Moreover, both tools ran very quickly and are suitable for online predictions. The average execution times of dagSim were 3.09 seconds for scenario 1 and only 0.76 seconds for scenario 2, with very low variability across multiple runs (coefficient of variation[3] (CV) of 0.06 in both cases). Vice versa, JMT took on average 156 and 83 seconds, respectively.

For scenarios 3 and 4, despite the higher variability (CVs of 0.9 and 0.8, respectively), the average execution times were still short, 1.2 and 2.4 seconds, respectively. Note that in this latter scenario the higher variability was due to the different size of the underlying dataset (which has an impact on the number of tasks within stages and the number of simulated events).

The execution times of the analytical Task Precedence model was very short, varying from only 4.18 milliseconds (for scenario 2) to up to 40 milliseconds (for scenario 4). They were also mostly stable (i.e., low CVs) across all scenarios. The average execution times are 5.35 ms, 4.59 ms, 9.42 ms and 28.38 ms for scenarios 1 to 4, respectively, whereas the corresponding CVs are 0.12, 0.05, 0.32, and 0.29. Thus, comparing both tools, dagSim's execution times exceed those of our analytical model by some orders of magnitude: their ratio varies from around 10 to over 680. However, the Task Precedence model is limited to assess average execution time, whereas dagSim can provide also percentiles of application performance, thus enabling much finer-grained analyses.

5 CONCLUSIONS

In this paper, we analized an analytical models and proposed an ad-hoc simulator for the performance prediction of Spark applications running on cloud clusters.

Multiple cloud configurations and workloads (including SQL and iterative machine learning benchmarks) have been considered. From the results we achieved, Lundstrom and the dagSim simulator perform very well for predicting the average system response time and are effective in capturing the dynamic resource assignment implemented in Spark, achieving 11.07% and 6.06% average percentage error across all the experiments, respectively.

In our future work, we plan to extend our models to cope with scenarios where multiple applications run concurrently competing to access the resources in the same clusters. Finally, we will embed the models into a run-time optimization tool for dynamically managing cloud resources with the aim of providing application execution within an a priori fixed deadline while minimizing cloud operational costs.

ACKNOWLEDGEMENT

The authors' work has been partially funded by the EUBra-BIGSEA project by the European Commission under the Cooperation Programme (MCTI/RNP 3rd Coordinated Call), Horizon 2020 grant agreement 690116. This research was also be partially funded by CNPq and FAPEMIG, Brazil.

REFERENCES

[1] [n. d.]. Apache Spark Survey 2016 Results Now Available. ([n. d.]). https://databricks.com/blog/2016/09/27/spark-survey-2016-released.html
[2] [n. d.]. The Digital Universe in 2020. ([n. d.]). http://idcdocserv.com/1414
[3] D. Ardagna, S. Bernardi, E. Gianniti, S. Karimian Aliabadi, D. Perez-Palacin, and J. I. Requeno. 2016. Modeling Performance of Hadoop Applications: A Journey from Queueing Networks to Stochastic Well Formed Nets. In *ICA3PP*. 599–613. https://doi.org/10.1007/978-3-319-49583-5_47
[4] E. Barbierato. 2016. *dagSim Documentation.* Technical Report. Politecnico di Milano. https://github.com/eubr-bigsea/dagSim/blob/master/simlib/Documentation/scheduler/manual/1.63/manual.html
[5] M. Bertoli, G. Casale, and G. Serazzi. 2009. JMT: Performance Engineering Tools for System Modeling. *ACM SIGMETRICS Performance Evaluation Review* 36, 4 (2009), 10–15.
[6] K. Chen, J. Powers, S.Guo, and F. Tian. 2014. CRESP: Towards Optimal Resource Provisioning for MapReduce Computing in Public Clouds. *IEEE TPDS* 25, 6 (2014), 1403–1412. https://doi.org/10.1109/TPDS.2013.297
[7] G. Chiola. 1985. A Software Package for the Analysis of Generalized Stochastic Petri Net Models. In *International Workshop on Timed Petri Nets*. 136–143.
[8] G. Ciardo, R. L. Jones, III, A. S. Miner, and R. I. Siminiceanu. 2006. Logic and Stochastic Modeling with SMART. *Perform. Eval.* 63 (June 2006), 578–608. Issue 6. https://doi.org/10.1016/j.peva.2005.06.001
[9] H. Derrick. 2015. Survey Shows Huge Popularity Spike for Apache Spark. (2015). http://fortune.com/2015/09/25/apache-spark-survey
[10] W.J. Fokkinkk. 2000. *Introduction to Process Algebra.* Springer.
[11] J. Hillston. 1996. *A Compositional Approach to Performance Modelling.* Cambridge University Press, New York, NY, USA.
[12] M. Leeser J. Bhimani, N. Mi. [n. d.]. Scalable Performance Prediction Techniques for Big Data Processing in Distributed Multi-Core Systems. ([n. d.]). http://hdl.handle.net/2047/D20215315
[13] H. V. Jagadish, Johannes Gehrke, Alexandros Labrinidis, Yannis Papakonstantinou, Jignesh M. Patel, Raghu Ramakrishnan, and Cyrus Shahabi. 2014. Big Data and Its Technical Challenges. *Commun. ACM* 57, 7 (July 2014), 86–94.
[14] J. Laskowski. 2016. Mastering Apache Spark. (2016). https://www.gitbook.com/book/jaceklaskowski/mastering-apache-spark
[15] E. D. Lazowska, J. Zahorjan, G. S. Graham, and K. C. Sevcik. 1984. *Quantitative System Performance.* Prentice-Hall. http://homes.cs.washington.edu/~lazowska/qsp/
[16] V. W. Mak and S. F. Lundstrom. 1990. Predicting Performance of Parallel Computations. *IEEE Trans. Parallel Distrib. Syst.* 1, 3 (July 1990), 257–270. https://doi.org/10.1109/71.80155
[17] Microsoft. [n. d.]. Sizes for Windows Virtual Machines in Azure. https://docs.microsoft.com/en-us/azure/virtual-machines/windows/sizes. ([n. d.]). [Online; accessed 15-January-2017].
[18] Microsoft. 2016. What is PaaS? (2016). https://azure.microsoft.com/en-us/overview/what-is-paas/
[19] R. D. Nelson and A. N. Tantawi. 1988. Approximate Analysis of Fork/Join Synchronization in Parallel Queues. *IEEE Trans. Computers* 37, 6 (1988), 739–743.
[20] A. D. Popescu. 2015. *Runtime Prediction for Scale-Out Data Analytics.* Ph.D. Dissertation. IC, Lausanne. https://doi.org/10.5075/epfl-thesis-6629
[21] W. Reisig, G. Rozenberg, and P. S. Thiagarajan. 2013. *In Memoriam: Carl Adam Petri.* Springer Berlin Heidelberg, Berlin, Heidelberg, 1–5. https://doi.org/10.1007/978-3-642-38143-0_1
[22] G. Song, Z. Meng, F. Huet, F. Magoules, L. Yu, and et al. 2013. A Hadoop MapReduce Performance Prediction Method. In *HPCC*. 820–825.
[23] D. Towsley, J. C.S. Lui, and R. R. Muntz. 1998. Computing Performance Bounds of Fork-Join Parallel Programs under a Multiprocessing Environment. *IEEE Transactions on Parallel & Distributed Systems* 9, 3 (1998), 295–311. https://doi.org/10.1109/71.674321
[24] S. K. Tripathi and D. Liang. 2000. On Performance Prediction of Parallel Computations with Precedent Constraints. *IEEE Transactions on Parallel & Distributed Systems* 11 (2000), 491–508. https://doi.org/10.1109/71.852402
[25] K. S. Trivedi. 2002. SHARPE 2002: Symbolic Hierarchical Automated Reliability and Performance Evaluator. In *DSN*. IEEE Computer Society, Washington, DC, USA, 544.
[26] E. Varki and L. W. Dowdy. 1996. Analysis of Balanced Fork-join Queueing Networks. *SIGMETRICS Perform. Eval. Rev.* 24, 1 (May 1996), 232–241. https://doi.org/10.1145/233008.233048
[27] E. Vianna, G. Comarela, T. Pontes, J. Almeida, V. Almeida, K. Wilkinson, H. Kuno, and U. Dayal. 2013. Analytical Performance Models for MapReduce Workloads. *International Journal of Parallel Programming* 41, 4 (2013), 495–525. https://doi.org/10.1007/s10766-012-0227-4
[28] Gu. Wang, A. R. Butt, P. Pandey, and K. Gupta. 2009. A Simulation Approach to Evaluating Design Decisions in MapReduce Setups. In *MASCOTS*. IEEE Computer Society, 1–11.
[29] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica. 2010. Spark: Cluster Computing with Working Sets. In *HotCloud*. USENIX Association, Berkeley, CA, USA, 10–10. http://dl.acm.org/citation.cfm?id=1863103.1863113

[3] Ratio of standard deviation to mean value.