

Data Multiverse: The Uncertainty Challenge of Future Big Data Analytics

Radu Tudoran^(✉), Bogdan Nicolae, and Götz Brasche

Huawei German Research Center, Riesstraße 25, 80992 München, Germany
{radu.tudoran,bogdan.nicolae,goetz.brasche}@huawei.com

Abstract. With the explosion of data sizes, extracting valuable insight out of big data becomes increasingly difficult. New challenges begin to emerge that complement traditional, long-standing challenges related to building scalable infrastructure and runtime systems that can deliver the desired level of performance and resource efficiency. This vision paper focuses on one such challenge, which we refer to as the analytics uncertainty: with so much data available from so many sources, it is difficult to anticipate what the data can be useful for, if at all. As a consequence, it is difficult to anticipate what data processing algorithms and methods are the most appropriate to extract value and insight. In this context, we contribute with a study on current big data analytics state-of-art, the use cases where the analytics uncertainty is emerging as a problem and future research directions to address them.

Keywords: Big data analytics · Large scale data processing · Data access model · Data uncertainty · Approximate computing

1 Introduction

Data is the new natural resource. Its ingestion and processing leads to valuable insight that is transformative in all aspects of our world [12]. Science employs data-driven approaches to create complex models that explain nature in great detail, otherwise impossible to obtain using traditional analytical approaches by themselves. In industry, data science is an essential value generator: it leverages a variety of modern data sources (mobile sensors, social media, etc.) to understand customer behaviors and financial trends, which ultimately facilitates better business decisions.

Helped by the rise of cloud computing, an entire data market has emerged that enables users to share and consume massive amounts of data from a variety of distributed data sources. At the core of this data market is the so called *big data analytics* ecosystem: an ensemble of techniques and middleware specifically designed to interpret unstructured data on-the-fly (e.g. Spark [16], Flink [1], Storm [13], etc.). As more value is extracted out of the data market using big data analytics, the interest for collecting more data emerges, which ultimately unleashes a chain reaction. An evidence for this is already visible in overall

annual world network traffic: the end of 2016 will mark the beginning of the Zettabyte era (i.e. 1 ZB), with growth predictions showing 2.3 ZB by 2020 [2].

However, while there is an obvious advantage of access to more data in that more insight can be potentially extracted, this also introduces an unprecedented challenge: it is becoming increasingly difficult for the user to know in advance how useful the data is (either whole or parts of it) or what algorithm would work best. We call this the *analytics uncertainty*. Because of it, users are often trapped in a sequential trial-and-error process: apply data transformation, interpret result, adjust algorithm, repeat. Given the data market explosion, this is simply too slow to produce the desired results in a timely fashion. Therefore, a new way to reason about big data at scale is needed.

In this paper we study the current state of art from the analytics uncertainty perspective. Our contribution can be summarized as follows: (1) we discuss the state-of-art big data analytics ecosystem and its expressivity in terms of data processing mechanisms offered to the users; (2) we discuss a series of use case patterns that exhibit an inherent analytics uncertainty; (3) based on these patterns, we identify and discuss the limitations of the current approaches.

2 Background and Related Work

We summarize in this Section major classes of big data analytics approaches.

2.1 MapReduce-Like

Data-oriented batch programming models that separate the computation from its parallelization gained rapid popularity beginning with the MapReduce [7] paradigm. While convenient for the user due to automated parallelization by design, at runtime-level this introduces a significant level of complexity. One major contribution in this area is the *data-locality centered design*: the storage layer is co-located with the compute elements and exposes the data locations such that the computation can be scheduled close to the data. Using this approach, scalability is possible even on commodity hardware. Combined with the ease of use, this lead to a wide adoption of MapReduce in production environments.

2.2 In-Memory Generic Processing

Over time, MapReduce presented several limitations, both in terms of expressivity (formulating a solution by means of *map* and *reduce* is often difficult) and performance (high overhead for I/O despite co-location of compute and storage). To this end, a new generation of *in-memory big data analytics* is increasingly gaining popularity over MapReduce such as Spark, Flink. By making heavy use of in-memory data caching such engines minimizes the interactions with the storage layer, which further reduces I/O bottlenecks due to slow local disks, extra copies and serialization issues. To improve expressivity these engines facilitates the development of multi-step data pipelines using a directed acyclic graph

(DAG) as a runtime construct based on the high-level control flow of the application defined via the exposed API.

One illustrative example is Spark [16], which relies on two main parallel programming abstractions: (1) *resilient distributed datasets* (RDDs), a partitioned data structure hosting the data itself; and (2) parallel operations on the RDDs. RDD holds provenance information (referred to as *lineage*) and can be rebuilt in case of failures by partial recomputation from ancestor RDDs. RDD operations can be either *narrow* (i.e. each output partition can be computed from its corresponding input partition directly) or *wide* (i.e., each output partition is computed by combining pieces from many other input partitions). Wide transformations are particularly challenging because they involve complex all-to-all data exchanges, which may introduce overhead with respect to performance, scalability and resource utilization [10]. Furthermore, optimized broadcast of data is another important issue [11].

2.3 Stream Processing

Live data sources (e.g., web services, social and news feeds, sensors, etc.) are increasingly playing a critical role in big data analytics. By introducing an online dimension to data processing, they improve the reactivity and “freshness” of the results, which ultimately can potentially lead to better insights. As a consequence, *big data stream processing* saw a rapid rise recently. Storm [13] was one of the first low-level engines that introduced basic semantics in terms of bolts and sprouts. As the need for highly reactive processing grew, other more sophisticated approaches appeared such as S4 [9], MillWheel [3], Apache Flink, Apache APEX or Apache Samza. These approaches address issues such as fault tolerance, low latency and consistency guarantees (at least one, at most one, exactly one). A complementary effort towards low latency also gained significant traction: the idea of optimizing batch processing to handle frequent mini-batches. This approach was popularized by Spark via DStreams [17], then followed by Apache Storm Trident and data management tools like JetStream [14].

A significant effort to formalize stream computing in a way that captures the liveness of input data and the results by design is the *window* concept. Windows are stream operators that offer a mechanism to specify when old data is discarded (eviction policy), when to trigger a computation over the accumulated data, what function defines the computation and how it manages intermediate states that lead to the result (which may involve data duplication). Many complex applications such as live machine learning are based on the concept of window [6, 8, 15]. Active research is carried out both at the level of performance optimizations [6] and expressivity (e.g., extended triggering mechanisms and watermarking [4]).

3 Big Data Use Cases with Emerging Analytics Uncertainty

Big data analytics is data-centric: it needs to constantly adapt and evolve to match the nature of the data available to it. This contrasts the traditional approaches that require the data to adapt to the application. In this section, we develop this perspective using three illustrative use cases where the analytics uncertainty is the main driver for the need to adapt.

A/B Testing: A big data solution running in production often has to solve a difficult trade-off: on one hand it has to be stable enough to deliver consistent results, but on the other hand it needs to be constantly refined to maximize the delivered value (i.e., can further value be extracted?). Despite significant advances in techniques to reduce the cycle of adopting innovation in production through agile DevOps-based software engineering, such approaches are insufficient for big data analytics. As a consequence, *A/B testing* (also called *split-run testing*) is often employed: the main stable solution A is constantly producing results, while an alternative, trial-and-error B solution running in parallel is used to challenge the results. If the B solution obtains a better result, it already replaces the initial result. Eventually, if the B solution consistently delivers better results, then it replaces the A solution and the process restarts.

Machine Learning: A key class of applications that are based on big data analytics is machine learning. Similar with A/B testing, machine learning typically trains and deploys a stable model (i.e. the *champion* model), while at the same time it needs to continuously improve it. To accelerate the improvement of the champion, a typical approach is to construct alternative models (trained with different data, different or variations of the main algorithm, or a combination thereof) that challenge the champion model for the top spot. The challenge in this context is to run both the champion and the challenger model in a seamless fashion, so that they complement each other and generate mutual feedback used for online improvement.

Deep Exploration: Big data is a powerful tool to discover trends and extract insight out of data without necessarily knowing in advance what to look for. The thin line between this and machine learning is given by the goal of the analysis, which for exploration algorithms moves from building a model that solves a particular well defined problem to synthesizing the data, i.e., transforming the data from its raw form into an knowledge augmented form. In this process, various hypothesis and classes of exploration operations are carried with respect to the data [5]. This raises the need to enable efficient multi-path exploration, mainly against large data sets or within performance constrains.

4 Limitations of State-of-Art and Opportunities

Current state of art approaches described in Sect. 2 lack the support to deal with the analytics uncertainty as illustrated in Sect. 3. This happens both at

conceptual level (i.e., users lack mechanisms and APIs from the runtime and need to implement their own application-level approach) as well as at runtime level (i.e., the runtime is unaware of the intent of the application to deal with the analytics uncertainty and therefore misses optimization opportunities). In this context, we identify two important dimensions:

The *breadth uncertainty*: When users need to process the same data set with multiple algorithms simultaneously, they need to manage complex workflows and intermediate states at application level: when to branch into an alternate direction, what intermediate state to compare with, where to roll back to try a different direction, etc. Lacking support from the runtime, users are forced to interact with various layers explicitly (e.g., use the storage layer to persist intermediate states), which can lead to scalability and performance issues. Furthermore, unaware of the relationship between the alternate directions, the runtime will often perform suboptimally (e.g., duplicate data unnecessarily). Thus, an interesting research direction in this context is how to expose a data processing model that natively enables the exploration of alternate directions, which ultimately enables the construction of optimized runtimes.

The *temporal uncertainty*: In addition to processing the same data set with multiple algorithms simultaneously, an important aspect of the analytics uncertainty is also the temporal dimension. Specifically, as new data is accumulating, the question of “freshness” arises: does it make sense to process all available data or focus only on the most recent data to gain the most relevant insight into the future? Where is the right trade-off, i.e. how far back into the history does it make sense to search? Answering such a question naturally leads to the need of combining batch processing with stream processing. However, the current ecosystem of big data analytics frameworks is fragmented: users have to use different runtimes for different jobs. This becomes an issue both because the application has to explicitly manage different jobs on different runtimes (which leads to complexity) and because the runtimes do not talk to each other (e.g. share common data and states) and as such perform suboptimally. Thus, an interesting research direction is to extend data processing models with native support to focus on the temporal aspect.

5 Conclusions

In this paper we studied the problem of *analytics uncertainty*: how to enable extracting valuable insight from an ever-increasing data market whose usefulness is not known in advance. We identified several use cases where this problem arises and discussed the limitations of existing state of art big data analytics approaches in handling such use cases. Our view argues in favor of the need to build a more expressive data management model that facilitates the construction of corresponding optimized runtimes. In this context, we highlighted the importance of addressing two uncertainty dimensions: *breadth* and *temporal*. Based on the findings, we plan to explore these two directions in future work.

References

1. Flink. <https://flink.apache.org/>
2. The Zettabyte Era: Trends and Analysis. Cisco Systems, White Paper 1465272001812119 (2016)
3. Akidau, T., Balikov, A., Bekiroglu, K., Chernyak, S., Haberman, J., Lax, R., McVeety, S., Mills, D., Nordstrom, P., Whittle, S.: Millwheel: Fault-tolerant stream processing at internet scale. In: *Very Large Data Bases*, pp. 734–746 (2013)
4. Akidau, T., Bradshaw, R., Chambers, C., Chernyak, S., Fernandez-Moctezuma, R.J., Lax, R., McVeety, S., Mills, D., Perry, F., Schmidt, E., Whittle, S.: The dataflow model: A practical approach to balancing correctness, latency, and cost in massive-scale, unbounded, out-of-order data processing. *Proc. VLDB Endowment* **8**, 1792–1803 (2015)
5. Cao, L., Wei, M., Yang, D., Rundensteiner, E.A.: Online outlier exploration over large datasets. In: *21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2015, Sydney, Australia*, pp. 89–98 (2015)
6. Carbone, P., Traub, J., Katsifodimos, A., Haridi, S., Markl, V.: Cutty: Aggregate sharing for user-defined windows. In: *25th ACM International Conference on Information and Knowledge Management, CIKM 2016*, pp. 1201–1210 (2016)
7. Dean, J., Ghemawat, S.: Mapreduce: Simplified data processing on large clusters. In: *6th Conference on Symposium on Operating Systems Design and Implementation, OSDI 2004*, pp. 10:1–10:13. USENIX Association, San Francisco (2004)
8. Hammad, M.A., Aref, W.G., Elmagarmid, A.K.: Query processing of multi-way stream window joins. *VLDB J.* **17**(3), 469–488 (2008)
9. Neumeyer, L., Robbins, B., Kesari, A., Nair, A.: S4: Distributed stream computing platform. In: *10th IEEE International Conference on Data Mining Workshops, ICDMW 2010, Los Alamitos, USA*, pp. 170–177 (2010)
10. Nicolae, B., Costa, C., Misale, C., Katrinis, K., Park, Y.: Leveraging adaptive I/O to optimize collective data shuffling patterns for big data analytics. *IEEE Trans. Parallel Distrib. Syst.* (2017)
11. Nicolae, B., Kochut, A., Karve, A.: Towards scalable on-demand collective data access in IaaS clouds: An adaptive collaborative content exchange proposal. *J. Parallel Distrib. Comput.* **87**, 67–79 (2016)
12. Hey, T., Tansley, S., Tolle, K.M.: *The Fourth Paradigm: Data-Intensive Scientific Discovery*. Microsoft Research, Redmond (2009)
13. Toshniwal, A., et al.: Storm@twitter. In: *2014 ACM SIGMOD International Conference on Management of Data, SIGMOD 2014, Snowbird, USA*, pp. 147–156 (2014)
14. Tudoran, R., Costan, A., Nano, O., Santos, I., Soncu, H., Antoniu, G.: Jetstream: Enabling high throughput live event streaming on multi-site clouds. *Future Gener. Comput. Syst.* **54**, 274–291 (2016)
15. Yang, D., Rundensteiner, E.A., Ward, M.O.: Shared execution strategy for neighbor-based pattern mining requests over streaming windows. *ACM Trans. Database Syst.* **37**(1), 5:1–5:44 (2012)
16. Zaharia, M., Chowdhury, M., Das, T., Dave, A., Ma, J., McCauly, M., Franklin, M.J., Shenker, S., Stoica, I.: Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In: *The 9th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2012, San Jose, USA* (2012)
17. Zaharia, M., Das, T., Li, H., Shenker, S., Stoica, I.: Discretized streams: An efficient and fault-tolerant model for stream processing on large clusters. In: *4th USENIX Conference on Hot Topics in Cloud Computing, HotCloud 212* (2012)

Semantic Keyword-Based Search on Structured Data
Sources

COST Action IC1302 Second International KEYSTONE
Conference, IKC 2016, Cluj-Napoca, Romania,

September 8–9, 2016, Revised Selected Papers

Calì, A.; Gorgan, D.; Ugarte, M. (Eds.)

2017, X, 197 p. 41 illus., Softcover

ISBN: 978-3-319-53639-2