A five-layer architecture for big data processing and analytics

Julie Yixuan Zhu*

Department of Electrical and Electronic Engineering,
The University of Hong Kong, Hong Kong
Email: yxzhu@eee.hku.hk
*Corresponding author

Bo Tang

Department of Computer Science and Engineering,
Southern University of Science and Technology,
1088 Xueyuan Ave, Nanshan Qu, Shenzhen Shi,
Guangdong Sheng, China
Email: tangb3@sustc.edu.cn

Victor O.K. Li

Department of Electrical and Electronic Engineering,
The University of Hong Kong, Hong Kong
Email: vli@eee.hku.hk

Abstract: Big data technologies have attracted much attention in recent years. The academia and industry have reached a consensus, that is, the ultimate goal of big data is about transforming 'big data' to 'real value'. In this article, we discuss how to achieve this goal and propose five-layer architecture for big data processing and analytics (BDPA), including a collection layer, a storage layer, a processing layer, an analytics layer, and an application layer. The five-layer architecture targets to set up a de facto standard for current BDPA solutions, to collect, manage, process, and analyse the vast volume of both static data and online data streams, and make valuable decisions for all types of industries. Functionalities and challenges of the five-layers are illustrated, with the most recent technologies and solutions discussed accordingly. We conclude with the requirements for the future BDPA solutions, which may serve as a foundation for the future big data ecosystem.

Keywords: big data processing and analytics; BDPA; online big data stream; five-layer architecture.

Reference to this paper should be made as follows: Zhu, J.Y., Tang, B. and Li, V.O.K. (2019) 'A five-layer architecture for big data processing and analytics', *Int. J. Big Data Intelligence*, Vol. 6, No. 1, pp.38–49.

Biographical notes: Julie Yixuan Zhu received her PhD in the Department of Electrical and Electronic Engineering, The University of Hong Kong (HKU), in 2016. She received her BE in Electronic Engineering from the Tsinghua University, Beijing, China, in 2012. Her research interests include urban computing, spatio-temporal data mining, indoor localisation, and time series data analytics.

Bo Tang received his PhD in Computer Science from The Hong Kong Polytechnic University in 2017. He is currently an Assistant Professor in Southern University of Science and Technology, China. His research interests include similarity search on high dimensional dataset and data exploration on multidimensional dataset.

Victor O.K. Li received his SB, SM, EE and ScD degrees in Electrical Engineering and Computer Science from MIT in 1977, 1979, 1980, and 1981, respectively. He is the Chair of Information Engineering, Cheng Yu-Tung Professor in Sustainable Development, and Head of the Department of Electrical and Electronic Engineering at the University of Hong Kong (HKU). He has received numerous awards, including the PRC Ministry of Education Changjiang Chair Professorship at Tsinghua University, the UK Royal Academy of Engineering Senior Visiting Fellowship in Communications, the Croucher Foundation Senior Research Fellowship, and the Order of the Bronze Bauhinia Star.

This paper is a revised and expanded version of a paper entitled 'A four-layer architecture for online and historical big data analytics' presented at 2nd International Conference on Big Data Intelligence and Computing (DataCom), Auckland, New Zealand, 8–12 August 2016.

1 Introduction

With the world's data doubling every two years (Gantz and Reinsel, 2012), increasingly more applications require the capability of handling 'big data'. These applications are of various purposes including finance, public health, the 'internet of things', transportation, social media, sensor networks, manufacturing, networking, telecommunications, etc. For IT-related industry, different online service providers (OSPs) have developed their own set of technologies for big data analytics to resolve their own problems, such as online search (e.g., Google), online transaction processing (OLTP) (e.g., Amazon), and social network applications (e.g., Facebook and Twitter). For traditional industry, many companies and institutes are starting or planning to collect various data based on their business activities.

The initial goals of big data processing and analytics (BDPA) were to handle the 5V's of big data (Sagiroglu and Sinanc, 2013): volume, velocity, variety, veracity and value. Recently, with the development of big data technologies and solutions, both the academia and industry have reached a consensus, that is, the ultimate goal of big data is about transforming 'big data' to 'real value'. In this article, we discuss how to achieve this goal by proposing five-layer architecture for BDPA. The architecture is driven by the increasing volumes of data, especially the online big data streams, and by the demands of analysing data for all types of industries. The proposed architecture extends (Zhu et al., 2016) in three aspects. First, we have added a collection layer at the bottom of the original architecture to illustrate the importance of data quality. BDPA require us to extract information from multiple, diversified data sources, such as text, GPS, image, video, etc. with various qualities. Second, since the development of big data solutions is progressing fast, we have updated the case studies of big data solutions to the current solutions. Third, we believe that the choice of solutions in each layer is ultimately determined by the demands of applications. Therefore, unlike in Zhu et al. (2016), where we integrated online and historical data processing together in one layer, in this manuscript we have moved the stream processing as one function of the analytical layer, to satisfy the different requirements of analytics (such as, to discover insights offline, or on the fly).

The online big data streams discussed in this article share the following characteristics:

- *Large volume* – data are generated, transmitted and accumulated in large volume.
- *Continuous arrival* – real-time data arrive continuously.
- *Multiple sources* – data streams are from multiple distributed sources, and may be structured and unstructured.
- *Time-varying* – the size, format, and arrival rate of data streams change over time.
- *Unbounded streaming* – data come in unbounded amounts, requiring unbounded memory for processing and analytics.

To handle online big data streams, as well as the accumulated historical data, the overall BDPA solution should coordinate the following two types of functionalities:

- *Processing* – to handle large historical data repository with parallel processing techniques, such as MapReduce (MR) (Dean and Ghemawat, 2008), the dominant batch programming model. In practice, the processing of big data should consider not only the data volume, but also the data diversity, indexing efficiency, and the requirements from the specific application.
- *Analytics* – to discover knowledge and insights from data, by deploying specific models and algorithms based on the characteristics of the data. For example, for on-line big data stream, a streaming tool, such as Spark streaming (https://github.com/amplab-extras/SparkR-pkg) may be used to detect useful information, events and cross-correlation via processing online big data streams in real-time. For graphical data analysis, the GraphX (http://spark.apache.org/mllib/) library may be used.

The *analytics* part of BDPA solution is to discover insights and make decisions based on data. Thus, it could be built on top of *processing*. We believe that the two functionalities, which handle both online and historical data with scalability and reliability, are important for real-world BDPA based on two reasons:

1 The online data streams account for an increasing proportion of big data analytics. For example, in 2012, Google received over 2 million search queries per minute (http://www.internetonlinestats.com/google-search-statistics/). The number doubles to over 4 million per minute in the year 2014 (https://web-assets.domo.com/blog/wp-content/uploads/2014/04/DataNeverSleeps_2.0_v2.jpg). People and brands on Twitter sent more than 200 million tweets per minute in 2012 (http://www.internetonlinestats.com/twitter-statistics/), and more than 400 million by the end of 2015 (http://www.internetonlinestats.com/

twitter-statistics/). By the year 2020, over 50 billion devices are expected to be connected to the internet (Gerhardt et al., 2012), greatly advancing data-driven technologies for the 'internet of things' while requiring real-time big data analytics solutions to meet the users' demands.

2 Existing big data analytics technologies, such as IBM Infosphere (Biem et al., 2010), Twitter Storm (https://github.com/nathanmarz/storm/wiki/Tutorial), Apache S4 (Neumeyer et al., 2010), are de-signed to process and analyse streaming data, which are massive, heterogeneous, time-varying and unbounded. These technologies are realised without a storage system. The de-sign goals are mostly to guarantee quick event detection and real-time processing for learning tasks. However, to discover insight and value from the steaming data, one usually needs to combine information extracted from historical data.

Currently, there is no widely accepted BDPA solution, especially a general purpose solution fit for both traditional and internet industries. Hence in this paper, we propose architecture for the current BDPA solutions, which may serve as a de facto standard to help achieve value from data. We organise our article as follows. First we come up with the principles and desired characteristics of architecture for BDPA. Then we introduce five-layer architecture, including the functionalities and challenges for each layer. It is beyond the scope of this paper to discuss the details of existing technologies in each layer, so we only list some examples to discuss the functionalities of each layer and briefly focus on the motivations, merits and demerits of the corresponding techniques. In the end, we discuss some currently well-used BDPA solutions and conclude with the requirements for future BDPA.

2 Principles and desired characteristics for BDPA architecture

Driven by different types of demand on data analytics, such as time complexity, space complexity and quality of service (QoS), and inspired by the existing layer-based big data analytics stack (Franklin, 2013), we propose to design a layering architecture for BDPA. Layering architecture here does not refer to a real layered stack as in Franklin (2013) or as an enterprise cloud system such as Windows Azure (https://azure.microsoft.com/en-us/) and Amazon EC2 (https://aws.amazon.com/ec2/), but a structuring technique (Zimmermann, 1980) which comprises a succession of layers for the general purpose of allowing different big data analytics technologies in each layer to communicate and cooperate with technologies in the adjacent layers. For data driven analytics, where the inputs are online and historical big data, designing a layering architecture needs to consider the following principles and desired characteristics:

1 The architecture needs to collect structured, unstructured, multi-source, and cleaned-up data, generated by multiple data sources. These data come in various formats, such as, text, image, trajectory in the spatiotemporal space, etc. A collection layer is needed at the bottom, to prepare adequate and high-quality data for all kinds of analytical demands in the upper layers.

2 The architecture needs basic, extensible storage infrastructure for historical data storage and migrated data from the online data stream. Thus the storage layer, serving as the second lowest layer, is needed to guarantee I/O performance and scalability.

3 Since the massive, continuously arriving, heterogeneous, time-varying and unbounded online big data stream brings uncertainty to BDPA, an online and historical data processing layer is needed to handle both online inflow and batch files fetched from the storage layer for the analytics task. Usually this layer is based on a control or scheduling module to balance the two paradigms of computation, built on top of the storage layer. This layer aims to provide a cooperative and scalable distributed programming framework for the vast data streams and the accumulated historical big data, and should accommodate the various goals of analytics (e.g., prediction, recommendation, causal inference, recognition, clustering, etc.).

4 An analytics layer is required to perform data mining, prediction, and user-customised tasks. This layer needs to be designed to be value-centric, to turn 'big data' into 'big value'.

5 An application layer is required to serve as a flexible layer for the whole architecture, realising the demands from different industries and presenting the system status and results to data scientists where the experts' opinions are needed and giving feedbacks to the lower layers.

6 The input of the system consists of two types of data, i.e., streaming data and batch data. The online data streams flow into the online and historical data processing layer, while the historical data are imported and stored in the storage system. Thus the online and historical data processing layer needs to handle different processing tasks with different computing resource allocation and data migration strategies.

7 The output of the architecture, i.e., the analytics results for value extraction and knowledge discovery, is from the analytics layer. For example, analytics results acquired at the output can be used for contextualisation or visualisation.

8 The architecture solves the analytics problem in general, rather than for specific problems.

3 Five-layer architecture

To satisfy the principles and desired characteristics for the layering architecture, we come up with the following five-layer architecture for BDPA, as shown in Figure 1. The layers are collection layer, storage layer, processing layer, analytics layer, and application layer, from the bottom to the top. This section will introduce the functionalities, example case studies, as well as challenges for each layer.

3.1 Collection layer

The data collection layer serves as the foundation for the entire BDPA solution. With increasingly more data sources, such as smartphone data, networking data, sensor data, social media data, health data, etc. a collection layer is needed to integrate multi-source, structured and unstructured data for further management. The streaming data will be fed into the processing layer, and the accumulated historical data will be stored in the storage layer, in order to be further analysed with specific analytical tools in the analytical layer, based on the demands from the application layer. Solutions in this layer include both open data sources and third-party data providers.

- *Open data*: these data are released to the public by governments, organisations, research labs, or companies free of charge, to advance the development of BDPA solutions. Most of these data are not connected with individuals or are anonymous before publication.
- *Third-party data providers*: these data are collected through professional data collection agencies, such as Data-tang (http://factory.datatang.com/) and Analysyschina (http://www.analysyschina.com/about.html), via crawlers, commercial arrangement, crowd-sourcing, etc. The data are further customised to the users' requirements, providing data as a service (DAAS). Typically, these providers conduct preliminary cleansing and annotation before releasing these data, thus offering better data quality.

Challenges in this layer include diversity, privacy, labelling, and fault-tolerance.

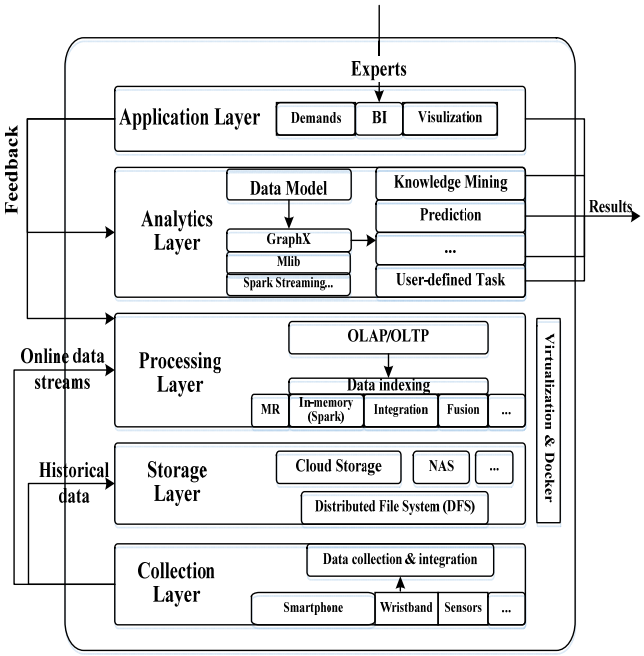
1 In practice, the collected data come in different formats, such as GPS, text, image, video. This layer is required to handle multi-source data efficiently, extract information from unstructured data, and integrate diversified data sources.

2 Data privacy is required to protect the personal information of users, especially when dealing with sensitive data, such as users' locations, transactions, etc.

3 Required by the analytical tasks of different applications, labelled data is appreciated for discovering unknown knowledge with higher precision. For example, for data analysis based on deep learning, some datasets without enough labels may not perform as well as expected. Labelling requires automatic and reliable annotation during data collection.

4 Fault-tolerance is required in the collection layer, to minimise the cases of missing values, or errors caused by the system, networking, etc.

Figure 1 Five-layer architecture for BDPA



3.2 Storage layer

The storage layer serves as the second layer and provides basic, distributed extensible storage infrastructure for historical data storage.

BDPA architecture requires storage technologies which have the following characteristics: scalable, with tiered storage, self-managing, highly available, widely accessible, and recoverable. In current big data storage market, big data storage architectures can be classified in Table 1. Obviously, they are not mutually exclusive, so some companies may use more than one.

Case studies

- *Distributed nodes architecture*: the most popular solution is Hadoop distributed file system (HDFS) (Borthakur, 2008). In most cases, HDFS works together with MR to execute Hadoop jobs. The obvious application for that storage architecture is many small files. Generally, it is the lowest cost commodity solution. However, it has relatively slow performance and is hard to manage.

- *Scale-out NAS*: it has the ability to scale throughput and capacity in tandem or independently. The most famous solution is Azure cloud storage (Calder et al., 2011). It can accommodate tens of petabytes of data, sufficient for many commercial applications. However, it is not suitable for pre-processing intensive applications (e.g., data format convertion, ingestion of data). Its greatest strength may be for large file processing.
- *All-SSD arrays*: Nutanix (https://www.nutanix.com/) and a number of other flash memory storage start-ups are offering pricey solid-state drive (SSD) arrays storage. These kinds of storage architecture yield information several times faster than more traditional architectures (i.e., scale-out NAS and distributed nodes). It is tailored for real-time processing where I/O is the performance bottleneck.
- *Object-based storage*: it is used for diverse purposes such as storing photos on Facebook, songs on Spotify, or files in online collaboration services, such as Dropbox. It has significant benefits for big data analytical systems, as it is highly scalable and it uses replication and distributed hash tables rather than redundant array of inexpensive disks (RAID) to ensure recoverability. It also supports peer-to-peer file sharing.

Table 1 Big data storage technologies

Technologies	*Description*	*Best use cases*
Distributed nodes	• Low-cost commodity hard-ware • Scale in tandem with the compute environment	Hadoop, Small distributed files
Scale-out network-attached storage (NAS)	• Automated storage tiering • Capable of scaling through-put and capacity in tandem or separately • Distributed or clustered file system	Large file processing, more traditional extraction or transformation of big data
All-solid-state-drive (SDD) arrays	• Several times faster than traditional storage technologies • Implemented like JBOD and distributed nodes	Real-time processing applications
Object-based storage	• Store data in flexible containers, not blocks • Use hash tables and replication • Allow peer-to-peer sharing across distributed nodes	Organisations willing to find a really competitive advantage

Challenges in the storage layer include locality, scalability, data migration, intelligence and independence, detailed below:

1 Algorithms are required to enhance high computation locality, and to be correlation-aware for distributed computation.

2 Error-correction mechanisms are required for data recovery and management of heterogeneous sources of data.

3 Data migration is required when queries arrive, and the streaming data, instead of being discarded, should be selected, grouped, and appended to the existing storage system.

4 The storage layer should be relatively independent and capable of integrating enterprise-level infrastructure.

5 Intelligence is required for query optimisation with distributed storage systems.

3.3 Processing layer

The data processing layer is the core in BDPA systems. It is a fundamental layer of many data analytical tasks. In the big data era, data sources are extremely heterogeneous in their structure and content. These different data sources are of widely differing data qualities (e.g., coverage, accuracy). Generally, the data processing layer performs parallel computing, data cleansing, data integration, data fusion, data indexing, virtualisation, and so on.

- *Parallel computing*: for the parallel processing technologies, two well-known examples are MR and SQL server translator. MR provides an innovative and scalable parallel programming model by allocating computation resources to distributed nodes. The basic idea is to divide the processing phase into two: map and reduce. In the map phase data is filtered and sorted into intermediate key/value pairs, then the reduce phase merges the intermediate values belonging to the same key. In practice, MR has been derived from Google's papers to an open-sourced framework for large-scale concurrent processing, Hadoop, comprised of the HDFS and the Hadoop MR. Hadoop stands out in terms of integrity, high availability, scalability and elasticity. However, it performs relatively poorly in stream processing compared to batch processing. When data come in continuous streams, the input files (in the format of text, key-value, binary sequence) from HDFS have to be segmented in small granules, causing significant delay. Furthermore, the 'reduce' phase in Hadoop only launches after all the 'map' jobs finish, which prolongs the waiting time. To improve the time efficiency of Hadoop jobs, improvement works have been proposed including pipelining the MR (Condie et al., 2010; Borthakur et al., 2011), adding interactive awareness and caching mechanisms such as Haloop

(Bu et al., 2010) and Pregel (Malewicz et al., 2010), scheduling by considering data popularity (Ananthanarayanan et al., 2011), and in-memory computing [Spark (Zaharia et al., 2010)]. To enrich the operations based on MR, OSPs and organisations come up with SQL-translators built on top of MR, as warehouses and middleware, such as Hive (Thusoo et al., 2009), SCOPE (Chaiken et al., 2008), Pig (Olston et al., 2008), etc. The SQL-translators could execute MR tasks with developer-friendly languages, for example, 95% Hadoop jobs at Facebook are generated by Hive (Lee et al., 2011). Since SQL-translators are based on MR, they are still not suitable for stream processing. In practice, the choices of processing solutions are normally determined by the specific demands of applications or the characteristics of data. Therefore, we only consider MR as the core function of the processing layer, and move the SQL-translator and in-memory processing (SPARK) to the analytics layer, to make the BDPA solution more flexible to the demands of the applications.

- *Big data cleansing*: many data cleansing systems have been proposed in both academic and industry (Herzog et al., 2007), e.g., Big-dansing (Khayyat et al., 2015), Cleanix (Wang et al., 2016). In the big data era, the quality of data cleansing is measured by the following metrics:
  1 Scalability – a good data cleansing system should scale out to thousands of machines in a shared nothing manner.
  2 Usability – it should provide a simple and friendly user interface for both expert and non-expert user.
  3 Abstraction – measures the flexibility of data cleansing system. For instance, users also need the flexibility to define rules to correct typos or errors in the dataset. However, none of the existing data cleansing systems dominate others in the market.
- *Big data integration*: it is more challenging than traditional data integration due to volume, velocity, variety and veracity. Researchers from Google and AT&T have pro-posed several big data integration systems to overcome the challenges of big data integration. For example, Solomon (Berti-Equille et al., 2009) was proposed to detect copying among different data sources. Alexander (Rekatsinas et al., 2015) helps administrators select the sources to balance the quality and the cost of integration. Time machine (Althoff et al., 2015) generates a timeline of events and relations for entities by integrating, and cleansing temporal information in the internet.
- *Big data fusion*: data fusion links data of diverse types, from heterogeneous data sources, in support of unified data query, search and analysis. It is hard as it needs to consider the semantics of the data set. We briefly introduce zoom data fusion system (http://www.zoomdata.com/). It makes multiple data sources appear as one source without moving data. It provides user friendly operations (e.g., drag and drop) and a powerful computation system to conduct data fusion. It also allows users to enrich big data with lookup information from relational database management systems, and with available metric functions to handle unstructured data sources, from data warehouses or other sources. Zoom data is also capable of correlating real-time data feeds with historical data.
- *Big data indexing*: indexing was proposed to speed up data query and data analysis in traditional OLTP and online analytical processing (OLAP). In general, indexes are a list of tags, names, subjects, etc. of a group of items which references where the data can be found (Adamu et al., 2016). However, existing indexing techniques (e.g., R-tree, B-tree) may not be suitable for big data analytics, e.g., the big data sets always have many columns. The big data indexing requirements are:
  1 Speed – it could search over billions, even trillions of data values in seconds.
  2 Multi-variable – it should be efficient for combining results from individual variable search results.
  3 Volume – the index size must be a fraction of the original data.
  4 Parallelism – the big data index should be easily partition into pieces for parallel and distributed processing.
  5 Speed of index generation – for in situ processing, index should be built at the rate of data generation.
- *Virtualisation*: virtualisation (Xing and Zhan, 2012) is a well-used technique in cloud computing regarding to both the storage and processing layer. It creates virtual (rather than physical) versions of computer hardware platforms, operating system, storage, etc. for provisioning, thus providing flexibility and scalability with regard to the computing resources. An emerging technology similar to virtualisation is Docker (Merkel, 2014), which is a container technology for Linux that allows a developer to package up an application with all of the parts it needs. By using containers, resources can be isolated, services restricted, and processes provisioned to have an almost completely private view of the operating system with their own process ID space, file system structure, and network interfaces. Multiple containers share the same kernel, but each container can be constrained to only use a defined amount of resources such as CPU, memory and I/O.

The processing layer faces such challenges as scalability, autonomic provisioning, low latency requirements, hot plugging, flexibility, prediction capability and fault tolerance mechanism, described in detail below:

1 Techniques in this layer should be scalable according to the computation demands.

2 Adaptive hardware resource allocation should be considered, automatically determining the scale of the Hadoop cluster to optimise processing.

3 MR tasks need to be added, reduced, launched, or suspended in real-time to ensure low latency of the processing system.

4 Subsystems in this layer are expected to be hot pluggable in order to be flexibly assembled, to achieve optimum performance.

5 The system requires a balance between stream processing and batch processing.

6 Technologies in this layer should be capable of predicting future arriving data stream volume and properly con-figuring the system for optimised performance in advance.

7 Moreover, packet losses and failures in the incoming data stream must be properly handled.

3.4 Analytics layer

BDPA systems require a mix of data analytics tools for different user requirements. In response to this trend, a number of academic (Malewicz et al., 2010; Zaharia et al., 2012) and commercial systems (Toshniwal et al. 2014) have been developed to support such use cases. The earliest data analytics systems, such as MR, gave users a powerful, but less abstract programming interface. Apache Spark is a fast and general engine for large-scale data processing and analytics. It runs programs up to 100x faster than Hadoop MR in memory. It is quite easy to use. Spark powers a set of libraries, e.g., machine learning, and streaming, rendering data analytics much easier than other systems. We overview these data analytics libraries as follows:

- *Apache spark SQL (Armbrust et al., 2015)*: it is a module for working with structured data. It includes a cost-based optimiser, columnar storage and code generation to make SQL queries faster. It also scales to thousands of nodes and multi-hour querying using Spark engine. Spark SQL can perform relation operations (e.g., join) on both internal data sources [i.e., Spark's resilient distributed datasets (RDDs)] and external data sources (e.g., datasets in HDFS). In order to support many heterogeneous data sources and the wide range of big data algorithms, Spark SQL includes a highly extensible optimiser catalyst.
- *Apache spark streaming (http://spark.apache.org/streaming/)*: many modern data processing and analysis environments require complex computation on steaming data in real-time. For example, a Twitter user requires making a number of complex decisions, often based on the data that has just been created. Many stream data processing and analytics have been proposed to handle this kind of processing and analysis requirements, for example: storm (Toshniwal et al., 2014), and Spark streaming (http://spark.apache.org/streaming/). Here we briefly introduce the apache sparking streaming project. Spark streaming is a large-scale near-real-time stream processing system. It can scale to hundreds of nodes. The processing latencies are in seconds. Spark streaming also integrates apache spark's batch and interactive processing API, thus it provides many benefits for implementing complex algorithms. It reads data from many data sources: Flume, Kafka, Twitter or traditional data storage systems (NAS, HDFS). In addition, one can also customise the data sources.
- *Apache spark MLlib (Meng et al., 2016)*: machine learning is one core model in BDPA systems. It can find patterns and make predictions from data based on work in statistics, data mining, pattern recognition and predictive analytics. Apache spark MLlib is the largest distributed ma-chine learning library, which tightly integrates with Spark. It provides fast and scalable implementations of standard learning algorithms for common learning settings including classification, regression, collaborative filtering, clustering, and dimensionality reduction. It benefits from data-parallelism or model-parallelism to store and operate on data or models in spark.
- *Apache spark GraphX (Gonzalez et al., 2014)*: a number of graph processing and analytics systems have been proposed [Pregel (Malewicz et al., 2010), PowerGraph (Gonzalez et al., 2012)] to meet the analysis requirements, e.g., PageRank, community detection in big data applications. However, most of these graph processing systems abandon fault tolerance in pursuit of system performance. Apache spark GraphX is built on top of apache spark. Its programming abstraction extends the spark data flow operators by introduction several customised graph operators (e.g., subgraph, vertices, leftjoinV, reverse).

There are many other data analytics libraries or sub-models have been developed for apache spark. For example, GeoSpark (https://github.com/DataSystemsLab/GeoSpark) is a geospatial library for efficient spatial and temporal system. SparkR (https://github.com/amplab-extras/SparkR-pkg) is an R frontend for Spark. We omit the detailed discussion of these projects and refer the interested reader to the apache spark project (http://spark.apache.org/).

Challenges for the analytics layer include multiplicity, analytical efficiency, adaptivity, pluggability, and complexity:

1 Since current big data stream analytics concentrate on three aspects, namely, artificial intelligence, knowledge discovery and prediction tasks driven by various services, the analytics layer should support both commonly used and user-defined analytics applications.

2 Immediate prediction in spite of the potentially unbounded streaming is required.

3 Adaptive data mining models are expected for structured, unstructured, multi-sourced, time-varying big data streams.

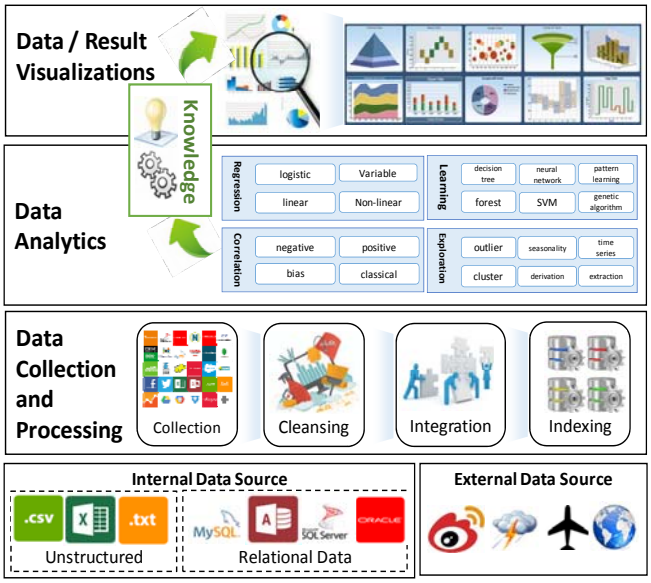
4 These analytics applications should be pluggable for developers to operate in open-source environments with multiple upper-level languages.

5 In addition, it is desired to handle increasingly complex statistical models for seeing both the forest and the trees on the run, by analysing the overall datasets instead of the samples or the data fragmentations.

3.5 Application layer

The application layer acts as the highest layer. With the previous four layers, BDPA systems can build various applications for different users. For example, business intelligence, stock analysis and prediction. The outputs of these applications help users to make fast decisions.

Figure 2 InfidataCA – city commercial areas analysis system (see online version for colours)



For example, Infidata (http://www.infidata.cn/) is a start-up company, which focuses on helping organisations to build big data processing applications. One of its commercial products, Shenzhen's commercial areas analysis system (InfidataCA in Figure 2), is a real commercial system which matches our five-layer big data processing and analysis architecture. IndidataCA was developed to analyse the commercial areas in Shenzhen, China. In data collection layer, it collects external data (i.e., Weibo) and the basic geographical data of that city. The internal data sources are the traffic data (e.g., real-time bus, taxi and metro transit) in Shenzhen. In data storage layer, it stores the data from collection layer in distributed nodes architecture (i.e., HDFS). In data processing layer, it cleans and integrates these heterogeneous data sources (e.g., original and destination mapping in taxi trajectories). In data analytics layer, it employs GeoSpark (https://github.com/DataSystemsLab/GeoSpark) to analyse the trajectories of taxi, buses to identify the abnormal cases in the traffic systems. InfidataCA uses spark streaming (http://spark.apache.org/streaming/) to process these real-time trajectories and detect the traffic jams in the city. The spark MLlib (http://spark.apache.org/mllib/) was used to conduct further pre-diction and mining. In data visualisation layer, InfidataCA visualises these analysis result for end-users and helps them to make fast decisions. For example, InfidataCA reports the real-time pedestrian numbers in these commercial areas, and this information will assist the bus company to schedule their buses.

The challenges for the application layer are in the overall model and technology selection, and include adaptivity, value, and generalisability issues:

1 The application layer is supposed to decide the acceptance or rejection of different analytics models and algorithms, as well as different resource allocation strategies for the storage layer and processing layer. This layer should be capable of adaptively making the above-mentioned adjustment.

2 Extraction of 'value' based on the business logic is expected.

3 The whole solution should be universally applicable to multiple fields and industries.

4 Discussion of well-used big data solutions

4.1 Layer techniques bench-marking

The above mentioned five-layers constitute a bottom-up structure with different priorities, supporting BDPA by three mechanisms. First, one layer provides computation and API to the next higher-level layer, while the decision-making functionality in the highest-level layer (application layer) controls the whole system based on value-oriented decisions. Second, one layer has the ability to schedule and optimise resource allocations for its own and all the lower-level layers, thus provisioning adaptive processing and analytics resources according to application demands. Third, the choices of tools or products (such as spark streaming, or spark SQL) within each layer, is commonly determined by the upper layers, such as, what the application targets at, or what the characteristics of the data are.

Regarding the techniques used in the five-layers, there have been quite a number of existing solutions targeting at one specific layer or multiple layers. We first give a comparative study on the performances of solutions within the storage, processing, and analytics layers. Then, for those vertical solutions covering all layers, we review these solutions and summarise the corresponding challenges.

Table 2 Comparison in the performances of different solutions in the storage, processing, and analytics layers

	Solutions		*Performance*		*Benchmark*	*Ref*
Storage	Nosql/HDFS vs. SQL		Nosql/HDFS → (Millions of transactions/day) SQL → (Thousands of transactions/day)		YCSB, BDB, TPC	https://github.com/benstopford/awesome-db-benchmarks
Processing	Hadoop in physical vs. virtual environment		Physical (significantly >) virtual		Pi, TestDFSIO, TeraSort.	https://www.cse.wustl.edu/~jain/cse570-13/ftp/bigdatap.pdf
Analytics	Spark vs. Hadoop	a	2.63–0.41x speed up	a	Pagerank	http://udspace.udel.edu/bitstream/handle/19716/17628/2015_LiuLu_MS.pdf?sequence=1
		b	2.76–1.53x speed up	b	WordCount	
		c	0.85–1.21x speed up	c	RunningSort	
		d	0.97–1.60x speed up	d	TeyraSort	
		e	13.83–89.01x speed up	e	Naïve-Bayes	
		f	4.06–2.20x speed up	f	K-means	
	Spark vs. Flink vs. Kafka		Spark can reach 5x or higher throughput over other popular streaming systems.		Databricks-yahoo streaming benchmark	https://databricks.com/blog/2017/10/11/benchmarking-structured-streaming-on-databricks-runtime-against-state-of-the-art-streaming-systems.html

Table 3 Challenges faced by popular vertical solutions in each layer

Layers and challenges	*1 collection layer*				*2 storage layer*				*3 processing layer*			
Commercial and open-source solutions	*1.1 Diversity*	*1.2 Privacy*	*1.3 Labelling*	*1.4 Fault-tolerance*	*2.1 Locality and scalability*	*2.2 Data migration*	*2.3 Independence*	*2.4 Intelligence*	*3.1 Autonomic provisioning*	*3.2 Scalability*	*3.3 Low latency*	*3.4 Hot plugging*
Azure				√	√	√	√	√	√	√	√	
AWS				√	√	√	√	√	√	√	√	
Hortonworks		√			√	√	√	√		√	√	√
InfidataCA	√			√								
BDAS					√	√	√	√		√	√	√

Layers and challenges	*4 analytics layer*				*5 application layer*			
Commercial and open-source solutions	*4.1 Multiplicity*	*4.2 Analytical efficiency*	*4.3 Streaming*	*4.4 Complexity capability*	*5.1 Model selection*	*5.2 Adaptivity*	*5.3 Value*	*5.4 Generalizability*
Azure	√	√	√	√	√	√	√	
AWS	√	√	√	√	√	√	√	
Hortonworks	√	√	√	√		√	√	
InfidataCA	√	√	√	√				
BDAS	√	√	√	√		√	√	

Note: √ symbol indicates the challenges each solution will address.

The storage, processing, and analytics layers in the five-layer architecture are the core layers for demonstrating the efficiency and scalability of big data solutions. Therefore, we survey the corresponding solutions in the three layers, with comparative statistics on their performances, as shown in Table 2.

In the storage layer, benchmark tests (https://github.com/benstopford/awesome-db-benchmarks) are

conducted in both Nosql/HDFS based storage systems, and traditional relational (SQL) databases. Basically, if the users generate thousands of transactions per day, SQL database will have no problem in terms of the response time. However, when the users generate more than a million transactions per day, SQL database will not be able to cope, and Nosql/HDFS storage systems would be a better choice to guarantee the throughput. In the processing layer, the comparison of benchmarks (https://www.cse.wustl.edu/~jain/cse570-13/ftp/bigdatap.pdf) for Hadoop in physical and virtual environment show that Hadoop in physical environment significantly outperforms Hadoop in virtual environment. This indicates that the processing layer takes advantage of the physical computational resources rather than just schedules the computational resources. In the analytics layer, statistics show that (http://udspace.udel.edu/bitstream/handle/19716/17628/2015_LiuLu_MS.pdf?sequence=1) spark outperforms Hadoop in terms of complex computations, such as Naïve Bayes, but shares similar performances for basic operations, such as TeraSort and RunningSort. For analysing streaming data, Spark streamlining can reach 5X or higher throughput over Flink and Kafka (https://databricks.com/blog/2017/10/11/benchmarking-structured-streaming-on-databricks-runtime-against-state-of-the-art-streaming-systems.html). Recently, spark-based solutions have obtained more attention in the big data analytics architectures.

4.2 Challenges faced by BDPA solutions

Up to now the academia and industry have reached a consensus that the ultimate goal for big data solutions is to obtain 'value'. To better understand this goal, we list the challenges of the BDPA solutions regarding to each layer in Table 3. We also compare five vertical big data solutions [i.e., AWS (https://aws.amazon.com/ec2/), Azure (https://azure.microsoft.com/en-us/), Hortonworks (https://hortonworks.com), InfidataCA (http://www.infidata.cn/), and BDAS (Franklin, 2013)] as to which challenges each will address.

There are many vertical solutions provided by established companies, either as big data on top of cloud platforms, or as typical big data platforms. AWS (https://aws.amazon.com/ec2/) and Azure (https://azure.microsoft.com/en-us/) provide big data solutions based on their cloud platforms, taking advantages of both the scalable storage system and pay-as-you-go computation resources. The big data solutions of Azure and AWS are usually proposed as [data as a service (Daas)]', divided either by applications (mobile services, websites, etc.) or by industries (finance, health, etc.). The cloud platform based big data solutions focus more on the challenges of both the infrastructure and the analytics models, i.e., the storage, processing, and analytics layer. Hortonworks (https://hortonworks.com) and InfidataCA (http://www.infidata.cn/) provide two other perspectives on big data platforms. Hortonworks' big data platform is composed primarily by Hadoop 2.0 and HDFS. Hadoop 2.0 provides the resource management and pluggable architecture which enable a wide variety of data access methods, allowing multiple applications to run on the same platform. These big data platforms focus more on the real cases rather than general application functionalities. The big data solutions from Hortonworks are presented either by use cases or by industries.

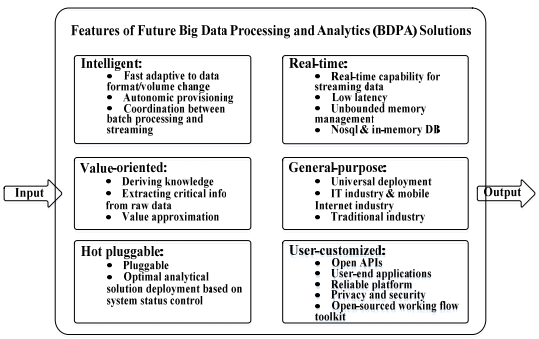
Start-up companies tend to focus on the data analytics part, and avoid high maintenance cost. They are more focused on the data quality, getting reliable labels, and intelligent analysis. For open-source big data platforms, the Berkeley data analytics stack (BDAS) stands out as a well-used software stack (Franklin, 2013), together with many other innovative components in BDAS, such as spark streaming (https://github.com/amplab-extras/SparkR-pkg) and GraphX (http://spark.apache.org/mllib/). BDAS is built by the AMPLab, from academia, and it focuses more on the advanced technologies to tackle the challenges at the analytics layer.

In practice, more user-friendly BDPA solutions are expected to incorporate flexible sets of layers. It is noted from Table 2 that for the challenges of privacy (1.2), labelling (1.3), fault-tolerance (1.4), hot-plugging (3.4), and generalisability (5.4), the discussed solution could not satisfy the requirements quite well, indicating there could be more opportunities to improve the data quality at the collection layer, improve the efficiency of data processing from the system level, and finally make the BDPA solution more customised to different applications.

5 Conclusions

We are entering the data era, where BDPA solutions allow the possibilities of getting "value" in both internet and traditional industries. Future solutions should possess such desirable features as being intelligent, real-time, value-oriented, general-purpose, dismountable, and user-customised, as shown in Figure 3. Realising such features are challenging. Solutions on realising the functionalities in five-layers, designing elements in these layers, and scheduling middleware layer, need to be proposed for the future BDPA architecture, which will be an ecosystem built on the five-layers we present, with solutions providing reliable and efficient interfaces to interact with other solutions and satisfy the demands of real-world applications.

Figure 3 Features of future BDLA solutions



Acknowledgements

Bo Tang was supported by the Science and Technology Innovation Committee Foundation of Shenzhen (Grant No. ZDSYS201703031748284)

References

Adamu, F.B. et al. (2016) 'A survey on big data indexing strategies', *SLAC National Accelerator Laboratory (SLAC)*, No. SLAC-PUB-16460.

Althoff, T. et al. (2015) 'TimeMachine: timeline generation for knowledge-base entities', *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM.

Ananthanarayanan, G. et al. (2011) 'Scarlett: coping with skewed content popularity in MapReduce clusters', *Proceedings of the Sixth Conference on Computer Systems*, ACM.

Armbrust, M. et al. (2015) 'Spark SQL: relational data processing in spark', *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, ACM.

Berti-Equille, L. et al. (2009) 'Sailing the information ocean with awareness of currents: discovery and application of source dependence', *CIDR*.

Biem, A. et al. (2010) 'IBM infosphere streams for scalable, real-time, intelligent transportation services', *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, ACM.

Borthakur, D. (2008) *HDFS Architecture Guide, Hadoop Apache Project* [online] https://hadoop.apache.org/docs/r1.2.1/hdfs_design.pdf. (accessed 2018)

Borthakur, D. et al. (2011) 'Apache Hadoop goes real-time at Facebook', *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*, ACM.

Bu, Y. et al. (2010) 'HaLoop: efficient iterative data processing on large clusters', *Proceedings of the VLDB Endowment*, Vols. 3.1–3.2, pp.285–296.

Calder, B., Wang, J., Ogus, A., Nilakantan, N., Skjolsvold, A., McKelvie, S., Xu, Y. et al. (2011) 'Windows azure storage: a highly available cloud storage service with strong consistency', in *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, ACM, pp.143–157.

Chaiken, R. et al. (2008) 'SCOPE: easy and efficient parallel processing of massive datasets', *Proceedings of the VLDB Endowment*, Vol. 1.2, pp.1265–1276.

Condie, T. et al. (2010) 'MapReduce online', *NSDI*, Vol. 10, No. 4.

Dean, J. and Ghemawat, S. (2008) 'MapReduce: simplified data processing on large clusters', *Communications of the ACM*, Vol. 51.1, pp.107–113.

Franklin, M. (2013) 'The Berkeley data analytics stack: present and future', in *2013 IEEE International Conference on Big Data*, IEEE, pp.2–3.

Gantz, J. and Reinsel, D. (2012) 'The digital universe in 2020: big data, bigger digital shadows, and biggest growth in the far east', *IDC iView: IDC Analyze the Future*.

Gerhardt, B., Griffin, K. and Klemann, R. (2012) *Unlocking Value in the Fragmented World of Big Data Analytics*, Cisco Internet Business Solutions Group, June.

Gonzalez, J.E. et al. (2012) 'Powergraph: distributed graph-parallel computation on natural graphs', Presented as part of the *10th USENIX Symposium on Operating Systems Design and Implementation (OSDI 12)*.

Gonzalez, J.E. et al. (2014) 'Graphx: graph processing in a distributed dataflow framework', *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14)*.

Herzog, T.N. et al. (2007) *Data Quality and Record Linkage Techniques*, Springer Science and Business Media, New York.

Khayyat, Z. et al. (2015) 'Bigdansing: a system for big data cleansing', *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, ACM.

Lee, R. et al. (2011) 'Ysmart: yet another SQL-to-MapReduce translator', *2011 31st International Conference on Distributed Computing Systems (ICDCS)*, IEEE.

Malewicz, et al. G. (2010) 'Pregel: a system for large-scale graph processing', *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, ACM.

Malewicz, G. et al. (2010) 'Pregel: a system for large-scale graph processing', *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, ACM.

Meng, X. et al. (2016) 'Mllib: machine learning in apache spark', *JMLR*. Vol. 17, No. 34, pp.1–7.

Merkel, D. (2014) 'Docker: lightweight Linux containers for consistent development and deployment', *Linux Journal*, No. 239, p.2.

Neumeyer, L. et al. (2010) 'S4: distributed stream computing platform', *2010 IEEE International Conference on Data Mining Workshops (ICDMW)*, IEEE.

Olston, C. et al. (2008) 'Pig Latin: a not-so-foreign language for data processing', *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, ACM.

Rekatsinas, T. et al. (2015) 'Finding quality in quantity: the challenge of discovering valuable sources for integration', *CIDR*.

Sagiroglu, S. and Sinanc, D. (2013) 'Big data: a review', *IEEE International Conference on In Collaboration Technologies and Systems (CTS)*, May, pp.42–47.

Thusoo, A. et al. (2009) 'Hive: a warehousing solution over a MapReduce framework', *Proceedings of the VLDB Endowment*, Vol. 2.2, pp.1626–1629.

Toshniwal, A. et al. (2014) 'Storm@ twitter', *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, ACM.

Wang, H. et al. (2016) 'Cleanix: a parallel big data cleaning system', *ACM SIGMOD Record*, Vol. 44, No. 4, pp.35–40.

Xing, Y. and Zhan, Y. (2012) 'Virtualization and cloud computing', *Future Wireless Networks and Information Systems*, Springer Berlin Heidelberg, pp.305–312.

Zaharia, M. et al. (2012) 'Resilient distributed datasets: a fault-tolerant abstraction for in-memory cluster computing', *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*, USENIX Association.

Zaharia, M., Chowdhury, M., Franklin, M.J., Shenker, S. and Stoica, I. (2010) 'Spark: cluster computing with working sets', in *Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing*, Vol. 10, p.10.

Zhu, J.Y., Xu, J. and Li, V.O.K. (2016) 'A four-layer architecture for online and historical big data analytics', *14th Intl Conf on Pervasive Intelligence and Computing Dependable, Autonomic and Secure Computing*, *2nd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/ CyberSciTech)*, *2016 IEEE 14th Intl C*, IEEE.

Zimmermann, H. (1980) 'OSI reference model – the ISO model of architecture for open systems interconnection', *IEEE Transactions on Communications*, Vol. 28, No. 4, pp.425–432.

Websites

http://www.internetonlinestats.com/google-search-statistics/

http://www.internetonlinestats.com/twitter-statistics/

https://github.com/nathanmarz/storm/wiki/Tutorial

https://azure.microsoft.com/en-us/

https://aws.amazon.com/ec2/

https://www.nutanix.com/

http://www.zoomdata.com/

http://spark.apache.org/streaming/

https://github.com/benstopford/awesome-db-benchmarks

https://www.cse.wustl.edu/~jain/cse570-13/ftp/bigdatap.pdf

http://udspace.udel.edu/bitstream/handle/19716/17628/2015_LiuLu_MS.pdf?sequence=1

https://hortonworks.com

https://github.com/DataSystemsLab/GeoSpark

https://github.com/amplab-extras/SparkR-pkg

http://spark.apache.org/

http://www.infidata.cn/

http://spark.apache.org/mllib/

http://factory.datatang.com/

http://www.analysyschina.com/about.html

https://web-assets.domo.com/blog/wp-content/uploads/2014/04/DataNeverSleeps_2.0_v2.jpg

https://databricks.com/blog/2017/10/11/benchmarking-structured-streaming-on-databricks-runtime-against-state-of-the-art-streaming-systems.html