Challenges in High Performance Big Data Frameworks

Alessandro V. Papadopoulos
Mälardalen University
Västerås, Sweden
alessandro.papadopoulos@mdh.se

Martina Maggio
Lund University
Lund, Sweden
martina.maggio@control.lth.se

***Abstract*—Nowadays, we live in a society with billions of devices that are interconnected and interact together to improve the quality of our lives. The management and processing of information and knowledge have by now become our main resources, and the fundamental factors of economic and social development, and it is achieved through Big Data Frameworks (BDFs). The amount of such data is becoming larger every day, and this calls for scalable and reliable BDFs, that can process such data also with real-time requirements. For example, the data collected by an autonomous car should be processed, combined, and interpreted as fast as possible in order to guarantee a safe interaction with the surrounding environment, and of the passengers.**

This paper analyses the main limitations of current BDFs while providing some key challenges for increasing their flexibility. In particular, we focus on performance aspects, envisioning adaptation as a viable way to automate and improve performance in Big Data Applications.

Index Terms*—*big data; self-adaptive systems; autonomic computing

I. Introduction

The term big data is used to refer to non-traditional strategies and technologies used to gather, organize, and process large datasets [34]. Big data refers to the problem of working with data exceeding the computing capacity (computation or storage) of a single computer. This problem is not new, but its pervasiveness and scale are unprecedented. In the past few years, the development of big data frameworks has revolutionised the industrial practice in many different domains [38]: From robotics [41] to smart cities [18], [39], to automotive applications [27], [42], and to handling patient data for large-scale healthcare databases [17]—in all these domains processing a large amount of data is an enabler of new research and new solutions. Big data systems are uniquely suited for surfacing difficult-to-detect patterns and providing insight into behaviors that are impossible to find through conventional means. Think for example about analysing in parallel hundreds of thousands tumor images [29] and determining, based on the results of the analysis, the best therapy to be applied in a specific patient case.

The goal of most big data systems is to surface insights and connections (i) from large amount of data (*Volume*) (ii) of heterogeneous data (*Variety*), (iii) almost in real-time (*Velocity*), that would not be possible using conventional methods. These three dimensions are typically called the big data "3 Vs" [26], and are the essential ingredients of a big data application. In addition to the classical "3 Vs", emerging domains, such as autonomous vehicles or smart grids, call for additional dimensions [21], like, for example:

- the data should be meaningful for the problem being analyzed, even if data can be biased, noisy, or incomplete (*Veracity*),
- the data should be accurate and correct for the intended use, especially if decision making is based on the collected and processed data (*Validity*),
- how long the collected data should be stored depends on the specific application at hand (*Volatility*)
- the data can be *geo-distributed*, making difficult to synchronize the different sources of data [2] (*Venue*).

For example, in a smart traffic light system, a smart traffic light needs to both locally get information from the vehicles approaching the intersection, analyze the pedestrians crossing the intersection, but also to coordinate with the other traffic light in the surrounding areas to avoid or at least reduce congestions in the city. Such a complex scenario, involves both local and global calculations, and motivated the introduction of new computation paradigms, such as *fog computing* [2].

In this paper, we analyze the main features of the main BDFs currently available, highlighting their technical limitations in terms of flexibility, and adaptability with respect to changes in the environment where corresponding big data applications can be deployed. The main goal of the paper is to identify possible research directions in creating high-performance and flexible BDFs.

II. Related work

The amount of data produced by mobile phones, wearables, sensors and computers brings novel challenges in data storage and analysis. A number of different big data technologies have been developed to cope with such an increasing demand, such as Hadoop, Hbase or CouchDB. Occasionally, big data

This work was partially supported by the Swedish Foundation for Strategic Research under the project "Future factories in the cloud (FiC)" with grant number GMT14-0032, by the Wallenberg Autonomous Systems Program (WASP), by the Swedish Research Council (VR) for the projects "Cloud Control", "Feedback Computing" and "Power and temperature control for large-scale computing infrastructures"

978-1-5386-7879-4/18/$31.00 ©2018 IEEE
DOI 10.1109/HPCS.2018.00039

technologies are used to implement data-mining techniques, but more often the they are used for data processing in support of the data-mining techniques and other data-science activities, and even for real-time processing, i.e., data streaming applications.

The term *big data* was first used in 1944, to refer to the imminent explosion of stored information, and then remained unused until the nineties. In the beginning of the third millennium, Google publishes a paper describing the Google File System [11]. While the paper does not explicitly mention the term "big data", the content of the work precisely fits the paradigm of modern big data frameworks. The Google File System introduces a new technology to handle the complexity of processing amount of data that previous technologies could not process. Nowadays, this technology has gained traction and has been far more important, as the amount of collected data in many different domains steadily increases [34].

As a result, a number of BDFs were created, giving rise to an articulated *scenario* [25], [30]. Currently, there are three main types of big data frameworks:

1) **Batch-only** frameworks, such as Apache Hadoop.
2) **Stream-only** frameworks, such as Apache Storm, Apache Kafka [20], and Apache Samza.
3) **Hybrid** frameworks, such as Apache Flink and Apache Spark [43].

Independently of the typology of the BDF, there are several commonalities of the frameworks, in terms of operators, e.g., stateless and stateful operators, as well as models, e.g., the MapReduce programming model, and the abstraction of BDA description based on Directed Acyclic Graphs (DAGs).

MapReduce is a popular programming model and execution environment for BDAs [8], and there have been some pioneering works on performance modeling [10], [40] and control [1], [3], [4], [22].

Spark is an alternative to MapReduce that generalises the types of data flows supported, overcoming the limitation of acyclic data flow models, and extending it with a data-sharing abstraction called "resilient distributed datasets," or RDDs [43]. It was originally developed to support interactive data exploration and analytics, and for iterative jobs. Spark tries to keep data in memory at each cluster node, and avoids reloading data from disk as much as possible, trying to minimize memory latency.

Another very famous framework is Kafka [20], that targets real-time processing of data with high throughput and low latencies. Kafka was originally developed by LinkedIn but now it is an open-source project of Apache. Kafka is a streaming platform with three main capabilities: (i) publish and subscribe to stream of records (called topics), (ii) store streams of records, and (iii) process streams of records as they occur. The main structure of Kafka is a distributed producer-consumer architecture, where producers store messages at a set of (stateless) servers (called brokers), from where consumers can pull messages. Kafka targets real-time streaming applications, and data pipelines connecting systems or applications. Kafka can scale horizontally by adding or removing brokers to a cluster.

From a conceptual modelling viewpoint, however, Kafka, Spark, MapReduce and similar technologies can be described in analogous ways without loss of generality. In fact, virtually any BDF is nowadays based on `MapReduce`, or built on top of it, trying to optimize how resources are accessed or utilized. This allows one to address the need to exploit parallelism, which is inherent to big data applications. Parallelism appears both at the application level, which is apparent, and at the data level, i.e., when parallelism does not appear among parts of the application, rather it is intrinsic to the individual operation that is being executed on the data.

III. Limitations and challenges

There are several limitations of current BDFs. We here identified four main limitations, and corresponding challenges for the development of the next generation BDFs. Such limitations result in researchers that need to develop their own big data frameworks [6], [13], [14], and also it may hinder the possibility to adopt big data frameworks in safety-critical applications, such as in human-robot interaction, autonomous vehicles, or health-care applications.

A. Programming language

Most of the BDFs are written in Scala, Java, or Python [33]. The reason why Java was adopted was the rich set of libraries and used extensively in industrial applications. However, they are not optimized for high performance computing, and real-time operation, and the utilization of compiled languages, such as C/C++, can significantly improve their run-time execution, providing a more predictable behavior, thanks also to programming models like MPI and OpenMP [28], [31].

B. Adaptive resource management

The programmer that wants to take advantage of the performance-potential offered by BDFs must know a lot of the internals of the framework that he/she selects, and should then optimize many parameters of the execution, including resource-allocation, data-processing, and data-storage parameters. This calls for self-adaptiveness of the software system, to adapt to the dynamic and uncertain environment in which the big data application is deployed [5], [35].

Although all modern BDFs do take care of parallelism and its exploitation, they typically employ threshold based-mechanisms for scaling horizontally or vertically, and this significantly limits their performance. Such an approach has proven successful in practice, but the underlying standpoint makes it difficult to guarantee *a priori* any characteristic of the BDA execution based on conveniently qualified expected environmental conditions. More advanced mechanisms for scaling the applications can be used, exploiting machine learning [24], or control-theoretic approaches [7], [16], [22].

In this respect, there is a clear need of performance models of components and operators of big data frameworks, for designing and analysing their behavior, and the fundamental

limitations from a theoretical perspective. One example is provided in [12], where the stream join operator is modeled and validated with a real implementation from a performance perspective, providing insight on how the number of threads affect latency and throughput of tuples as a function of the incoming workload, even in overloaded scenarios. In general, however, building models to represent complex static and dynamic component compositions in both the system and the underlying infrastructure, exploiting multiple architecture styles, and accurately representing combinations of heterogeneous and uncertain workloads challenges the state of the art in performance modeling [19].

Finally, big data computation is now best effort: a computation is started with no guarantees on its completion time, its accuracy, or its resource consumption. It is clear that this limits the applicability of BDFs, and safety-critical applications cannot be addressed easily with the current technology.

C. Self-adaptation of big data applications

As highlighted in [36], current BDFs do not provide an extensive support for the development self-adaptive applications. The self-adaptation is typically limited to the internal mechanisms of the BDF, and to optimize the resource utilization. BDFs should, on the other hand, include suitable interfaces for the runtime adaptation of the big data applications themselves [9].

D. Safety assurances

Criticality of systems implies stringent requirements on safety assurance and security. When a system might harm humans or the environment, decision-makers require assurance that it manages risk acceptably. Most safety standards are primarily focused on closed or controlled models of development and on pre-release assurance of safety.

Currently, BDFs are inflexible and cannot really handle seamlessly the needs emerging in different application domains, especially regarding safety properties. The need for new dimensions with respect to the classical "3 Vs", is just one way to see this problem.

The safety-critical nature of many big data applications, e.g., human-robot interaction, autonomous vehicles, and the smart traffic light system, require the system designer to provide safety guarantees on the actual execution of the overall system. This is not only true when humans are in the loop, but also to avoid accidents that would cause economic losses, e.g., when two autonomous robots crash in a factory.

BDFs are not able to provide such guarantees, especially due to the large scale of the application that make performance prediction at design time (without performance models) practically infeasible. Due to the data growth, shared cloud-based infrastructures, and application evolution, even assurances at design time would become invalid almost immediately. Some researchers provided some preliminary results on runtime certification [23], [32], [37], but they do not extend easily to big data applications.

E. Challenges

The main challenges for the development of high-performance BDFs that should be overcome can be summarized as follows:

1) BDFs are mainly developed with programming languages not explicitly targeting high performance computing, e.g., Java, Python or Scala. The main challenge is to develop a **high performance framework**, and to improve the real-time support.
2) **Performance models**, that would allow both fast prototyping of big data applications, as well as improve the efficiency of how the resources are utilized, are missing. Performance models are also important for providing fundamental limits on the obtainable performance, and can constitute the basis for a deeper understanding of big data technologies.
3) There is currently little support for runtime adaptation and resource management, resulting in difficulty in designing autonomic computing solutions [15]. The main challenge is to develop suitable interfaces and mechanisms for the efficient **runtime adaptation**, in order to design big data applications that require very little knowledge of the environment where they will be developed, providing *plug-and-play* solutions.
4) Most of the existing BDFs do not target **safety-critical applications** explicitly. The main challenge is to develop suitable verification-oriented models of big data components, that can provide runtime guarantees of their execution, even for safety-critical applications.

IV. Conclusion

In this paper we briefly analyzed the main BDFs present on the market, highlighting their main limitations. It is clear that big data is an enabling technology for several application domains, but big data technologies have not been extensively used in the context of high performance computing, and for safety-critical applications. We believe that this could be addressed by the introduction of self-adaptive mechanisms that would increase the capabilities of current BDFs, as well as their flexibility. In particular, we identified four main challenges for big data technologies, discussing the main limitations that should be addressed to solve such problems.

There are several other challenges related to BDFs, including data quality, data security, data management, privacy [21], but we limited the scope of this work to only performance and assurance issues.

References

[1] M. Berekmeri, D. Serrano, S. Bouchenak, N. Marchand, and B. Robu. A control approach for performance of big data systems. *IFAC Proceedings Volumes*, 47(3):152–157, 2014.

[2] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli. Fog computing and its role in the internet of things. In *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, MCC '12, pages 13–16, New York, NY, USA, 2012. ACM.

[3] M. Cardosa, P. Narang, A. Chandra, H. Pucha, and A. Singh. STEAMEngine: Driving mapreduce provisioning in the cloud. In *2011 18th Int. Conf. on High Performance Computing*, pages 1–10, Dec 2011.

[4] S. Cerf, M. Berekmeri, B. Robu, N. Marchand, and S. Bouchenak. Adaptive optimal control of mapreduce performance, availability and costs. In *Feedback Computing*, 2016.

[5] B. H. C. Cheng, R. Lemos, H. Giese, P. Inverardi, J. Magee, J. Andersson, B. Becker, N. Bencomo, Y. Brun, B. Cukic, G. Marzo Serugendo, S. Dustdar, A. Finkelstein, C. Gacek, K. Geihs, V. Grassi, G. Karsai, H. M. Kienle, J. Kramer, M. Litoiu, S. Malek, R. Mirandola, H. A. Müller, S. Park, M. Shaw, M. Tichy, M. Tivoli, D. Weyns, and J. Whittle. *Software Engineering for Self-Adaptive Systems*, chapter Software Engineering for Self-Adaptive Systems: A Research Roadmap, pages 1–26. 2009.

[6] K. Crawford and J. Schultz. Big data and due process: Toward a framework to redress predictive privacy harms. *BCL Rev.*, 55, 2014.

[7] T. De Matteis and G. Mencagli. Proactive elasticity and energy awareness in data stream processing. *J. Syst. Softw.*, 127(C):302–319, May 2017.

[8] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113, Jan. 2008.

[9] S. Deng, B. Wang, S. Huang, C. Yue, J. Zhou, and G. Wang. Self-adaptive framework for efficient stream data classification on storm. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pages 1–14, 2017.

[10] A. Ganapathi, Y. Chen, A. Fox, R. Katz, and D. Patterson. Statistics-driven workload modeling for the cloud. In *Data Engineering Workshops (ICDEW), 2010 IEEE 26th Int. Conf. on*, pages 87–92, March 2010.

[11] S. Ghemawat, H. Gobioff, and S.-T. Leung. The google file system. In *Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles*, SOSP '03, pages 29–43, New York, NY, USA, 2003. ACM.

[12] V. Gulisano, A. V. Papadopoulos, Y. Nikolakopoulos, M. Papatriantafilou, and P. Tsigas. Performance modeling of stream joins. In *11th ACM Int. Conf. on Distributed and Event-based Systems*, DEBS '17, pages 191–202, 2017.

[13] I. A. T. Hashem, I. Yaqoob, N. B. Anuar, S. Mokhtar, A. Gani, and S. U. Khan. The rise of "big data" on cloud computing: Review and open research issues. *Information Systems*, 47:98–115, 2015.

[14] H. Herodotou, H. Lim, G. Luo, N. Borisov, L. Dong, F. B. Cetin, and S. Babu. Starfish: a self-tuning system for big data analytics. In *Conference on Innovative Data Systems Research*, volume 11, pages 261–272, 2011.

[15] J. O. Kephart and D. M. Chess. The vision of autonomic computing. *Computer*, 36(1):41–50, 2003.

[16] A. Khoshkbarforoushha, A. Khosravian, and R. Ranjan. Elasticity management of streaming data analytics flows on clouds. *Journal of Computer and System Sciences*, 89:24 – 40, 2017.

[17] M. J. Khoury and J. P. Ioannidis. Big data meets public health. *Science*, 346(6213):1054–1055, 2014.

[18] R. Kitchin. The real-time city? big data and smart urbanism. *GeoJournal*, 79(1):1–14, 2014.

[19] J. Klein and I. Gorton. Runtime performance challenges in big data systems. In *Proceedings of the 2015 Workshop on Challenges in Performance Methods for Software Development*, WOSP '15, pages 17–22, New York, NY, USA, 2015. ACM.

[20] J. Kreps, N. Narkhede, J. Rao, et al. Kafka: A distributed messaging system for log processing. In *Proceedings of the 6th International Workshop on Networking Meets Databases (NetDB)*, pages 1–7, 2011.

[21] I. Lee. Big data: Dimensions, evolution, impacts, and challenges. *Business Horizons*, 60(3):293 – 303, 2017.

[22] A. Leva and A. V. Papadopoulos. Modelling and control of big data frameworks. In *20th IFAC World Congress*, pages 6110–6115, 2017.

[23] Y. J. Lim, G. Hong, D. Shin, E. Jee, and D.-H. Bae. A runtime verification framework for dynamically adaptive multi-agent systems. In *2016 International Conference on Big Data and Smart Computing (BigComp)*, pages 509–512, Jan 2016.

[24] J. Lin and D. Ryaboy. Scaling big data mining infrastructure: The twitter experience. *SIGKDD Explor. Newsl.*, 14(2):6–19, Apr. 2013.

[25] X. Liu, N. Iftikhar, and X. Xie. Survey of real-time processing systems for big data. In *Proceedings of the 18th International Database Engineering & Applications Symposium*, IDEAS '14, pages 356–361, New York, NY, USA, 2014. ACM.

[26] R. Lu, H. Zhu, X. Liu, J. K. Liu, and J. Shao. Toward efficient and privacy-preserving computing in big data era. *IEEE Network*, 28(4):46–50, July 2014.

[27] Y. Lv, Y. Duan, W. Kang, Z. Li, and F.-Y. Wang. Traffic flow prediction with big data: a deep learning approach. *IEEE Transactions on Intelligent Transportation Systems*, 16(2):865–873, 2015.

[28] Y. Ma, H. Wu, L. Wang, B. Huang, R. Ranjan, A. Zomaya, and W. Jie. Remote sensing big data computing: Challenges and opportunities. *Future Generation Computer Systems*, 51:47 – 60, 2015. Special Section: A Note on New Trends in Data-Aware Scheduling and Resource Provisioning in Modern HPC Systems.

[29] D. D. Manojbhai, K. K. Pradipkumar, and R. R. Amenakshi. Big image analysis for identifying tumor pattern similarities. In *2016 Int. Conf. on Advanced Communication Control and Computing Technologies (ICACCCT)*, pages 39–43, May 2016.

[30] N. Marz and J. Warren. *Big Data: Principles and Best Practices of Scalable Realtime Data Systems*. Manning Publications Co., Greenwich, CT, USA, 1st edition, 2015.

[31] J. L. Reyes-Ortiz, L. Oneto, and D. Anguita. Big data analytics in the cloud: Spark on hadoop vs mpi/openmp on beowulf. *Procedia Computer Science*, 53:121 – 130, 2015. INNS Conference on Big Data 2015 Program San Francisco, CA, USA 8-10 August 2015.

[32] J. Rushby. Runtime certification. In M. Leucker, editor, *Runtime Verification*, pages 21–35, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[33] S. Sagiroglu and D. Sinanc. Big data: A review. In *Collaboration Technologies and Systems (CTS), 2013 Int. Conf. on*, pages 42–47, 2013.

[34] S. Sakr and A. Zomaya, editors. *Encyclopedia of Big Data Technologies*. Springer International Publishing AG, 2018.

[35] M. Salehie and L. Tahvildari. Self-adaptive software: Landscape and research challenges. *ACM Trans. Auton. Adapt. Syst.*, 4(2):14:1–14:42, May 2009.

[36] S. Schmid, I. Gerostathopoulos, C. Prehofer, and T. Bures. Self-adaptation based on big data analytics: A model problem and tool. In *2017 IEEE/ACM 12th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, pages 102–108, May 2017.

[37] D. Schneider and M. Trapp. Conditional safety certification of open adaptive systems. *ACM Trans. Auton. Adapt. Syst.*, 8(2):8:1–8:20, July 2013.

[38] D. Singh and C. K. Reddy. A survey on platforms for big data analytics. *Journal of Big Data*, 2(1):8, Oct 2014.

[39] A. M. Townsend. *Smart cities: Big data, civic hackers, and the quest for a new utopia*. WW Norton & Company, 2013.

[40] A. Verma, L. Cherkasova, and R. H. Campbell. *Resource Provisioning Framework for MapReduce Jobs with Performance Goals*, pages 165–186. Springer Berlin Heidelberg, 2011.

[41] A. White, J. Modayil, and R. S. Sutton. Surprise and curiosity for big data robotics. In *AAAI-14 Workshop on Sequential Decision-Making with Big Data, Quebec City, Quebec, Canada*, 2014.

[42] J. Yu, F. Jiang, and T. Zhu. Rtic-c: A big data system for massive traffic information mining. In *2013 International Conference on Cloud Computing and Big Data*, pages 395–402, Dec 2013.

[43] M. Zaharia, R. S. Xin, P. Wendell, T. Das, M. Armbrust, A. Dave, X. Meng, J. Rosen, S. Venkataraman, M. J. Franklin, A. Ghodsi, J. Gonzalez, S. Shenker, and I. Stoica. Apache spark: A unified engine for big data processing. *Commun. ACM*, 59(11):56–65, 2016.