Big Data Research ••• (••••) •••–•••

Contents lists available at ScienceDirect

Big Data Research

www.elsevier.com/locate/bdr

Big Data Systems Meet Machine Learning Challenges: Towards Big Data Science as a Service

Radwa Elshawi [a,b], Sherif Sakr [b,c,*], Domenico Talia [d], Paolo Trunfio [d]

[a] Princess Nora bint Abdul Rahman University, Saudi Arabia
[b] University of Tartu, Estonia
[c] King Saud bin Abdulaziz University for Health Sciences, Saudi Arabia
[d] University of Calabria, Italy

ARTICLE INFO

Article history:
Received 28 December 2017
Received in revised form 28 March 2018
Accepted 27 April 2018
Available online xxxx

ABSTRACT

Recently, we have been witnessing huge advancements in the scale of data we routinely generate and collect in pretty much everything we do, as well as our ability to exploit modern technologies to process, analyze and understand this data. The intersection of these trends is what is, nowadays, called *Big Data Science*. Big Data Science requires scalable architectures for storing and processing data. Cloud computing represents a practical and cost-effective solution for supporting Big Data storage, processing and for sophisticated analytics applications. We analyze in details the building blocks of the software stack for supporting Big Data Science as a commodity service for data scientists. In addition, we analyze and classify the state-of-the-art of big data analytics frameworks, available today mostly on Clouds, based on their supported service models. Furthermore, we provide various insights about the latest ongoing developments and open challenges in this domain.

1. Big Data science

The continuous growth and integration of data storage, computation, digital devices and networking empowered a rich environment for the explosive growth of Big Data as well as the tools through which data is produced, shared, cured and analyzed [43].

In addition to the **4Vs** (Volume, Velocity, Variety and Veracity), it is vital to consider an additional feature of *Big Data* that is ***Value***. Value is obtained by analyzing Big Data and extracting from them hidden patterns, trends and knowledge models by using smart data analysis algorithms and techniques. Data science methods must be able to analyze Big Data and extract features we don't know. Those learned features improve the value of data that will make it possible to better understand phenomena and behaviors, optimizing processes, and improving machine, business and scientific discovery. Therefore, we cannot look at Big Data Science without considering data analysis and machine learning as key steps for including value as a Big Data Science strategy.

In practice, big data analytics tools enable data scientists to discover correlations and patterns via analyzing massive amounts of data from various sources that are of different types. Recently, Big Data science [3] has emerged as a modern and important data analysis discipline. It is considered as an amalgamation of classical disciplines such as statistics, artificial intelligence, mathematics and computer science with its sub-disciplines including database systems, machine learning and distributed systems. It combines existing approaches with the aim of turning abundantly available data into value for individuals, organizations, and society. The ultimate goal of data science techniques is to convert data into meaningful information. Both in business and in science, data science methods have shown to facilitate more robust decision making capabilities. In the last few years, we have witnessed a huge emergence of Big Data Science in various real-world applications such as business optimization, financial trading, healthcare data analytics and social network analysis, just to name but a few [43]. In particular, we can think of the relationship between Big Data and data science as being like the relationship between crude oil and an oil refinery.

A McKinsey global report described Big Data as "*Data whose scale, distribution, diversity, and/or timeliness require the use of new technical architectures and analytics to enable insights that unlock the new sources of business value*" [36]. The Big Data phenomenon has spurred the scientific communities to reconsider their research methods and processes [55] by reorienting them towards insights obtained by learning from data. In 2007, Jim Gray, the Turing Award winner, separated data-intensive science from computa-

* Corresponding author.
E-mail addresses: rmelshawi@pnu.edu.sa (R. Elshawi), sakrs@ksau-hs.edu.sa (S. Sakr), talia@dimes.unical.it (D. Talia), paolo.trunfio@unical.it (P. Trunfio).

https://doi.org/10.1016/j.bdr.2018.04.004
2214-5796/

tional science. He called for a paradigm shift in the computing architecture and large scale data processing platforms known as the Fourth Paradigm [23]. Experiments, study of theorems and laws, and simulation were in a chronological way the previous three paradigms. Gray argued that this new paradigm does not only represent a shift in the methods of scientific research, but also a shift in the way that people think. He declared that the only way to deal with the challenges of this new paradigm is to build a new generation of computing systems to manage, analyze and visualize the data deluge. Spurred by continuous and dramatic advancements in processing power, memory, storage, and an unprecedented wealth of data, Big Data processing platforms have been developed to tackle the increasingly complex data science jobs. Lead by the `Hadoop` framework [52] and its ecosystem, Big Data processing systems are showing remarkable success in several business and research domains [43]. In particular, for about a decade, the Hadoop platform represented the defacto standard of the Big Data analytics world. However, we have recently been witnessing a new wave of Big Data 2.0 processing platforms [43] that are dedicated to specific verticals such as structured SQL data processing (e.g., `Hive` [49], `Impala` [26], `Presto`[1]), large scale graph processing (e.g., `Giraph` [44], `Graphlab` [33], `GraphX` [19]), large scale stream processing data (e.g., `Storm`,[2] `Heron` [29], `Flink` [14], `Samza` [40], `Kafka` [28]) and machine learning and data analysis (`Pig` [18], `Mahout` [41], `Spark MLib` [38], `Azure ML` [48]).

The techniques and technologies of Big Data Science have been able to penetrate all facets of the business and research domains. From the modern business enterprise to the lifestyle choices of today's digital citizen, the insights of Big Data analytics are driving changes and improvements in every arena [37]. Several big data surveys have been presented in the literature [66–74]. In this paper, we are taking a different perspective from the previous surveys. In particular, we are comprehensively covering the systems and frameworks perspective for the different layers of the data analytics techniques (e.g., machine learning, deep learning). We summarize the main contributions of this paper as follows:

- We analyze the main features and the building blocks of the software stack for supporting Big Data science on Clouds as a commodity service for data scientists.
- We identify a set of main requirements for effectively achieving the vision of providing Big Data analytics as a service.
- We analyze and classify the state-of-the-art of Big Data analytics frameworks based on their supported service models.
- We provide various insights about the latest ongoing developments and open challenges in this domain.

The reminder of this paper is organized as follows. Section 2 provides an overview of the role of cloud computing and its service models as a main component for supporting the implementation of software stacks of Big Data science as a service. Section 3 discusses the main requirements of deploying Big Data analysis jobs on cloud environments. Section 4 analyzes the state-of-the-art of Big Data analytics frameworks based on their supported service model. Section 5 discusses and provides insights on some of the open challenges towards achieving the goals and the vision of providing Big Data Science as a Service.

2. Cloud computing for Big Data

Cloud computing represents a paradigm shift in the process of provisioning computing infrastructures. This paradigm shifts the location of infrastructure to more centralized and larger scale datacenters in order to reduce the costs associated with the management of software and hardware resources [54]. Clouds provide users with the perception of accessing (virtually) unlimited computing resources where scalability is secured by elastically adding computing resources as the requirement of the workload increases. They revolutionized the information technology industry by providing the flexibility in the way that computing resources are consumed by supporting the philosophy of the pay-as-you-go pricing model for the resources and services used by the consumers. Therefore, cloud computing represented a crucial step towards realizing the long-held dream of envisioning computing as a utility where the economy of scale principles help to effectively drive down the cost of computing infrastructure. In practice, big technology companies (e.g., Amazon, Google, Microsoft) have dedicated a lot of resources and investments in establishing their own data centers and cloud-based services across the world to provide assurances on reliability by providing redundancy for their supporting infrastructure, platforms and applications to their cloud consumers. These cloud services can be provided according to the *Infrastructure as a Service* (IaaS) model (e.g. `Amazon Elastic Compute Cloud (EC2)`[3] and `Google Compute Engine`[4]), the *Platform as a Service*(PaaS) model (e.g. `Microsoft Azure`[5] and `Google App Engine`[6]), or the *Software as a Service (SaaS)* model (e.g., `Salesforce.com`[7] and `Zoho`.[8] Big Data analysis applications can be implemented within each of the three cloud service models: with the IaaS model, a set of virtualized resources can be provided to developers as a computing infrastructure to run their Big Data analysis applications or to implement their Big Data analysis systems from scratch; with the PaaS model, a supporting platform can be provided to developers that have to build their own Big Data applications or extend existing ones; finally, with the SaaS model, a well-defined Big Data analysis process or a ready-to-use data analysis tool can be provided as an Internet service to end-users, who may directly use it through a Web browser.

As a matter of fact, Big Data and cloud computing technologies have been combined in a way that has made it easier and more flexible than ever for everyone to step into the world of Big Data processing [61]. In particular, this technology combination has enabled even small companies and individual data scientists to collect and analyze terabytes of data. For instance, `Amazon EC2` is provided as a commodity service which can be purchased and exploited merely by using a credit card to pay for the service. In addition, several cloud-based data storage solution (e.g., `Amazon Simple Storage Service (S3)`,[9] `Amazon RDS`,[10] `Amazon DynamoDB`,[11] `Google Cloud Data Store`,[12] `Google Cloud SQL`[13]), for different data forms, have been provided enabling hosting massive amounts of data at very low cost and on demand. Furthermore, various Big Data processing frameworks have been made available via cloud-based solutions [42]. For example, Amazon has also released `Amazon Elastic MapReduce (EMR)`[14] as a cloud service that allows its users to easily and cost-effectively analyze massive sizes of data without the need

[1] https://prestodb.io/.
[2] http://storm.apache.org/.
[3] https://aws.amazon.com/ec2/.
[4] https://cloud.google.com/compute/.
[5] https://azure.microsoft.com/.
[6] https://cloud.google.com/appengine/.
[7] https://www.salesforce.com/.
[8] https://www.zoho.com/.
[9] https://aws.amazon.com/s3/.
[10] https://aws.amazon.com/rds/.
[11] https://aws.amazon.com/dynamodb/.
[12] https://cloud.google.com/datastore/.
[13] https://cloud.google.com/sql/.
[14] http://aws.amazon.com/elasticmapreduce/.

Please cite this article in press as: R. Elshawi et al., Big Data Systems Meet Machine Learning Challenges: Towards Big Data Science as a Service, Big Data Res. (2018), https://doi.org/10.1016/j.bdr.2018.04.004

to get involved in the challenging and time-consuming aspects of running a Big Data analytics job such as setup, configuration, management and performance tuning of complex computing clusters. Other cloud-based Big Data processing services include `Databricks Spark`,[15] `Amazon Redshift`,[16] `Google BigQuery`[17] and `Azure HDInsight`.[18]

In practice, these cloud-based services allow third-parties to execute Big Data analysis tasks over a huge amount of data with minimum effort and cost by abstracting the complexity entailed in developing and maintaining complex computing clusters. Therefore, they paved the way and provided the fundamental elements of the software stack of providing *Big Data Science as a service* in a way that follows the cloud-based trend of providing everything-as-a-service (XaaS) [5].

3. Main requirements of Big Data analysis on Clouds

In general, it has been well-recognized that Clouds can effectively support Big Data science applications since they provide scalable storage and computing services, as well as software platforms for developing and running large-scale data analysis on top of such services [56]. However, in order to achieve this goal, we envision a list of main requirements that should be met by cloud-based data analysis systems to be used in Data Science solutions. In particular, from the infrastructure point of view, the following requirements should be met:

- *Standardized access*: The infrastructure should expose its services using standard technologies (e.g., Web services, microservices) making them usable as building blocks for higher level services and applications.
- *Heterogeneous/distributed data support*: The infrastructure should be able to cope with very large and high dimensional datasets that are stored in different formats in a single data center or geographically distributed across many sites.
- *Scalability*: The infrastructure should be able to handle a growing workload (deriving from larger data to process or heavier algorithms to execute) by dynamically allocating the needed resources (processors, storage, network). Moreover, as soon as the workload decreases, the infrastructure should release the unrequired resources.
- *Efficiency*: The infrastructure should minimize resource consumption for a given task to execute. In the case of parallel tasks, efficient allocation of processing nodes should be assured.
- *Security*: The infrastructure should provide effective security mechanisms to ensure data protection, identity management, and privacy.

Among the requirements at the architectural level, the main two are:

- *Service-orientation*: The infrastructure should be designed as a set of network-enabled software components (services) implementing the different operations of the system to facilitate their effective reuse, composition, and interoperability.
- *Openness and extensibility*: A Big Data science architecture should be open to the integration of new tools and services. Moreover, existing services should be open for extension, but closed for modification, according to the open-closed principle.

In practice, the main resources in Big Data analysis applications are data sources, analysis tools and results. For managing these resources, some main requirements are needed as follows:

- *Data management*: Data sources and data output (results) can be represented in different formats, such as files, relational databases, NoSQL tables, or semi-structured docs. A system should provide mechanisms to store and access such data sources independently from their specific format. In addition, metadata formalisms should be defined and used to describe the relevant information associated with data sources (e.g., location, type, format), for enabling their access, use and manipulation.
- *Tool management*: Data analysis tools include algorithms and services for data selection, preprocessing, transformation, data mining, and output evaluation. Systems should provide mechanisms to access and use such tools independently from their specific implementation. Metadata can be used to describe the most important features of tools (e.g., functions, results, use).

Finally, a Big Data analysis system must provide efficient mechanisms for designing analysis tasks in Big Data science applications (design management) and controlling their execution (execution management).

- *Design management*: Big Data analysis applications involve complex data mining patterns generally expressed as trees, graphs or workflows. In all those cases, complex data analysis computations are designed as structured patterns that link together data sources, data transformation/analysis algorithms, and output management tools. A general system should provide environments to effectively design all the above-mentioned classes of data analysis tasks.
- *Execution management*: Systems have to provide a parallel/distributed execution environment that supports the efficient execution of a large number of data processing tasks. The execution environment should cope with a variety of applications. In particular, the execution environment should provide functionalities that are related to the different phases of data science application execution, such as: parallel data access, compute resource allocation; running application based on user specifications, results presentation. Additionally, systems must allow users to monitor application execution.

4. Big Data science frameworks

The Big Data phenomenon has created ever-increasing pressure for scalable data processing solutions. Several data management and processing systems have been recently implemented. In particular, the NoSQL database approach became popular in the last years as an alternative or as a complement to relational databases, to ensure horizontal scalability of simple read/write database operations distributed over many servers [12,60]. NoSQL systems like `HBase`,[19] `Cassandra`,[20] `MongoDB`,[21] `Couchbase`[22] and `DynamoDB`[23] are today efficiently used to support the implementation of Big Data analysis frameworks and applications. Indeed, NoSQL databases provide efficient mechanisms and techniques to store and access scalar values, binary objects, and more complex data.

[15] https://databricks.com/product/databricks.
[16] https://aws.amazon.com/redshift/.
[17] https://cloud.google.com/bigquery/.
[18] https://azure.microsoft.com/en-us/services/hdinsight/.
[19] https://hbase.apache.org/.
[20] http://cassandra.apache.org/.
[21] https://www.mongodb.com/.
[22] https://www.couchbase.com/.
[23] https://aws.amazon.com/dynamodb/.

Please cite this article in press as: R. Elshawi et al., Big Data Systems Meet Machine Learning Challenges: Towards Big Data Science as a Service, Big Data Res. (2018), https://doi.org/10.1016/j.bdr.2018.04.004

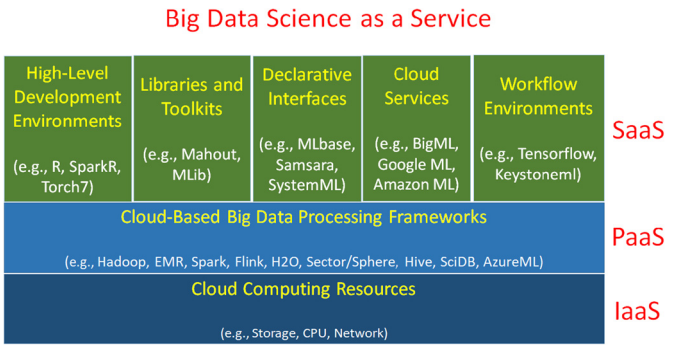

Fig. 1. Big Data Science as a Service software stack.

The increasing data analysis requirements of almost all application domains have created a crucial need for designing and building a new generation of Big Data science tools that can efficiently and effectively analyze massive amounts of data in order to elicit worthy information, detect interesting insights and discover meaningful patterns and knowledge [47]. According to the three main cloud service models (Section 2), Big Data Science tools supporting analysis and learning from data can be classified either as `Paas` or as `SaaS`. This classification is illustrated in Fig. 1 that summarizes the Big Data Science as a Service software stack. Typically, the `PaaS` layer covers cloud-based data analysis frameworks that are used to implement data science software as a service while the `SaaS` layer covers cloud-based machine learning software or libraries that have been developed for extracting knowledge models, learning from data and making predictions. In the following sub-sections, we classify the data analysis frameworks according to these two main services models: PaaS and SaaS.

4.1. PaaS data analysis frameworks

PaaS frameworks allow users to focus on creating and running applications rather than building and maintaining the underlying infrastructure and services. Data analysis PaaS frameworks offer compute and storage services as well as data analysis and machine learning services that help developers in creating applications more quickly and efficiently. We will discuss some of the most used frameworks in the following.

`MapReduce` [59] is a programming model developed by Google for large-scale data processing to cope efficiently with the challenge of processing enormous amounts of data generated by Internet-based applications. Since its introduction, MapReduce has proven to be applicable to a wide range of domains, including machine learning and data mining and social data analysis. Today, MapReduce is widely recognized as one of the most important programming models for Cloud computing environments and it is supported by Google and other leading Cloud providers such as Amazon, with its `Elastic MapReduce` service, and Microsoft, with its `HDInsight` or on top of private Cloud infrastructures such as `OpenStack`,[24] with its `Sahara`[25] service. The `Hadoop`[26] project is well-recognized as the most popular open source MapReduce implementation for developing parallel applications that analyze big amounts of data. It can be adopted for developing distributed and parallel applications using various programming languages (e.g., `Java`, `Ruby`, `Python`, `C++`). Hadoop relieves developers from having to deal with classical distributed computing issues, such as load balancing, fault tolerance, data locality, and network bandwidth saving. As a result, Hadoop became a reference for several other frameworks, such as: `Giraph` for graph analysis; `Storm` for streaming data analysis; `Hive`, which is a data warehouse software for querying and managing large datasets; `Pig`, which is a dataflow language for exploring large datasets; `Tez`[27] for executing complex directed-acyclic graph of data processing tasks; `Oozie`,[28] which is a workflow scheduler system for managing Hadoop jobs.

`Apache Spark` [57] is a software platform for Big Data analysis based on the in-memory processing model. A key difference from Hadoop and its approach of always storing intermediate data in distributed file systems, Spark stores its data in RAM and queries it repeatedly so as to obtain better performance for some class of applications (e.g., iterative machine learning algorithms). For many years, Hadoop has been considered the leading open source Big Data framework, but recently Spark has become the more popular framework so that it is supported by every major Hadoop vendors. In fact, for particular tasks, Spark is up to 100 times faster than Hadoop in memory and 10 times faster on disk. Several other libraries have been built on top of Spark: `Spark SQL` [58] for dealing with SQL and DataFrames, `MLib` [38] for machine learning, `GraphX` [19] for graphs and graph-parallel computation, and Spark Streaming to build scalable fault-tolerant streaming applications. For these reasons, Spark is becoming the primary execution engine for data processing and, in general, a must-have for Big Data applications. But even though in some applications Spark can be considered a better alternative to Hadoop, in many other applications it has some limitations that make it complementary to Hadoop. The main limitation of Spark is that it does not provide its own distributed and scalable storage system which is a fundamental requirement for Big Data applications that use huge and continually increasing volume of data stored across a very large number of nodes. To overcome this limit, Spark has been designed to run on top of several data sources, such as Cloud object storage (e.g., `Amazon S3 Storage`, `Swift Object Storage`), distributed filesystem (e.g., `HDFS`), NoSQL databases (e.g., `HBase`, `Cassandra`), and others.

`Sector/Sphere`[29] is a Cloud framework designed to implement data analysis applications involving large, geographically distributed datasets in which the data can be naturally processed in parallel [21]. The framework includes two components: a storage service called Sector, which manages the large distributed datasets with high reliability, high performance IO, and with uniform access, and a compute service called Sphere, which makes use of the Sector service to simplify data access, increase data IO bandwidth, and exploit wide area high performance networks. Both of them are available as open source software. Sphere is a compute service built on top of Sector and provides a set of programming interfaces to write distributed data analysis applications. Sphere takes streams as inputs and produces streams as outputs. A stream consists of multiple data segments that are processed by Sphere Processing Engines (SPEs) using slave nodes. Usually there are many more segments than SPEs. Each SPE takes a segment from a stream as an input and produces a segment of a stream as output. These output segments can in turn be the input segments of another Sphere process. Developers can use the Sphere client APIs to initialize input streams, upload processing function libraries, start Sphere processes, and read the processing results.

`H2O`[30] is an open source framework that provides a parallel processing engine which is equipped with math and machine learning libraries. It offers support for various programming lan-

[24] https://www.openstack.org/.
[25] https://wiki.openstack.org/wiki/Sahara.
[26] http://hadoop.apache.org/.
[27] https://tez.apache.org/.
[28] http://oozie.apache.org/.
[29] http://sector.sourceforge.net/.
[30] http://www.h2o.ai.

guages including Java, R, Python, and Scala. The machine learning algorithms are implemented on top of the H2O's distributed MapReduce framework and exploit the Java Fork/Join framework for implementing multi-threading. H2O implements many machine learning algorithms, such as generalized linear modeling (e.g., linear regression, logistic regression), Naïve Bayes, principal components analysis (PCA), time series, K-means clustering, neural networks, and others. H2O also implements complex data mining strategies such as Random Forest, Gradient Boosting, and Deep Learning. Users on H2O can build thousands of models and compare them to get the best prediction results. H2O runs on a several cloud platforms, including `Amazon EC2` and `S3` Storage, `Microsoft Azure` and `IBM DSX`.[31]

Microsoft introduced `AzureML` [48] as a machine learning framework solution which provides a cloud-based visual environment for constructing data analytics workflows. Azure ML is often described as a SaaS, however it can be seen also a PaaS since it can be used develop SaaS solutions on top of it. It is provided as a fully managed service by Microsoft where users neither need to buy any hardware/software nor manually manage any virtual machines. AzureML provides data scientists with a Web-based machine learning IDE for creating and automating machine learning workflows. In addition, it provides scalable and parallel implementations of popular machine learning techniques as well as data processing capabilities using a drag-and-drop interface. AzureML can read and import data from various sources including HTTP URL, Azure Blob Storage, Azure Table and Azure SQL Database. It also allows data scientists to import their own custom data analysis scripts (e.g., in R or Python). `Cumulon` [24] has been present as a system which is designed to help users rapidly develop and deploy matrix-based big-data analysis programs in the cloud. It provides an abstraction for distributed storage of matrices on top of HDFS. In particular, matrices are stored and accessed by tiles. A Cumulon program executes as a workflow of jobs. Each job reads a number of input matrices and writes a number of output matrices; input and output matrices must be disjoint. Dependencies among jobs are implied by dependent accesses to the same matrices. Dependent jobs execute in serial order. Each job executes multiple independent tasks that do not communicate with each other. Hadoop-based Cumulon inherits important features of Hadoop such as failure handling, and is able to leverage the vibrant Hadoop ecosystem. While targeting matrix operations, Cumulon can support programs that also contain traditional, non-matrix Hadoop jobs.

`SciDB` [11] has been introduced as an analytical database which is oriented toward the data management needs of scientific workflows. In particular, it mixes statistical and linear algebra operations with data management operations using a multi-dimensional array data model. SciDB supports both a functional (AFL) and a SQL-like query language (AQL) where AQL is compiled into AFL.

`MADlib` [22] provided a suite of SQL-based implementation for data mining and machine learning algorithms that are designed to get installed and run at scale within any relational database engine that supports extensible SQL, with no need for data import/export to other external tools. The analytics methods in MADlib are designed both for in- or out-of-core execution, and for the shared-nothing scale-out parallelism provided by modern parallel database engines, ensuring that computation is done near to the data. The core functionality of MADlib is written in declarative SQL statements, which orchestrate data movement to and from disk, and across networked computers.

`MLog` [32] has been presented as a high-level language that integrates machine learning into data management systems. It extends the query language over the SciDB data model [11] to allow users to specify machine learning models in a way similar to traditional relational views and relational queries. It is designed to manage all data movement, data persistence, and machine-learning related optimizations automatically. The data model of MLog is based on tensors instead of relations. In fact, all data in MLog are tensors and all operations are a subset of linear algebra over tensors.

4.2. SaaS data analysis frameworks

With the increasing need for data analysis requirements in several domains, a set of frameworks have been implemented to simplify and accelerate the process of developing Big Data analytics jobs by the end users. Those programming frameworks are tailored for implementing data analysis and machine learning applications as compositions of high-level services with the goal of reducing the programming burden and complexity.

High-level development environments Taking into account high-level frameworks that abstract from processing architecture, we must present the `R` system. `R`[32] is currently considered as the defacto standard in statistical and data analytics research. It is the most popular open source and cross platform software which has very wide community support. It is flexible, extensible and comprehensive for productivity. `R` provides a programming language which is used by statisticians and data scientists to conduct data analytics tasks and discover new insights from data by exploiting techniques such as clustering, regression, classification and text analysis. It is equipped with very rich and powerful library of packages. In particular, `R` provides a rich set of built-in as well as extended functions for data extraction, data cleaning, data loading, data transformation, statistical analysis, machine learning and visualization. In addition, it provides the ability to connect with other languages and systems (e.g., `Python`). In practice, a main drawback with `R` is that most of its packages were developed primarily for in-memory and interactive usage, i.e., for scenarios in which the data fit in memory. With the aim of tackling this challenge and providing the ability to handle massive datasets, several systems have been developed to support the execution of R programs on top of the distributed and scalable Big Data processing platforms such as `Hadoop` (e.g., `Ricardo` [17], `RHadoop`[33] and `RHIPE`,[34] `Segue`[35]) and `Spark` [53] (e.g., `SparkR` [51]). For example, `RHIPE` is an R package that brings the MapReduce framework to R users and enables them to access the Hadoop cluster from within the R environment. In particular, by using specific R functions, users are able to launch MapReduce jobs on the Hadoop cluster where the results can be easily retrieved from HDFS. `Segue` enables users to execute MapReduce jobs from within the R environment on Amazon Elastic MapReduce platforms. `SparkR` has become a popular `R` package that supports a light-weight frontend to execute `R` programs on top of the `Apache Spark` [53] distributed computation engine and allows executing large scale data analysis tasks from the `R` shell. `Pydoop` [31] is a Python package that provides an API for both the Hadoop framework and the HDFS. `Torch7` [16] has been presented as a mathematical environment and versatile numeric computing framework for building machine learning algorithms. `Theano` [6] has been presented as a linear algebra compiler that optimizes mathematical computations and produces efficient low-level implementations.

[31] https://datascience.ibm.com/.

[32] https://www.r-project.org/.

[33] https://github.com/RevolutionAnalytics/RHadoop.

[34] https://github.com/tesseradata/RHIPE.

[35] https://code.google.com/archive/p/segue/.

Please cite this article in press as: R. Elshawi et al., Big Data Systems Meet Machine Learning Challenges: Towards Big Data Science as a Service, Big Data Res. (2018), https://doi.org/10.1016/j.bdr.2018.04.004

Toolkits and libraries Among different toolkits, Apache Mahout [41] is an open-source toolkit which is designed to solve very practical and scalable machine learning problems on top of the Hadoop platform. Thus, Mahout is primarily meant for distributed and batch processing of massive sizes of data on a cluster. In particular, Mahout is essentially a set of Java libraries which is well integrated with Apache Hadoop and is designed to make machine learning applications easier to build. Recently, Mahout has been extended to provide support for machine learning algorithms for collaborative filtering and classification on top of Spark and H2O platforms. MLib [38] has been presented as the Spark's [53] distributed machine learning library that is well-suited for iterative machine learning tasks. It provides scalable implementations of standard learning algorithms for common learning settings including classification, regression, collaborative filtering, clustering, and dimensionality reduction. MLlib supports several languages (e.g., Java, Scala and Python) and provides a high-level API that leverages Spark's rich ecosystem to simplify the development of end-to-end machine learning pipelines.

Declarative interfaces/languages Several declarative machine learning implementations have been implemented on top of Big Data processing systems [8]. For example, Samsara [45], has been introduced as a mathematical environment that supports declarative implementation for general linear algebra and statistical operations as part of the Apache Mahout library. It allows its users to specify programs in R-like style using a set of common matrix abstractions and linear algebraic operations. Samsara compiles, optimizes and executes its programs on distributed dataflow systems (e.g., Apache Spark, Apache Flink, H2O). MLbase [27] has been implemented to provides a general-purpose machine learning library with a similar goal to Mahout's goal which is to provide a viable solution for dealing with large-scale machine learning tasks on top of the Spark framework. It supports a Pig Latin-like [18] declarative language to specify machine learning tasks and implements and provides a set of high-level operators that enable its users to implement a wide range of machine learning methods without deep systems knowledge. In addition, it implements an optimizer to select and dynamically adapt the choice of learning algorithm.

Apache SystemML [7] provides a declarative machine learning framework which is developed to run on top of Apache Spark. It supports R and Python-like syntax that includes statistical functions, linear algebra primitives and ML-specific constructs. It applies cost-based compilation techniques to generate efficient, low-level execution plans with in-memory single-node and large-scale distributed operations.

ScalOps [10] has been presented as a domain-specific language (DSL) that moves beyond single pass data analytics (i.e., MapReduce) to include multi-pass workloads, supporting iteration over algorithms expressed as relational queries on the training and model data. The physical execution plans of ScalOps consists of dataflow operators which are executed using the Hyracks data-intensive computing engine [9].

Mxnet [15] is a library that has been designed to ease the development of machine learning algorithms. It blends declarative symbolic expression with imperative tensor computation and offers auto differentiation to derive gradients. MXNet is designed to run on various heterogeneous systems, ranging from mobile devices to distributed GPU clusters.

Cloud machine learning services In general, one of the main advantage of working with cloud-based SaaS tools is that users do not have to worry about scaling their solution. Instead, ideally, the provided service should be able to automatically scale if the consumption of computing resources for the defined analytical models has increased and according to the user defined configurations and requirements. Google has provided a cloud-based SaaS machine learning platform[36] which is equipped with pre-trained models in addition to a platform to generate users' models. The service is integrated with other Google services such as Google Cloud Storage and Google Cloud Dataflow. It encapsulates powerful machine learning models that support different analytics applications (e.g. image analysis, speech recognition, text analysis and automatic translation) through REST API calls. Similarly, Amazon provides its machine learning as a service solution[37] (AML) which guides its users through the process of creating data analytics models without the need to learn complex algorithms or technologies. Once the models are created, the service makes it easy to perform predictions via simple APIs without the need to write any user code or manage any hardware or software infrastructure. AML works with data stored in Amazon S3, RDS or Redshift. It provides an API set for connecting with and manipulating other data sources. AML relies on Amazon SageMaker platform that allows the user to build, train, and deploy their machine learning models. IBM Watson Analytics[38] is another SaaS predictive analytic framework that allows its user to express their analytics job using natural English language. The service attempts to automatically spot interesting correlations and exceptions within the input data. It also provides suggestions on the various data cleaning steps and the adequate data visualization technique to use for various analysis scenarios.

The BigML[39] SaaS framework supports discovering predictive models from the input data using data classification and regression algorithms. In BigML, predictive models are presented to the users as an interactive decision tree which is dynamically visualized and explored within the BigML interface. BigML also provides a PaaS solution, BigML PredictServer,[40] which can be integrated with applications, services, and other data analysis tools. Hunk[41] is a commercial data analysis platform developed for rapidly exploring, analyzing and visualizing data in Hadoop and NoSQL data stores. Hunk uses a set of high-level user and programming interfaces to improve the speed and simplicity of getting insights from large unstructured and structured data sets. One of the key components of the Hunk architecture is the Splunk Virtual Index. This system decouples the storage tier from the data access and analytics tiers, so enabling Hunk to route requests to different data stores. The analytics tier is based on Splunks Search Processing Language (SPL) that is designed for data exploration across large, different data sets. The Hunk web framework allows building applications on top of the Hadoop Distributed File System (HDFS) and/or the NoSQL data store. Developers can use Hunk to build their Big Data applications on top of the data in Hadoop using a set of well known languages and frameworks. Indeed, the framework enables developers to integrate data and functionality from Hunk into enterprise Big Data applications using a web framework, documented REST API and software development kits for C#, Java, JavaScript, PHP and Ruby.

Kognitio Analytical Platform[42] has been designed as a Cloud service or supplied as a pre-integrated appliance that allows users to pull very large amounts of data from existing data storage systems into high-speed computer memory, allowing complex analytical questions to be answered interactively. Although Kognitio has its own internal disk subsystem, it is primarily used as an analytical layer on top of existing storage/data processing

[36] https://cloud.google.com/products/machine-learning/.
[37] https://aws.amazon.com/machine-learning/.
[38] https://www.ibm.com/analytics/watson-analytics/.
[39] https://bigml.com.
[40] https://bigml.com/predictserver.
[41] http://www.splunk.com/en_us/products/hunk.html.
[42] www.kognitio.com.

Please cite this article in press as: R. Elshawi et al., Big Data Systems Meet Machine Learning Challenges: Towards Big Data Science as a Service, Big Data Res. (2018), https://doi.org/10.1016/j.bdr.2018.04.004

systems, e.g., Hadoop clusters and/or existing traditional disk-based data warehouse products, Cloud storage, etc. A feature called External Tables allows persistent data to reside on external systems. Using this feature the system administrator, or a privileged user, can easily setup access to data that resides in another environment, typically a disk store such as the above-mentioned Hadoop clusters and data warehouse systems. To a final user, the Kognitio Analytical Platform looks like a relational database management system (RDBMS) that is similar to many commercial databases. However, unlike these databases, Kognitio has been designed specifically to handle analytical query workload, as opposed to the more traditional on-line transaction processing (OLTP) workload.

Nubytics is a Software-as-a-Service (SaaS) system that exploits Cloud facilities to provide efficient services for analyzing large datasets [13]. The system allows users to import their data to the Cloud, extract knowledge models using high performance data mining services, and exploit the inferred knowledge to predict new data and behaviors. In particular, Nubytics provides data classification and regression services that can be used in a variety of scientific and business applications. Scalability is ensured by a parallel computing approach that fully exploits the resources available on a Cloud. Nubytics differs from general purpose data analysis frameworks like Azure ML, Hadoop and Sparks, or data-oriented workflow management systems like DMCF, as it provides specialized services for data classification and prediction. These services are provided by a Web interface that allows data analysts to focus on the data analysis process without worrying on low level programming details. This approach is similar to that adopted by BigML. However, Nubytics also focuses on scalability, by implementing an ad hoc parallel computing approach that fully exploits the distributed resources of a Cloud computing platform.

Workflow environments In principle, workflows are used as an effective paradigm for data analysis programming. Several frameworks use this paradigm for integrating data analysis methods. They are mainly focusing on simplifying the process of orchestrating different component and reducing the time to production. Here, we discuss just a few systems that are representative of this class of frameworks.

Tensorflow [1] provides an interface for designing machine learning algorithms, and an implementation for executing such algorithms. In particular, Tensorflow takes computations described using a dataflow-like model and compiles them onto several hardware platforms, ranging from running inference on mobile device platforms (e.g., Android and iOS) to large-scale distributed systems of hundreds of machines and thousands of computational devices such as GPU cards. The main focus of Tensorflow is to simplify the real-world use of machine learning system and significantly reducing the maintenance burdens. TFX [64] has been presented by Google as a TensorFlow-based general-purpose machine learning platform that integrates different components including a learner for generating models based on training data, modules for analyzing and validating both data as well as models in addition to infrastructure for serving models in production. Keras[43] is a Python deep learning library which is capable of running on top of TensorFlow or Theano. It allows for easy and fast prototyping through user friendliness, modularity and extensibility. Böse et al. [65] presented a platform for large-scale machine learning (ML) approaches that enables the training and application of probabilistic demand forecasting models. The platform comprises of an end-to-end machine learning system, built on Apache Spark, that includes data preprocessing, feature engineering, distributed learning, evaluation, experimentation and ensembling.

The F2 analytics framework [20] has been designed to separate execution from data management and handles compute and data as equal first-class citizens. In particular, in this framework, data is managed separately while decisions to determine how data is partitioned or when it is to be processed are taken at runtime. The computation that processes the data can have lost semantics and run any of the available operations on whatever data is ready. One of the main advantages of this framework design is that it provides more flexibility in expressing analytics jobs by removing concerns regarding data partitioning, routing and what logic to specify during the runtime.

The Data Mining Cloud Framework (DMCF) [34] is a software system implemented for designing and executing data analysis workflows on Clouds. A Web-based user interface allows users to compose their applications and submit them for execution over Cloud resources, according to a Software-as-a-Service (SaaS) approach. The DMCF architecture has been designed to be deployed on different Cloud settings. Currently, there are two different deployments of DMCF: *i*) on top of a Platform-as-a-Service (PaaS) cloud, i.e., using storage, compute, and network APIs that hide the underlying infrastructure layer; *ii*) on top of an Infrastructure-as-a-Service (IaaS) cloud, i.e., using virtual machine images (VMs) that are deployed on the infrastructure layer. In both deployment scenarios, DMCF uses Microsoft Azure[44] as the cloud provider. The DMCF software modules can be grouped into *web components* and *compute components*. DMCF allows users to compose, check, and run data analysis workflows through a HTML5 web editor. The workflows can be defined using two languages: *VL4Cloud* (Visual Language for Cloud) [34] and *JS4Cloud* (JavaScript for Cloud) [35]. Both languages use three key abstractions: *Data* elements, representing input files (e.g., a dataset to be analyzed) or output files (e.g., a data mining model); *Tool* elements, representing software tools used to perform operations on data elements (partitioning, filtering, mining, etc.); and *Tasks*, which represent the execution of Tool elements on given input Data elements to produce some output Data elements.

Keystoneml framework [46] has been designed to support building complex and multi-stage pipelines that include feature extraction, dimensionality reduction, data transformations and training supervised learning models. It provides a high-level, type-safe API that is built around logical operators to capture end-to-end machine learning applications. To optimize the machine learning pipelines, Keystoneml applies techniques to do both per-operator optimization and end-to-end pipeline optimization. It uses a cost-based optimizer that accounts for both computation and communication costs. The optimizer is also able to determine which intermediate states should be materialized in the main memory during the iterative execution over the raw data. MBDAaaS [75] has been proposed as a framework for Model-based Big Data Analytics-as-a-Service that supports users with limited Big Data expertise in deploying data analytics pipelines. It provides a declarative model for specifying the goals of a given analytics in the form of pairs indicators/objectives which are then used to incrementally refine a platform-independent procedural model for specifying how analytics should be carried out in terms of an abstract workflow. The procedural models are then compiled in a ready-to-be-executed deployment model that semi-automatically suggests platform-dependent component configurations and supporting automatic provisioning of computational components and resources. To complete our discussion and offer a look of comparison, Table 1 summarizes the main features of the Big Data analysis frameworks we presented.

[43] https://keras.io/.

[44] http://azure.microsoft.com.

Please cite this article in press as: R. Elshawi et al., Big Data Systems Meet Machine Learning Challenges: Towards Big Data Science as a Service, Big Data Res. (2018), https://doi.org/10.1016/j.bdr.2018.04.004

Table 1
Summary of the main feature of Big Data Analysis Frameworks.

Framework	Model	Abstraction	Supported languages	Underlying engine	Availability
Hadoop	PaaS	MapReduce	Java, Python	Hadoop	Open source
Spark	PaaS	RDD	Scala, Python, Java, R	Spark, Yarn	Open source
H2O	PaaS	Procedural + Library	REST, R, Python	H2O	Open source
SciDB	PaaS	Declarative	SQL	SciDB	Open source
AzureML	SaaS/PaaS	Visual User Interface	REST	Microsoft Azure	Proprietary/Microsoft
R	SaaS/PaaS	Procedural	R	R	Open source
SparkR	SaaS/PaaS	Procedural	R	Spark	Open source
Mahout	SaaS/PaaS	Toolkit	Java, Scala	Hadoop, Spark	Open source
Spark MLib	SaaS/PaaS	Library	Scala, Python	Spark	Open Source
Samsara	SaaS/PaaS	Declarative	Java, Scala	Spark, Flink, H2O	Open Source
Apache SystemML	SaaS/PaaS	Declarative	R, Python	Spark	Open Source
Google ML	SaaS	Visual User Interface	Python	Google Cloud Dataflow	Proprietary/Google
Amazon ML	SaaS	Visual User Interface	N/A	Apache MXNet, TensorFlow, PyTorch	Proprietary/Amazon
BigML	SaaS/Paas	Visual User Interface	Python	BigML PredictServer	Proprietary/BigML
Tensorflow	SaaS	Visual User Interface/Library	Python, Haskell, Java, Go Julia, R, Scala	CUDA, TPU	Open Source
KeystoneML	SaaS/PaaS	Procedural + Library	Scala	Spark	Open Source

5. Discussion and open challenges

The world is progressively moving towards being populated by a data-driven society where data are the most valuable asset. The proliferation of Big Data and big computing have boosted the adoption of machine learning and data science across several application domains. For example, image recognition systems have reached and sometimes outperformed human quality,[45] voice-driven personal assistants (e.g., `Amazon Alexa`[46]) are now available and the dream of autonomous vehicles is currently becoming a reality.[47] In practice, efficient and effective analysis and exploitation of Big Data have become essential requirements for enhancing the competitiveness of enterprises and maintaining sustained social and economic growth of societies and countries. Therefore, Big Data Science has become a very active research domain with crucial impact on various scientific and business domains where it is significant to analyze massive and complex amounts of data. In practice, in many cases, the data to be analyzed can be stored in cloud-based data servers and elastic computing cloud resources can be exploited to facilitate the speeding up and scaling out of the data science tasks.

In spite of the high expectations on the promises and potential benefits of Big Data Science, there are still many challenges to overcome so that we are able to fully harness its full power. Examples of these open challenges and research directions include:

- *Data availability and data sharing*: In practice, Big Data science lives and dies by the data. It mainly rests on the availability of massive datasets, of that there can be no doubt. The more data that is available, the richer the insights and the results that Big Data science can produce. The bigger and more diverse the data set, the better the analysis can model the real world. Therefore, any successful Big Data science process has attempted to incorporate as many data sets from internal and public sources as possible. In reality, data is segmented, siloed and under the control of different individuals, departments or organizations. It is crucially required to motivate all parties to work collaboratively and share useful data/insights for the public. Recently, there has been an increasing trend for open data initiatives which supports the idea of making data publicly available to everyone to use and republish as they wish, without restrictions from copyright, patents or other mechanisms of control [25]. Online data markets [4] are emerging cloud-based services (e.g., `Azure Data Market`,[48] `Kaggle`,[49] `Connect`,[50] `Socrata`[51]). For example, Kaggle is a platform where companies can provide data to a community of data scientists so that they can analyze the data with the aim of discovering predictive, actionable insights and win incentive awards. In particular, such platforms follow a model where data and rewards are traded for innovation. More research, effort and development is still required in this direction.
- *Interoperability*: With the increasing number of platforms and services, interoperability is arising as a main issue. Standard formats and models are required to enable interoperability and ease cooperation among the various platforms and services. In addition, the service-oriented paradigm can play an effective role in supporting the execution of large-scale distributed analytics on heterogeneous platforms along with software components developed using various programming languages or tools. Furthermore, in practice, the majority of existing big-data-processing platforms (e.g., `Hadoop` and `Spark`) are designed based on the single-cluster setup with the assumptions of centralized management and homogeneous connectivity which makes them sub-optimal and sometimes infeasible to apply for scenarios that require implementing data analytics jobs on highly distributed data sets (e.g., across racks, clusters, data centers or multi-organizations). Some scenarios can also require distributing data analysis tasks in a hybrid mode among local processing of local data sources and model exchange and fusion mechanisms to compose the results produced in the distributed nodes.
- *Efficient distributed execution mechanisms*: In practice, the design of most of the statistical computation (e.g., `R`) and scientific computing tools (e.g. `Python`) is memory-bounded where data analysis algorithms rely on the in-memory data processing mechanism. While this approach may bring many benefits in terms of speeding up the processing and thus subsequently resulting in faster decisions being made, with Big Data sizes there could be scalability risks due to performance

[45] http://theconversation.com/digital-diagnosis-intelligent-machines-do-a-better-job-than-humans-53116.
[46] https://developer.amazon.com/alexa.
[47] https://www.technologyreview.com/s/609450/autonomous-vehicles-are-you-ready-for-the-new-ride/.
[48] http://datamarket.azure.com/browse/data.
[49] https://www.kaggle.com/.
[50] https://connect.data.com/.
[51] https://socrata.com/.

Please cite this article in press as: R. Elshawi et al., Big Data Systems Meet Machine Learning Challenges: Towards Big Data Science as a Service, Big Data Res. (2018), https://doi.org/10.1016/j.bdr.2018.04.004

issues if the processed data do not fit in the available main memory or it can be very costly if the required memory can be allocated in a cloud platform. Efficient and optimized distributed and parallel disk-based execution platforms for complex data analysis jobs (e.g., SparkR) are crucially required to tackle this challenge.

- *Iterative and explorative natures of data analytics process*: In general, a major obstacle for supporting Big Data analytics applications is the challenging and time consuming process of identifying and training an adequate predictive model. Therefore, data science is a highly iterative exploratory process where most scientists work hard to find the best model or algorithm that meets their data challenge. In practice, there is no one-model-fits-all solutions, thus, there is no single model or algorithm that can handle all data set varieties and changes in data that may occur over time. All machine learning algorithms require user defined inputs to achieve a balance between accuracy and generalizability. This task is referred to as parameter tuning. The tuning parameters impact the way the algorithm searches for the optimal solution. This iterative and explorative nature of the model building process is prohibitively expensive with very large datasets. Thus, recent research efforts (e.g., Auto-WEKA[52]) have been attempting to automate this process [62]. However, they have mainly focused on single node implementations and have assumed that model training itself is a black box, limiting their usefulness for applications driven by large-scale datasets [30].
- *Model management*: With the increasing usage of machine learning and the increasing number of models, the issues of model management, model sharing, model versioning and lifecycle management have become significantly important. For example, it is important to keep track of the models developed and understand the differences between them by recording their metadata (e.g., training sample, hyperparameters). ModelHub [39] has been proposed to provide a model versioning system to store and query the models and their versions, a domain specific language that serves as an abstraction layer for searching through model space in addition to a hosted service to store developed models, explore existing models, enumerate new models and share models with others. ModelDB [50] is another system for managing machine learning models that automatically tracks the models in their native environments (e.g. Mahout, SparkML), indexes them and allows flexible exploration of models using either SQL or a visual web-based interface. Along with models and pipelines, ModelDB stores several metadata (e.g., parameters of pre-processing steps, hyperparameters for models etc.) and quality metrics (e.g. AUC, accuracy). In addition, it can store the training and test data for each model.
- *Programming abstractions*: Despite the availability of several programming approaches that have been developed for implementing data analysis applications, new programming abstractions for Big Data analysis are still needed to simplify the task of programmers, reduce the code development time and close the gap between the data analysis algorithms and the scalable computing platforms on which they are executed [63]. Big Data analytics programming languages require novel complex abstract structures that must be close to data format and organization. Data exchange and transformation, data locality and near-data processing constructs are welcome to analize Big Data repositories and large streams. More research activities are needed to develop scalable higher-level models and paradigms that must be driven by data mining and machine learning techniques.
- *Usability*: In practice, building machine learning applications is a highly time-consuming process that requires substantial effort even from best-trained data scientists to deploy, operate and monitor. One of the main reasons behind this challenge is the lack of tools for supporting end-to-end machine learning application development that can ease and accelerate the job for end users. The DAWN project at Stanford [2] has recently announced its vision for the next five years with the aim of making the machine learning (ML) process *usable* for small teams of non-ML experts so that they can easily apply ML to their problems, achieve high-quality results and deploy production systems that can be used in critical applications. The main design philosophy of the DAWN project is to target the management of end-to-end ML workflows, empower domain experts to easily develop and deploy their models and perform effective optimization of the workflow execution pipelines using simple interfaces. Another important usability aspect is the *explainability* of the developed models. In general, explainability is very useful for machine learning models used and trained as blackboxes where the output models are not easy or intuitive to explain (e.g., SVM, neural networks, deep learning). For example, in the healthcare domain, the physicians should be able to understand and interpret why the developed models are meaningful and applicable. We believe that additional research efforts are crucially required to ease and accelerate the life cycle and the data science process and make it more usable.

Providing new solutions in all those research fields will promote innovative Big Data science and will allow new applications and procedures to be implemented by scientists and professions in research centers and companies where exploitation of machine intelligence and data analysis is greatly important for solving complex and challenging problems.

References

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al., Tensorflow: a system for large-scale machine learning, in: OSDI, vol. 16, 2016, pp. 265–283.

[2] P. Bailis, K. Olukoton, C. Re, M. Zaharia, Infrastructure for usable machine learning: the Stanford dawn project, arXiv preprint, arXiv:1705.07538, 2017.

[3] M. Baker, Data science: industry allure, Nature 520 (2015) 253–255.

[4] M. Balazinska, B. Howe, D. Suciu, Data markets in the cloud: an opportunity for the database community, Proc. VLDB Endow. 4 (12) (2011) 1482–1485.

[5] P. Banerjee, C. Bash, R. Friedrich, P. Goldsack, B.A. Huberman, J. Manley, C. Patel, P. Ranganathan, A. Veitch, Everything as a service: powering the new information economy, Computer 44 (3) (2011) 36–43.

[6] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. Goodfellow, A. Bergeron, N. Bouchard, D. Warde-Farley, Y. Bengio, Theano: new features and speed improvements, arXiv preprint, arXiv:1211.5590, 2012.

[7] M. Boehm, M.W. Dusenberry, D. Eriksson, A.V. Evfimievski, F.M. Manshadi, N. Pansare, B. Reinwald, F.R. Reiss, P. Sen, A.C. Surve, et al., SystemML: declarative machine learning on spark, Proc. VLDB Endow. 9 (13) (2016) 1425–1436.

[8] M. Boehm, A.V. Evfimievski, N. Pansare, B. Reinwald, Declarative machine learning – a classification of basic properties and types, arXiv preprint, arXiv:1605.05826, 2016.

[9] V. Borkar, M. Carey, R. Grover, N. Onose, R. Vernica, Hyracks: a flexible and extensible foundation for data-intensive computing, in: 2011 IEEE 27th International Conference on Data Engineering, ICDE, IEEE, 2011, pp. 1151–1162.

[10] V.R. Borkar, Y. Bu, M.J. Carey, J. Rosen, N. Polyzotis, T. Condie, M. Weimer, R. Ramakrishnan, G. Dror, N. Koenigstein, et al., Declarative systems for large-scale machine learning, IEEE Data Eng. Bull. 35 (2) (2012) 24–32.

[52] https://www.cs.ubc.ca/labs/beta/Projects/autoweka/.

Please cite this article in press as: R. Elshawi et al., Big Data Systems Meet Machine Learning Challenges: Towards Big Data Science as a Service, Big Data Res. (2018), https://doi.org/10.1016/j.bdr.2018.04.004

[11] P.G. Brown, Overview of SciDB: large scale array storage, processing and analysis, in: Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data, ACM, 2010, pp. 963–968.
[12] R. Cattel, Scalable SQL and NoSQL data stores, SIGMOD Rec. 39 (4) (2010) 12–27.
[13] E. Cesario, A. Iannazzo, F. Marozzo, F. Morello, D. Talia, P. Trunfio, Nubytics: scalable cloud services for data analysis and prediction, in: 2nd International Forum on Research and Technologies for Society and Industry, RTSI 2016, September 2016, pp. 1–6.
[14] P. Carbone, A. Katsifodimos, S. Ewen, V. Markl, S. Haridi, K. Tzoumas, Apache Flink™: stream and batch processing in a single engine, IEEE Data Eng. Bull. 38 (4) (2015) 28–38.
[15] T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, Z. Zhang, MXNet: a flexible and efficient machine learning library for heterogeneous distributed systems, arXiv preprint, arXiv:1512.01274, 2015.
[16] R. Collobert, K. Kavukcuoglu, C. Farabet, Torch7: a Matlab-like environment for machine learning, in: BigLearn, NIPS Workshop, 2011, number EPFL-CONF-192376.
[17] S. Das, Y. Sismanis, K.S. Beyer, R. Gemulla, P.J. Haas, J. McPherson, Ricardo: integrating R and Hadoop, in: Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data, ACM, 2010, pp. 987–998.
[18] A. Gates, O. Natkovich, S. Chopra, P. Kamath, S. Narayanam, C. Olston, B. Reed, S. Srinivasan, U. Srivastava, Building a highlevel dataflow system on top of MapReduce: the Pig experience, Proc. VLDB Endow. 2 (2) (2009) 1414–1425.
[19] J.E. Gonzalez, R.S. Xin, A. Dave, D. Crankshaw, M.J. Franklin, I. Stoica, GraphX: graph processing in a distributed dataflow framework, in: OSDI, 2014.
[20] R. Grandl, A. Singhvi, A. Akella, Fast and flexible data analytics with F2, arXiv preprint, arXiv:1703.10272, 2017.
[21] Y. Gu, R.L. Grossman, Sector and sphere: the design and implementation of a high-performance data cloud, Philos. Trans. A Math. Phys. Eng. Sci. 367 (1897) (2009) 2429–2445.
[22] J.M. Hellerstein, C. Ré, F. Schoppmann, D.Z. Wang, E. Fratkin, A. Gorajek, K.S. Ng, C. Welton, X. Feng, K. Li, et al., The MADlib analytics library: or MAD skills, the SQL, Proc. VLDB Endow. 5 (12) (2012) 1700–1711.
[23] T. Hey, S. Tansley, K. Tolle (Eds.), The Fourth Paradigm: Data-Intensive Scientific Discovery, Microsoft Research, October 2009.
[24] B. Huang, S. Babu, J. Yang, Cumulon: optimizing statistical data analysis in the cloud, in: Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data, ACM, 2013, pp. 1–12.
[25] N. Huijboom, T. Van den Broek, Open data: an international comparison of strategies, Eur.J. ePract. 12 (1) (2011) 4–16.
[26] M. Kornacker, A. Behm, V. Bittorf, T. Bobrovytsky, C. Ching, A. Choi, J. Erickson, M. Grund, D. Hecht, M. Jacobs, I. Joshi, L. Kuff, D. Kumar, A. Leblang, N. Li, I. Pandis, H. Robinson, D. Rorke, S. Rus, J. Russell, D. Tsirogiannis, S. Wanderman-Milne, M. Yoder, Impala: a modern, open-source SQL engine for Hadoop, in: CIDR, 2015.
[27] T. Kraska, A. Talwalkar, J.C. Duchi, R. Griffith, M.J. Franklin, M.I. Jordan, MLBase: a distributed machine-learning system, in: CIDR, 2013.
[28] J. Kreps, N. Narkhede, J. Rao, et al., Kafka: a distributed messaging system for log processing, in: Proceedings of the NetDB, 2011, pp. 1–7.
[29] S. Kulkarni, N. Bhagat, M. Fu, V. Kedigehalli, C. Kellogg, S. Mittal, J.M. Patel, K. Ramasamy, S. Taneja, Twitter heron: stream processing at scale, in: Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, ACM, 2015, pp. 239–250.
[30] A. Kumar, R. McCann, J.F. Naughton, J.M. Patel, Model selection management systems: the next frontier of advanced analytics, SIGMOD Rec. 44 (4) (2015) 17–22.
[31] S. Leo, G. Zanetti, Pydoop: a Python MapReduce and HDFS API for Hadoop, in: Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing, ACM, 2010, pp. 819–825.
[32] X. Li, B. Cui, Y. Chen, W. Wu, C. Zhang, MLog: towards declarative in-database machine learning, Proc. VLDB Endow. 10 (12) (2017) 1933–1936.
[33] Y. Low, J. Gonzalez, A. Kyrola, D. Bickson, C. Guestrin, J.M. Hellerstein, Distributed GraphLab: a framework for machine learning in the cloud, Proc. VLDB Endow. 5 (8) (2012).
[34] F. Marozzo, D. Talia, P. Trunfio, A workflow management system for scalable data mining on clouds, in: IEEE Transactions on Services Computing, 2016.
[35] F. Marozzo, D. Talia, P. Trunfio, JS4Cloud: script-based workflow programming for scalable data analysis on cloud platforms, Concurr. Comput., Pract. Exp. 27 (17) (2015) 5214–5237.
[36] J. Manyika, M. Chui, B. Brown, J. Bughin, R. Dobbs, C. Roxburgh, A.H. Byers, Big Data: The Next Frontier for Innovation, Competition, and Productivity, Technical report, McKinsey Global Institute, June 2011.
[37] V. Mayer-Schönberger, K. Cukier, Big Data: A Revolution that Will Transform How We Live, Work, and Think, Houghton Mifflin Harcourt, 2013.
[38] X. Meng, J. Bradley, B. Yuvaz, E. Sparks, S. Venkataraman, D. Liu, J. Freeman, D. Tsai, M. Amde, S. Owen, et al., MLlib: machine learning in apache spark, J. Mach. Learn. Res. 17 (34) (2016) 1–7.
[39] H. Miao, A. Li, L.S. Davis, A. Deshpande, ModelHub: towards unified data and lifecycle management for deep learning, arXiv preprint, arXiv:1611.06224, 2016.
[40] S.A. Noghabi, K. Paramasivam, Y. Pan, N. Ramesh, J. Bringhurst, I. Gupta, R.H. Campbell, Samza: stateful scalable stream processing at linkedin, Proc. VLDB Endow. 10 (12) (2017) 1634–1645.
[41] S. Owen, S. Owen, Mahout in Action, 2012.
[42] S. Sakr, Cloud-hosted databases: technologies, challenges and opportunities, Clust. Comput. 17 (2) (2014) 487–502.
[43] S. Sakr, Big Data 2.0 Processing Systems – A Survey, Springer Briefs in Computer Science, Springer, 2016.
[44] S. Sakr, F.M. Orakzai, I. Abdelaziz, Z. Khayyat, Large-Scale Graph Processing Using Apache Giraph, Springer, 2016.
[45] S. Schelter, A. Palumbo, S. Quinn, S. Marthi, A. Musselman, Samsara: declarative machine learning on distributed dataflow systems, in: NIPS Workshop MLSystems, 2016.
[46] E.R. Sparks, S. Venkataraman, T. Kaftan, M.J. Franklin, B. Recht, KeystoneML: optimizing pipelines for large-scale advanced analytics, in: 2017 IEEE 33rd International Conference on Data Engineering, ICDE, IEEE, 2017, pp. 535–546.
[47] D. Talia, P. Trunfio, F. Marozzo, Data Analysis in the Cloud, Elsevier, 2016.
[48] A. Team, AzureML: anatomy of a machine learning service, in: Proceedings of The 2nd International Conference on Predictive APIs and Apps, 2016, pp. 1–13.
[49] A. Thusoo, Z. Shao, S. Anthony, D. Borthakur, N. Jain, J.S. Sarma, R. Murthy, H. Liu, Data warehousing and analytics infrastructure at facebook, in: SIGMOD, 2010.
[50] M. Vartak, H. Subramanyam, W.-E. Lee, S. Viswanathan, S. Husnoo, S. Madden, M. Zaharia, ModelDB: a system for machine learning model management, in: Proceedings of the Workshop on Human-In-the-Loop Data Analytics, ACM, 2016, p. 14.
[51] S. Venkataraman, Z. Yang, D. Liu, E. Liang, H. Falaki, X. Meng, R. Xin, A. Ghodsi, M.J. Franklin, I. Stoica, M. Zaharia, SparkR: scaling R programs with Spark, in: SIGMOD, 2016.
[52] T. White, Hadoop: The Definitive Guide, O'Reilly Media, 2012.
[53] M. Zaharia, M. Chowdhury, M.J. Franklin, S. Shenker, I. Stoica, Spark: cluster computing with working sets, in: HotCloud, 2010.
[54] L. Zhao, S. Sakr, A. Liu, A. Bouguettaya, Cloud Data Management, Springer, 2014.
[55] A.Y. Zomaya, S. Sakr, Handbook of Big Data Technologies, Springer, 2017.
[56] I.A.T. Hashem, I. Yaqoob, N.B. Anuar, S. Mokhtar, A. Gani, S.U. Khan, The rise of "big data" on cloud computing: review and open research issues, Inf. Sci. 47 (2015) 98–115.
[57] M. Zaharia, R.S. Xin, P. Wendell, T. Das, M. Armbrust, A. Dave, X. Meng, J. Rosen, S. Venkataraman, M.J. Franklin, et al., Apache Spark: a unified engine for big data processing, Commun. ACM 59 (11) (2016) 56–65.
[58] M. Armbrust, R.S. Xin, C. Lian, Y. Huai, D. Liu, J.K. Bradley, X. Meng, T. Kaftan, M.J. Franklin, A. Ghodsi, et al., Spark SQL: relational data processing in Spark, in: Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, ACM, 2015, pp. 1383–1394.
[59] J. Dean, S. Ghemawat, MapReduce: simplified data processing on large clusters, Commun. ACM 51 (1) (2008) 107–113.
[60] S. Sakr, A. Liu, D.M. Batista, M. Alomari, A survey of large scale data management approaches in cloud environments, IEEE Commun. Surv. Tutor. 13 (3) (2011) 311–336.
[61] Sherif Sakr, Liang Zhao, Hiroshi Wada, Anna Liu, CloudDB AutoAdmin: towards a truly elastic cloud-based data store, in: 2011 IEEE International Conference on Web Services, ICWS, IEEE, 2011, pp. 732–733.
[62] C. Thornton, F. Hutter, H.H. Hoos, K. Leyton-Brown, Auto-WEKA: combined selection and hyperparameter optimization of classification algorithms, in: Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2013, pp. 847–855.
[63] D. Talia, Clouds for scalable big data analytics, Computer 46 (5) (2013) 98–101.
[64] Denis Baylor, Eric Breck, Heng-Tze Cheng, Noah Fiedel, Chuan Yu Foo, Zakaria Haque, Salem Haykal, Mustafa Ispir, Vihan Jain, Levent Koc, et al., TFX: a tensorflow-based production-scale machine learning platform, in: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2017, pp. 1387–1395.
[65] Joos-Hendrik Böse, Valentin Flunkert, Jan Gasthaus, Tim Januschowski, Dustin Lange, David Salinas, Sebastian Schelter, Matthias Seeger, Yuyang Wang, Probabilistic demand forecasting at scale, Proc. VLDB Endow. 10 (12) (2017) 1694–1705.
[66] C.P. Chen, C.-Y. Zhang, Data-intensive applications, challenges, techniques and technologies: a survey on big data, Inf. Sci. 275 (2014) 314–347.
[67] M. Chen, S. Mao, Y. Liu, Big data: a survey, Mob. Netw. Appl. 19 (2) (2014) 171–209.
[68] A. Fahad, N. Alshatri, Z. Tari, A. Alamri, I. Khalil, A.Y. Zomaya, S. Foufou, A. Bouras, A survey of clustering algorithms for big data: taxonomy and empirical analysis, IEEE Trans. Emerg. Topics Comput. 2 (3) (2014) 267–279.
[69] A. Gandomi, M. Haider, Beyond the hype: big data concepts, methods, and analytics, Int. J. Inf. Manag. 35 (2) (2015) 137–144.
[70] H. Hu, Y. Wen, T.-S. Chua, X. Li, Toward scalable systems for big data analytics: a technology tutorial, IEEE Access 2 (2014) 652–687.

Please cite this article in press as: R. Elshawi et al., Big Data Systems Meet Machine Learning Challenges: Towards Big Data Science as a Service, Big Data Res. (2018), https://doi.org/10.1016/j.bdr.2018.04.004

[71] S. Landset, T.M. Khoshgoftaar, A.N. Richter, T. Hasanin, A survey of open source tools for machine learning with big data in the Hadoop ecosystem, J. Big Data 2 (1) (2015) 24.

[72] S. Sakr, A. Liu, A.G. Fayoumi, The family of MapReduce and large-scale data processing systems, ACM Comput. Surv. (CSUR) 46 (1) (2013) 11.

[73] D. Singh, C.K. Reddy, A survey on platforms for big data analytics, J. Big Data 2 (1) (2015) 8.

[74] C.-W. Tsai, C.-F. Lai, H.-C. Chao, A.V. Vasilakos, Big data analytics: a survey, J. Big Data 2 (1) (2015) 21.

[75] Claudio Agostino Ardagna, Valerio Bellandi, Michele Bezzi, Paolo Ceravolo, Ernesto Damiani, Cedric Hebert, Model-based big data analytics-as-a-service: take big data to the next level, IEEE Trans. Serv. Comput. (2018), https://doi.org/10.1109/TSC.2018.2816941.

Please cite this article in press as: R. Elshawi et al., Big Data Systems Meet Machine Learning Challenges: Towards Big Data Science as a Service, Big Data Res. (2018), https://doi.org/10.1016/j.bdr.2018.04.004