; writing—original draft preparation, R.P.-J.; writing—review and editing, R.R.E. and J.T. All authors have read and agreed to the published version of the manuscript.

Funding: CITIC, as Research Center accredited by Galician University System, is funded by "Consellería de Cultura, Educación e Universidade from Xunta de Galicia", supported in an 80% through ERDF, ERDF Operational Programme Galicia 2014–2020, and the remaining 20% by "Secretaría Xeral de Universidades (Grant ED431G 2019/01). This project was also supported by the "Consellería de Cultura, Educación e Ordenación Universitaria" via the Consolidation and Structuring of Competitive Research Units—Competitive Reference Groups (ED431C 2018/49 and 2021/30).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Veiga, J.; Enes, J.; Expósito, R.R.; Touriño, J. BDEv 3.0: Energy Efficiency and Microarchitectural Characterization of Big Data Processing Frameworks. *Future Gener. Comput. Syst.* **2018**, *86*, 565–581. [CrossRef]
2. Dean, J.; Ghemawat, S. MapReduce: Simplified Data Processing on Large Clusters. *Commun. ACM* **2008**, *51*, 107–113. [CrossRef]
3. The Apache Software Foundation. Apache Hadoop. Available online: https://hadoop.apache.org (accessed on 30 July 2021).
4. Shvachko, K.; Kuang, H.; Radia, S.; Chansler, R. The Hadoop Distributed File System. In Proceedings of the IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST'2010), Incline Village, NV, USA, 3–7 May 2010; pp. 1–10.
5. Huang, S.; Huang, J.; Dai, J.; Xie, T.; Huang, B. The HiBench Benchmark Suite: Characterization of the MapReduce-based Data Analysis. In Proceedings of the IEEE 26th International Conference on Data Engineering Workshops (ICDEW'2010), Long Beach, CA, USA, 1–6 March 2010; pp. 41–51.
6. Blei, D.M.; Ng, A.Y.; Jordan, M.I. Latent Dirichlet Allocation. *J. Mach. Learn. Res.* **2003**, *3*, 993–1022.
7. Leskovec, J.; Chakrabarti, D.; Kleinberg, J.; Faloutsos, C.; Ghahramani, Z. Kronecker Graphs: An Approach to Modeling Networks. *J. Mach. Learn. Res.* **2010**, *11*, 985–1042.