Chapter 3
Big Data Outlook, Tools, and Architectures

Hajira Jabeen

Smart Data Analytics
University of Bonn, Germany

Abstract. Big data is a persistent phenomena, the data is being generated and processed in a myriad of digitised scenarios. This chapter covers the history of and aims to provide an overview of the existing terms and enablers related to big data. Furthermore, the chapter covers prominent technologies, tools, and architectures developed to handle this large data at scale. At the end, the chapter reviews knowledge graphs that address the challenges (e.g. heterogeneity, interoperability, variety) of big data through their specialised representation. After reading this chapter, the reader can develop an understanding of the broad spectrum of big data ranging from important terms, challenges, handling technologies, and their connection with large scale knowledge graphs.

1 Introduction

The digital transformation has impacted almost all aspects of modern society. The past decade has seen tremendous advancements in the areas of automation, mobility, the internet, IoT, health, and similar areas. This growth has led to enormous data-generation facilities, and data-capturing capabilities. In the first section, we review the definitions and descriptions of big data and discuss the drivers behind big data generation, the characteristics exhibited by big data, the challenges offered by big data, and the handling of this data by creating data value chains. In the section, we cover the software solutions and architectures used to realise the big data value chains. We further cover characteristics and challenges relating to big data. The section connects knowledge graphs and big data, outlining the rationale and existing tools to handle large-scale knowledge graphs.

2 Big Data: Outlook

Today, big data is widespread across and beyond every aspect of everyday life. This trend of increasing data was first envisioned and defined years ago. Notably, the first evidence of the term big data comes from a paper [87] published in 1997, where the authors described the problem as *BigData* when the data do not fit in the main memory (core) of a computing system, or on the local disk. According to the Oxford English Dictionary (OED), big data is defined as: “data of a

very large size, typically to the extent that its manipulation and management present significant logistical challenges." Later, when the terms velocity, variety, and volume were associated as characteristics of big data, the newer definitions of the term 'big data' came to cover these characteristics, as listed below:

1. "Big data is high volume, high velocity, and/or high variety information assets that require new forms of processing to enable enhanced decision making, insight discovery and process optimization," [271, 297].
2. "When the size of the data itself becomes part of the problem and traditional techniques for working with data run out of steam," [288].
3. Big Data is "data whose size forces us to look beyond the tried-and-true methods that are prevalent at that time," [217].
4. "Big Data technologies are a new generation of technologies and architectures designed to extract value economically from very large volumes of a wide variety of data by enabling high-velocity capture, discovery, and/or analysis," [470].
5. "Big data is high-volume, high-velocity and high-variety information assets that demand cost-effective innovative forms of information processing for enhanced insight and decision making," [271].
6. "Big Data is a term encompassing the use of techniques to capture, process, analyse and visualize potentially large datasets in a reasonable timeframe not accessible to standard IT technologies." By extension, the platform, tools and software used for this purpose are collectively called "Big Data technologies," [98].
7. "Big data can mean big volume, big velocity, or big variety," [414].
8. "The term is used for a collection of datasets so large and complex that it becomes difficult to process using on-hand database management tools or traditional data processing applications" [1].
9. "Big data represents the information assets characterized by such a high volume, velocity and variety to require specific technology and analytical methods for its transformation into value" [2].

Regardless of the defining text, big data is a persistent phenomenon and is here to stay. We take a brief overview of the key enabling technologies that made big data possible in the following section.

2.1 Key Technologies and Business Drivers

As recently as the year 2000, digital information only constituted about one quarter of all the stored information worldwide [3]. Other information was mainly stored on paper, film, or other analogue media. Today, by contrast, less than two percent of all stored information is non-digital. The key enablers of the big

[1] http://en.wikipedia.org/
[2] http://en.wikipedia.org/
[3] https://www.foreignaffairs.com/articles/2013-04-03/rise-big-data

digital data revolution are the advancements in technologies, be it increased internet speed, the availability of low-cost handheld mobile devices, or the myriad of applications ranging from social media to personal banking. At present, organizations view the acquisition and possession of data as significant assets. A report by the World Economic Forum [315], "Big Data, Big Impact," declared 'data' an asset akin to currency or gold. This fact has led to significant changes in business models. Besides, aggressive acquisition and retention of data have become more popular among organizations. Prominent examples are internet companies such as Google, Yahoo, Amazon, or Facebook, which are driven by new business models. Technology proliferation is one of the major enablers of big data acquisition. Cheaper and accessible technology is being used in almost all parts of modern society, be it smart devices, mobile devices, wearable devices, or resources to store data on the cloud, enabling customers to make purchases and book vacations among other functions. In the following section, we will cover a few of the prominent big data enabling technologies and key drivers.

Internet The advancements in internet bandwidth and streaming have enabled fast and efficient data transfer between physically distant devices. People around the globe are accessing the internet via their mobile devices and the number of connected devices is constantly increasing. The number of internet users increased from 4.021 billion to 4.39 billion from 2018 to 2019. Almost 4.48 billion people were active internet users as of October 2019, encompassing 58 percent of the global population[83]. In the age of digital society, there is a need for a powerful wireless network that can rapidly transfer large volumes of data. Presently, we are moving from 4G LTE to 5G NR, which will enable entirely new applications and data-collection scenarios. 5G not only comes with better bandwidth and faster speed but also lower latency. The low latency of 5G was demonstrated by "Orchestrating the Orchestra"[4] – an event that enabled musicians across different locations to perform at the Bristol 5G Smart Tourism event. Violinist Anneka Sutcliffe was playing in Bristol, Professor Mischa Dohler was playing the piano in The Guildhall, London, and vocalist Noa Dohler and violinist Rita Fernandes were at Digital Catapult in Euston, London. These developments have made it possible to share and curate large amounts of data at high speeds.

Automation and Digitization Digital automation is a relatively broad term and it covers tasks that can be done automatically with minimal human assistance, increasing the speed and accuracy as a result. Businesses are more and more favouring the use of automatization tools to achieve more throughput. For example, advancements in automatic tools like scanning systems no longer require manual entry, easing and speeding up the process at the cost of more and reliable data capture. Similarly, in terms of digital data, cameras and photos are another example. The number of digital photos taken in 2017 was estimated to

[4] https://www.bristol.ac.uk/news/2019/march/orchestrating-the-orchestra.html

be 1.2 trillion[5], which is roughly 160 pictures for every one of the roughly 7.5 billion people inhabiting planet earth. With more and more devices being digitized, more data is being created and stored in machine-readable form. Industries and businesses must harness this data and use it to their advantage.

Commodity computing Commodity hardware, also known as off-the-shelf, non-proprietary hardware, refers to low-cost devices that are widely available. These devices can be easily replaced with a similar device, avoiding vendor lock-in challenges. 'Commodity cluster computing' is the preference of using more of average-performing, low-cost hardware to work in parallel (scalar computing), rather than having a few high-performance, high-cost items of hardware. Hence, commodity computing enables the use of a large number of already existing computing resources for parallel and cluster computing without needing to buy expensive supercomputers. Commodity computing is supported by the fact that software solutions can be used to build multiple points of redundancy in the cluster, making sure that the cluster remains functional in case of hardware failures. Low-cost cluster computing resources have made it possible to build proprietary data centres on-premises and to reap the benefits of in-house big data handling and processing.

Mobile computing Handheld smart devices are becoming more and more common due to increased affordability, relevance, and digital literacy. There were 5.11 billion unique mobile users in 2018 and 5.135 billion in 2019, accounting for 67% of the global population, and it is estimated [83] that by 2020, almost 75% of the global population will be connected by mobile. Use of mobile computing has enabled almost everyone to access and generate data, playing a key role in big data generation and sharing.

Mobile applications Mobile devices are playing a key role in the present data explosion. Mobile phones are no longer only being used for voice calls. Currently, 56% of web access worldwide is generated by mobile devices. At the moment, more than 57% of the global population use the internet and more than 52% of the global population use mobile devices [83]. Businesses are developing mobile applications to not only assist users in ubiquitous computing but also to generate and capture data of interest. The mobile application development industry is creating mobile apps for almost all fields, and existing mobile applications cover a range of tasks like online banking, online purchases, social interactions, travelling, eating, studying, or entertainment. All of these applications not only assist in automated data collection related to tasks (e.g. orders) but also assist in generating additional data that was not easily possible before (e.g. correlating a new order with previous purchases).

[5] https://www.statista.com/statistics/617136/digital-population-worldwide/

Ubiquitous devices (IoT) The internet of things (IoT) enables scenarios where network connectivity and computing capability extends to objects, sensors and everyday items not normally considered computers, allowing these devices to generate, exchange and consume data with minimal human intervention. These devices are not only increasing in number to cover different facets of life but are also increasing their sensitivity. According to a GSMA report [16], between 2018 and 2025, the number of global IoT connections will triple to 25 billion. In an estimate by CGI [2], the total volume of data generated by IoT will reach 600 ZB per year by 2020.

Cloud infrastructure Cloud computing is the term used for storing, accessing and processing data over the internet from a remote server. This ability to store and manipulate data on the cloud using services like AWS [50], Google Cloud Platform [264], Cloudera [378], etc. has made it possible to store and analyse data on-demand with a pay-per-use model. Cloud computing saves costs, offers better performance, reliability, unlimited capacity, and quick deployment possibilities. Cloud computing has assisted organizations with providing the data centre management and efficient data-handling facilities.

2.2 Characteristics of Big Data

Driven by digital transformation, big data is identified by several key attributes. Interestingly, they all start with the letter 'V', and therefore are also called the V's of big data. The number of characteristic attributes is constantly increasing with advancements in technologies and underlying business requirements. In this section, we cover a few of the main V's used to describe big data.

Three Vs of Big Data [489, 271]

1. Volume: The size of data is increasing at unprecedented rates. It includes data generated in all fields including science, education, business, technology and governance. If we take the social media giant Facebook (FB) as an example, it has been reported that FB generates approximately 4 petabytes of data in 24 hours with 100 million hours of video watch-time. FB users create 4 million likes per minute, and more than 250 billion photos have been uploaded to Facebook since its creation, which equates to 350 million photos per day. Apart from the applications, a vast amount of data is being generated by web, IoT and many other automation tools continuously. All of this data must be captured, stored, processed and displayed.
2. Velocity: The speed at which the data is being generated has increased rapidly over the years. The high rate and speed is contributed by the increase in the use of portable devices to allow data generation and ever-increasing bandwidth that allows fast data transfer. In addition, the rate of data generation (from the Internet of Things, social media, etc.) is increasing as well. Google, for example, receives over 63,000 searches per second on any given day. And 15% of all searches have never been searched before on Google.

Therefore, it is critical to manage and analyse the data at the same rate at which it is being generated and stored in the system.

3. Variety: The data comes from a variety of data sources and is generated in different forms. It can be structured or unstructured data. Data comes in the form of text (emails, tweets), logs, signals, records, photos, videos, etc. This data cannot be stored and queried via traditional structured database management systems. It is important to develop new solutions that are able to store and query diverse data; *4 Vs of Big Data* [106].
4. Veracity: This is the quality, truthfulness or reliability of data. Data might contain biases, noise, or abnormalities. It is important to be aware of whether data being used can be trusted for use in making important decisions, or if the data is meaningful to the problem being analysed. The data is to be used to make decisions that can bring strategic competitive advantages to the business; *10 Vs of Big Data* [60].
5. Variability: This is the dynamic and evolving nature of data. The data flow is not constant or consistent. The speed, density, structure, or format of data can change over time and several factors influence the consistency of data that changes the pattern, e.g. more shoppers near Christmas, more traffic in peak hours etc.
6. Value: This refers to the worth of the data being extracted. For an organization, it is important to understand the cost and associated benefits of collection and analysis of data. It is important to know that the data can be turned into value by analysis, and that it follows set standards of data quality, sparsity or relatedness.
7. Visualization is often thought of as the only way in which customers can interact with models. It is important to visualize the reports and results that can be communicated and extracted from data in order to understand underlying patterns and behaviours.

   In addition to the characteristics mentioned above, some researchers have gone as far as to introduce 42 [395], or even 51 [243] different Vs to characterise big data.

2.3 Challenges of Big data

The characteristics of data combined with targeted business goals pose plenty of challenges while dealing with big data. In this section, we briefly cover the main challenges involved in using big data.

Heterogeneity Heterogeneity is one of the major features of big data, also characterised as the variety. It is data of different types and formats. The heterogeneous data introduces the problems of data integration in big data analytics, making it difficult to obtain the desired value. The major cause of data heterogeneity is disparate sources of data that generate data in different forms. The data can be text data coming from emails, tweets or replies; log-data coming

from web activities, sensing and event data coming from IoT; and other forms. It is an important challenge to integrate this data for value-added analytics and positive decision making.

Uncertainty of Data The data gathered from heterogeneous sources like sensors, social media, web activities, and internal-records is inherently uncertain due to noise, incompleteness and inconsistency (e.g., there are 80% - 90% missing links in social networks and over 90% missing attribute values for a doctor diagnosis in clinic and health fields). Efficient analysis to discover value from these huge amounts of data demands tremendous effort and resources. However, as the volume, variety and velocity of the data increases, the uncertainty inherent in the data also increases, leading to doubtful confidence in the resulting analytics and predicted decisions.

Scalability The volume of data is drastically increasing and therefore an important challenge is to deal with the scalability of the data. It is also important to develop efficient analytics solutions and architectures that can scale up with the increasing data without compromising the accuracy or efficiency. Most of the existing learning algorithms cannot adapt themselves to the new big-data paradigms like dealing with missing data, working with partial data access or dealing with heterogeneous data sources. While the problem complexity of big data is increasing at a very fast rate, the computational ability and the solution capability is not increasing at a similar pace, posing a vital challenge.

Timeliness When looking for added business values, timing is of prime importance. It is related to capturing data, execution of analytics and making decisions at the right time. In a dynamic and rapidly evolving world, a slight delay (sometimes microseconds) could lead to incorrect analytics and predictions. In an example case of a bogus online bank transaction, the transaction must be disapproved in a timely manner to avoid possible money loss.

Data security Data storage and exchange in organizations has created challenges in data security and privacy. With the increasing sizes of data, it is important to protect e.g. transaction logs and data, real-time data, access control data, communication and encryption data. Also, it is important to keep track of data provenance, perform granular auditing of logs, and access control data to determine any misuse of data. Besides, the difference between legitimate use of data and customer privacy must be respected by organizations and they must have the right mechanisms in place to protect that data.

2.4 Big Data Value Chain

The ability to handle and process big data is vital to any organization. The previous sections have discussed data generation abilities, and the characteristics

and challenges of dealing with big data. This section covers the required activities and actions to handle such data to achieve business goals and objectives. The term value chain [358] is used to define the chain of activities that an organization performs to deliver a product for decision support management and services. A value chain is composed of a sequence of interconnected sub-services, each with its own inputs, transformation processes, and outputs. The noteworthy services are described below.

Data Acquisition This is the process of gathering data, filtering, and cleaning the data for storage and data analysis. Data acquisition is critical, as the infrastructure required for the acquisition of big data must bear low, predictable latency for capturing data as well as answering queries. It should be able to handle very high transaction volumes in a distributed scalable environment, and be able to support flexible and dynamic heterogeneous data.

Data Analysis Interpreting the raw data and extraction of information from the data, such that it can be used in informed decision making, is called data analysis. There could be multiple domain-specific analysis based on the source and use of data. The analysis includes filtering, exploring, and transforming data to extract useful and often hidden information and patterns. The analysis is further classified as business intelligence, data mining, or machine learning.

Data Curation This is the active and continuous management of data through its life cycle [350]. It includes the organization and integration of data from multiple sources and to ascertain that the data meets given quality requirements for its usage. Curation covers tasks related to controlled data creation, maintenance and management e.g. content creation, selection, validation or preservation.

Data Storage This is persistent, scalable data management that can satisfy the needs of applications requesting frequent data access and querying. RDBMS have remained a de facto standard for organizational data management for decades; however, its ability to handle data of limited capacity and well-defined structure (ACID properties, Atomicity, Consistency, Isolation, and Durability) has made it less suitable to handle big data that has variety, in addition to volume and velocity. Novel technologies are being designed to focus on scalability and cope with a range of solutions handling numerous data models.

Data Usage This is the analysis of data covering the business activities assisting in business decision making. The analysis is made possible through the use of specialised tools for data integration or querying.

3 Tools and Architectures

3.1 Big Data Architectures

Several reference architectures have been proposed to support the design of big data systems. Big data architecture is the conceptual model that defines the structure and behaviour of the system used to ingest and process "big data" for business purposes. The architecture can be considered a blueprint to handle the ingestion, processing, and analysis of data that is too large or complex for traditional database systems. The aim is to design a solution based on the business needs of the organization. Based on the requirements, the proposed solution must be able to handle different types of workloads like batch processing or real-time processing. Additionally, it should be able to perform analytics and mining on this large-scale data.

Good architecture design can help organizations to reduce costs, assist in making faster and better decisions, and predict future needs or recommend new solutions. However, the creation of such a system is not straightforward and certain challenges exist in designing an optimal architecture.

Data Quality: This is one of the important challenges in all domains of data handling. The data could be noisy, incomplete or simply missing. Substantial processing is desired to make sure that the resulting data is of the desired quality. It is a widely known fact that "data preparation accounts for about 80% of the work of data scientists".

Data Integration: The architecture must be able to handle the integration of heterogeneous data coming from disparate sources. It is challenging to handle and integrate data of multiple sizes and forms coming at different speeds from multiple sources. Finally, the system should be able to carry out meaningful analytics on the data to gain valuable insights.

Data Scale: It is important to design a system that works at an optimal scale without over-reserving the resources. At the same time, it should be able to scale up as needed without compromising performance.

In order to comply with the data value chain, any big data architecture comprises of the components that can allow to perform desired operations.

Data Sources: The data of an organization might be originating from databases, real-time sources like web-logs, activity data, IoT devices and many others. There should be data ingestion and integration components embedded in the architecture to deal with these data sources.

Data Ingestion: If the data is coming from the real-time sources, the architecture must support the real-time data ingestion mechanism.

Data Storage: Depending upon the number and types of data sources, efficient data storage is important for big data architecture. In the case of multiple types of data sources, a no-SQL "data lake" is usually built.

Data Processing: The data in the system needs to be queried and analysed, therefore it is important to develop efficient data-querying solutions, or data-processing tools that can process the data at scale. These processing solu-

tions can either be real-time or batch, depending upon the originating data and organizational needs.

Data Analysis: Specialised tools to analyse data for business intelligence are needed to extract meaningful insights from the data.

Data Reporting, and Visualisation: These are the tools used to make reports from the analysed data and to present the results in visual form.

Process Automation: Moving the data across the big data architecture pipeline requires automated orchestration. The ingestion and transformation of the data, moving it for processing, storage, and deriving insights and reporting must be done in a repeatable workflow to continuously gain insights from the data.

Depending upon the type of data and the individual requirements of the organizations, the selected tasks must be handled by choosing corresponding services. To support the tasks and selected services, the overall architecture to realise the data value chain is designed. The big data architectures are mainly divided into three main types as below:

Lambda Architecture The lambda architecture, first proposed by Nathan [99], addresses the issue of slow queries results on batch data, while real-time data requires fast query results. Lambda architecture combines the real-time (fast) query results with the queries (slow) from batch analysis of older data. Lambda architecture creates two paths for the data flow. All data coming into the system goes through these two paths. Batch Layer: also known as the cold path, stores all the incoming data in its raw form and performs batch processing on the data. This offers a convenient way to handle reprocessing. This layer executes long-living batch-processes to do analyses on larger amounts of historical data. Speed Layer: also known as the hot path, analyses the data in real-time. This layer is designed for low latency. This layer executes small/mini batch-processes on data according to the selected time window (e.g. 1 second) to do analyses on the latest data. Serving Layer: This layer combines the results from the batch and speed processing layer to enable fast interactive analyses by users.

Kappa Architecture Kappa architecture was proposed by Jay Kreps [263] as an alternative to lambda architecture. Like Lambda architecture, all data in Kappa architecture flows through the system, but uses a single path, i.e. a stream processing system. Kappa architecture focuses only on data stream processing, real-time processing, or processing of live discrete events. Examples are IoT events, social networks, log files or transaction processing systems. The architecture assumes that: The events are ordered and logged to a distributed file system, from where they can be read on demand. The platform can repeatedly request the logs for reprocessing in case of code updates. The system can handle online machine learning algorithms.

Microservices-based architecture "Microservice Architecture" has emerged over the last few years to describe a particular way of designing software applica-

tions as suites of independently deployable services [283]. Microservices architecture makes use of loosely coupled services which can be developed, deployed and maintained independently. These services can be built for business capability, automated deployment, intelligence in the endpoints, and decentralized control of languages and data.

Microservices-based architecture is enabled by a multitude of technology advancements like the implementation of applications as services, emergence of software containers for service deployment, orchestration of containers, development of object stores for storing data beyond container lifecycle, requirement for continuous integration, automated testing, and code analysis to improve software quality. Microservices-based architecture allows fast delivery of individual services independently. In this architecture, all the components of big data architecture are treated as services, deployable on a cluster.

3.2 Tools to Handle Big Data

In order to deal with big data, a variety of specialised tools have been created. This section provides an overview of the existing tools based on their functionalities. A distributed platform handling big data is made up of components needed for the following tasks. We will cover the tools developed to perform these specific tasks in the preceding sections.

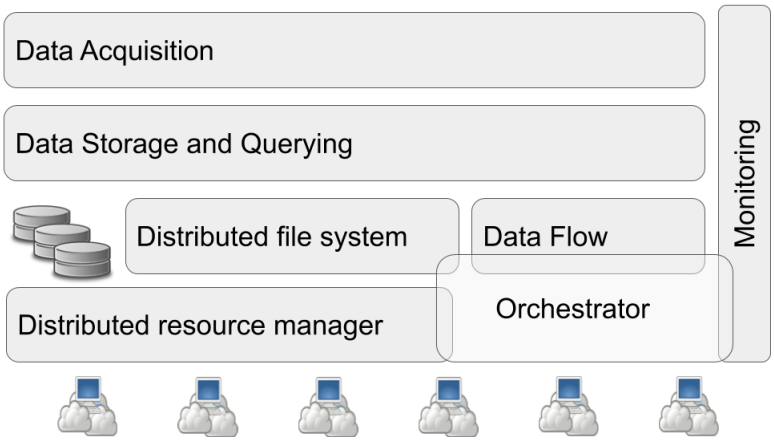

Fig. 1: Classification of tools to handle big data

Resource Orchestration Distributed coordination and consensus is the backbone of distributed systems. Distributed coordination deals with tasks like telling each node about the other nodes in the cluster and facilitating communication and high availability. High availability guarantees the presence of the mediator

node and avoids a single point of failure by replication resulting in a fault-tolerant system. In a distributed setting, the nodes must share common configurations and runtime variables and may need to store configuration data in a distributed key-value store. The distributed coordination manages the sharing of the locks, shared-variables, realtime-configurations at runtime among the nodes.

In addition, fault-tolerant distributed systems contain methods to deal with the consensus problem, i.e. the servers or mediators in the distributed system perform agreement on certain values or variables, e.g. there can be a consensus that the cluster with 7 servers can continue to operate if 4 servers get down, i.e. with only 3 servers running successfully. The popular orchestration tools are Apache zookeeper and etcd. The systems are consistent and provide primitives to be used within complex distributed systems. Such distributed orchestrators ease the development of distributed applications and make them more generic and fault resilient.

Apache Zookeeper Apache Zookeeper [209] is an open-source project that originated from the Hadoop ecosystem and is being used in many top-level projects including Ambari, Mesos, Yarn, Kafka, Storm, Solr, and many more (discussed in later sections). Zookeeper is a centralised service for managing cluster configuration information, naming and distributed synchronization and coordination. It is a distributed key-value store that allows the coordination of distributed processes through a shared hierarchical name space of data registers (znodes), like a file system. Zookeeper provides high throughput, low latency, high availability and strictly ordered access to the znodes. Zookeeper is used in large distributed clusters and provides fault tolerance and high availability. These aspects allow it to be used in large complex systems to attain high availability and synchronization for resilient operations. In these complex distributed systems, Zookeeper can be viewed as a centralized repository where distributed applications read and write data. It is used to keep the distributed application functioning together as a single unit by making use of its synchronization, serialization and coordination abilities.

Etcd Etcd [1] is a strongly consistent distributed reliable key-value store that is simple, secure and fast. It provides a reliable way to store data that needs to be accessed by a distributed system to provide consistent cluster coordination and state management. The name etcd is derived from distributing the Unix "/etc" directory used for global configurations. It gracefully handles leader elections and can tolerate machine failure, even in the leader node. The leaders in etcd handle all client requests needing consensus. Requests like reading can be handled by any cluster node. The leader accepts, replicates and commits the new changes after the followers verify the receipt.

Etcd uses the raft protocol to maintain the logs of state-changing events. It uses full replication, i.e. the entire data is available on every node, making it highly available. This also makes it possible that any node can act as a leader. The applications can read and write data to etcd and it can be used for storing database connection details, or feature flags. These values can be watched and allow the applications to reconfigure themselves when values change. In addition,

etcd consistency is used to implement leader election or distributed locking. etcd is used as the coordinating mechanism for Kubernetes and Cloud Foundry. It is also used in production environments by AWS, Google Cloud Platform and Azure.

Resource Management The big data platform works on top of a set of distributed computing and memory resources. The resource manager performs the task of resource allocation in terms of CPU time and memory usage. In a cluster, multiple applications are usually deployed at one time, e.g. it is common to have a distributed application like Apache Spark running in parallel to a distributed database for storage like Apache Hbase in the same cluster. A resource manager is an authority that arbitrates resources among all the applications in the system. In addition, the resource manager is also responsible for job scheduling with the help of a scheduler, or an application master.

YARN Yet another resource manager (YARN) [444] is an important integral part of the Hadoop ecosystem and mainly supports Hadoop workloads. In YARN, the application-level resource manager is a dedicated scheduler that runs on the master daemon and assigns resources to the requesting applications. It keeps a global view of all resources in the cluster and handles the resource requests by scheduling the request and assigning the resources to the requesting application. It is a critical component in the Hadoop cluster and runs on a dedicated master node.The resource manager has two components: a scheduler and an application manager. The application manager receives the job-submissions, looks for the container to execute the ApplicationMaster and helps in restarting the ApplicationMaster on another node in case of failure. The ApplicationMaster is created for each application and it is responsible for the allocation of appropriate resources from the scheduler, tracking their status and monitoring their progress. ApplicationMaster works together with the Node Manager. The Node manager runs on slave daemon and is responsible for the execution of tasks on each node. It monitors their resource usage and reports it to the ResourceManager. The focus of YARN on one aspect at a time enables YARN to be scalable, generic and makes it able to support multi-tenant cluster. The High available version of Yarn uses Zookeeper to establish automatic failover.

Mesos Apache Mesos is an open-source cluster manager [233] that handles workloads in a distributed environment through dynamic resource sharing and isolation. It is also called a distributed systems kernel. Mesos works between the application layer and the operating system and makes it easier to manage and deploy applications in large distributed clusters by doing resource management. It turns a cluster into a single large pool of resources by leveraging the features of modern kernels of resource isolation, prioritization, limiting, and accounting, at a higher level of abstraction. Mesos also uses zookeeper to achieve high availability and recovery from master failure. Mesos carries out microscale resource management as it works as a microkernel.

Data Flow: Message Passing Message passing is crucial to distributed big data applications that must deal with real-time data. This data could be event logs, user activities, sensor signals, stock exchanges, bank transactions, among many others. Efficient and fault free ingestion of this real-time data is critical for real-time applications. Message passing solutions are needed for real-time streaming applications and data flows.

Message passing tools, as the name suggests, assist in communication between the software components of a big data processing pipeline. The systems usually decouple the sender and receiver by using a message broker that hides the implementation details like the operating system or network interface from the application interfaces. This creates a common platform for messaging that is also easy to develop for the developers. The applications of message passing pipelines are website activity monitoring, metrics collection, log aggregation etc. Below we briefly discuss Apache Kafka, which is frequently used in real-time big data applications.

Apache Kafka Apache Kafka [147] is a distributed messaging system that uses the publish-subscribe mechanism. It was developed to support continuous and resilient messaging with high throughput at LinkedIn. Kafka is a fast, scalable, durable, and fault-tolerant system. It maintains feeds of messages in categories called topics. These topics are used to store messages from the producers and deliver them to the consumers who have subscribed to that topic.

Kafka is a durable, high volume message broker that enables applications to process, persist and re-process streaming data. Kafka has a straightforward routing approach that uses a routing key to send messages to a topic. Kafka offers much higher performance than message brokers like RabbitMQ. Its boosted performance makes it suitable to achieve high throughput (millions of messages per second) with limited resources.

Data Handling The data handling and acquisition assists in collecting, selecting, filtering and cleaning the data being received and generated. This data can be later stored in a data warehouse, or another storage solution, where further processing can be performed for gaining the insights.

Apache Flume Apache Flume [198] is a framework to collect massive amounts of streaming event data from multiple sources, aggregate it, and move it into HDFS. It is used for collecting, aggregating, and moving large amounts of streaming data such as log files, events from various sources like network traffic, social media, email messages etc. to HDFS. Flume provides reliable message delivery. The transactions in Flume are channel-based where two transactions (one sender and one receiver) are maintained for each message. If the read rate exceeds the write rate, Flume provides a steady flow of data between read and write operations. Flume allows ingestion of data from multiple servers (and sources) into Hadoop.

Apache Sqoop Most of the older companies have stored their data on RDBMS, but with the increase in data sizes beyond terabytes, it is important to switch to HDFS. Apache Sqoop [428] is a tool designed to transfer bulk data

between structured data stores such as RDBMS and Hadoop in an efficient manner. Sqoop imports data from external datastores into HDFS and vice versa. It can also be used to populate tables in Hive and HBase. Sqoop uses a connector-based architecture which supports plugins providing smooth connectivity to the external systems.

Data Processing Data-flow processing technologies are mainly categorised into batch (historical data) processing systems and stream (real-time) processing systems.
Batch processing systems are high throughput systems for processing high volumes of data collected over some time. The data is collected, entered, processed and then the batch results generated resulting in high latency systems.
Stream processing systems are high throughput i.e. the system continuously receives data that is under constant change (e.g. traffic control, sensor data, social media), low latency stream processing systems. The data is processed on the fly and produces real-time insights. There are three main methods for streaming: At least once, At most once, and Exactly once.
Until a few years ago, a clear distinction between these two processing systems existed. However, recent technologies such as Apache Spark and Apache Flink can handle both kinds of processing, diminishing this distinction. We will discuss some of the key technologies in the following sections.

Hadoop MapReduce Hadoop is a platform for distributed storage and analysis of very large data sets. It has four main modules: Hadoop Common, HDFS, MapReduce and YARN [153]. MapReduce is the distributed data processing engine of Hadoop. It is a programming model and provides a software framework to write the applications for distributed processing of very large amounts of data in parallel. MapReduce processes the data in two phases: The map phase and the reduce phase. In the map phase, the framework reads data from HDFS. Each dataset is called an input record and split into independent chunks that are processed by the map tasks in parallel. In the reduce phase, the results from the map phase are processed and stored. The storage target can either be a database or back HDFS or something else. Working with MapReduce requires a low level and specialised design thinking and programming models, making it challenging for developers to create generic applications. As a result, many tools have been developed around Hadoop MapReduce to address these limitations. These tools include:
Apache Pig: This provides a high-level language for expressing data analysis programs that can be executed in MapReduce [150]. The platform was developed by Yahoo. The developers can write programs for data manipulation and transformation as data flow sequences using Pig Latin language. These programs are easy to write, understand, and maintain. In addition, Apache Pig offers an infrastructure to evaluate and optimize these programs automatically. This allows developers to focus more on semantics and productivity. Apache Pig can execute its jobs in Apache Tez, or Apache Spark (covered in the following sections).
Apache Hive: This offers a higher-level API to facilitate reading, writing, and

managing large datasets [203] residing in distributed storage (e.g. HDFS) using SQL-like queries in a custom query language, called HiveQL. Implicitly, each query is translated into MapReduce commands.
Apache Mahout: This is a machine learning library [337] developed to be used with MapReduce. It provides an API for distributed or scalable machine learning algorithms mostly focusing on linear algebra. It provides algorithms like classification, likelihood estimation, and clustering. All algorithms are implicitly transformed into MapReduce jobs.

Apache Spark Apache Spark is a generic, in-memory data processing engine [480]. It provides high-level APIs in Java, Python and Scala. Apache Spark has simplified the programming complexity by introducing the abstraction of Resilient Distributed Datasets (RDD), i.e. a logical collection of data partitioned across machines. The rich API for RDDs manipulation follows the models for processing local collections of data, making it easier to develop complex programs. Spark provides higher-level constructs and libraries to further facilitate users in writing distributed applications. At the time of writing, Apache Spark provides four libraries:
Spark SQL - Offers support for SQL querying of data stored in RDDs, or an external data source. It allows structured data processing using high-level collections named dataset and data frame. A Dataset is a distributed collection of data and a DataFrame is a Dataset organized into named columns. It is conceptually similar to a table in a relational database. The DataFrames can be constructed in numerous different ways like reading from structured data files, tables in Hive, external databases, or existing RDDs.
Spark streaming - Spark implements stream processing by ingesting data in mini-batches. Spark streaming makes it easy to build scalable fault-tolerant real-time applications. The data can be ingested from a variety of streaming sources like Kafka, Flume (covered in earlier sections). This data can be processed using complex real-time algorithms using a high-level API.
MLlib Machine Learning Library - Provides scalable machine learning algorithms. It provides common algorithms for classification, regression, clustering, algorithms for feature extraction, feature selection and dimensionality reduction, high-level API for machine learning pipelines, saving and loading algorithms, and utilities for linear algebra and statistics.
GraphX - Provides a distributed graph processing using graph-parallel computation. GraphX extends the Spark RDD by introducing "Graph": a directed multigraph with properties attached to each vertex and edge. GraphX comes with a variety of graph operators like subgraph, joinVertices, or algorithms like pageRank, ConnectedComponents, and several graph builders that allow building a graph from a collection of vertices and edges from RDD or other data sources.

Apache Flink Apache Flink is a true distributed streaming data-flow engine [69] and offers a unified stream and batch processing. It treats batch processing as a special case of streaming with bounded data. The APIs offered by Flink are similar but the implementation is different. Flink executes arbitrary dataflow

programs in a data-parallel and pipelined manner. It offers a complete software stack of libraries using building blocks, exposed as abstract data types, for streams (DataStream API), for finite sets (DataSet API) and for relational data processing (relational APIs - the Table API and SQL). The high-level libraries offered by Apache Flink are:
Gelly: Flink Graph - provides methods and utilities to simplify the development of graph analysis applications in Flink. The graphs can be transformed and modified using high-level functions similar to the ones provided by the batch processing API. Gelly provides graph algorithms like pageRank, communityDetection, connectedComponents, or shortestPath finding.
Machine Learning:: FlinkML is a machine learning library aimed to provide a list of machine learning algorithms. At the moment, it has been temporarily deprecated in Apache Flink 1.9.0 for the sake of developing ML core and ML pipeline interfaces using high-level APIs.
FlinkCEP: Complex event processing for Flink - Allows detection of event patterns in the incoming stream.
State Processor API- - provides functionality to read, write, and modify save points and checkpoints using DataSet API. It also allows using relational Table API or SQL queries to analyze and process state data.

Data Storage: Distributed File Systems Distributed file systems allow access to the files from multiple hosts, in addition to distributing the storage of large files over multiple machines. Such systems mostly provide the interfaces and semantics, similar to the existing local files systems, while the distributed file system handles the network communication, data movement and distributed directories seamlessly.

Hadoop Distributed File System (HDFS) HDFS, the main component of the Hadoop ecosystem, has become the de facto standard for distributed file systems [62]. It is known as the most reliable storage system. HDFS is designed to run on commodity hardware, making it more popular for its cost-effectiveness. In addition to working with the conventional file management commands (e.g. ls, rm, mkdir, tail, copy, etc), HDFS also works with a REST API that complies with the FileSystem/FileContext interface for HDFS. HDFS architecture is designed to store very large files and does not suit models with large numbers of small files. The files are split into blocks which are then distributed and replicated across the nodes for fault-tolerance. HDFS stores data reliably, even in the case of hardware failure. HDFS provides parallel access to data, resulting in high throughput access to application data.

Data Storage and Querying RDBMS and SQL have remained the main choice for data storage and management for organizations for years. Gradually, the main strength of RDBMS technology (the fixed schema design) has turned into its fundamental weakness in the era of big and heterogeneous data. Today's data appears in structured and unstructured forms and originates from a variety of sources such as emails, log files, social media, sensor events etc. Besides, high

volumes of data are being generated and are subject to high rates of change. On the other hand, one of the key characteristics of big data applications is that they demand real-time responses, i.e. data needs to be stored, such that it can be accessed quickly when required. The non-conventional, relatively new NoSQL (not only SQL) stores are designed to efficiently and effectively tackle these big data requirements. Not only do these stores support dynamic schema design but they also offer increased flexibility, scalability and customization compared to relational databases. These stores are built to support distributed environments, with the ability to scale horizontally by adding new nodes as demand increases. Consistent with the CAP theorem (which states that distributed systems can only guarantee at most two properties from Consistency, Availability and Partition tolerance), NoSQL stores compromise consistency in favour of high availability and scalability. Generally, NoSQL stores support flexible data models, provide simple interfaces and use weak consistency models by abandoning the ACID (Atomicity, Consistency, Isolation, and Durability) transactions in favour of BASE (Basically Available, Soft state, Eventually Consistent) transaction models. Based on the data models supported by these systems, NoSQL databases can be categorised into four groups, i.e. key-value stores, document stores, column-oriented stores and graph databases. The following section describes these NoSQL database models in further detail and lists a few examples of the technologies per model.

Key-Value Stores Key-value stores can be categorised as the simplest NoSQL databases. These stores are designed for storing schema-free data as Key-Value pairs. The keys are the unique IDs for the data, and they can also work as indexes for accessing the data. The Values contain the actual data in the form of attributes or complex objects. All the values may not share the same structure.
Examples: Redis, Riak KV, Amazon DynamoDB, Memcached, Microsoft Azure Cosmos DB, and etcd.

Document Stores Document stores are built upon the idea of key-value stores. They pair each key with a complex data structure described as a document. These documents may contain different key-value pairs, key-array pairs or even nested documents. The document stores are designed for storing, retrieving and managing document-oriented information, also known as semi-structured data. There is no schema that all documents must adhere to as in the case for records in relational databases. Each document is assigned a unique key, which is used to retrieve the document. However, it is possible to access documents by querying their internal structure, e.g searching for a field with the specified value. The capability of the query interface is typically dependent on the encoding format like XML or JSON.
Examples: CouchDB, MongoDB

Column-Oriented Stores Column-oriented stores are also known as wide-column stores and extensible record stores. They store each column continuously, i.e. on disk or in-memory each column is stored in sequential blocks. Instead of storing data in rows, these databases are designed for storing data tables as

sections of columns of data. Therefore, these stores enable faster column-based analytics compared to traditional row-oriented databases.
Examples: Apache HBase, Cassandra

4 Harnessing Big Data as Knowledge Graphs

Today, the term big data is potentially misleading as the size is only one of many important aspects of the data. The word big promotes the misconception that more data means good data and stronger insights. However, it is important to realise that data volume alone is not sufficient to get good answers. The ways we distribute, organize, integrate, and represent the data matters as much as, if not more than, the size of the data. In this section, we briefly cover the variety or the heterogeneity of the data and the possibility of organizing this data as a graph. Organizing the data as a graph has several advantages compared to alternatives like database models. Graphs provide a more intuitive and succinct abstraction for the knowledge in most of the domains. Graphs encode the entities as nodes, and their relationships as edges between entities. For example, in social interactions the edges could represent friendship, co-authorship, co-worker-ship, or other types of relations, whereas people are represented as the nodes. Graphs have the ability to encode flexible, incomplete, schema-agnostic information that is typically not possible in the relational scenario. Many graph query languages cannot only support standard operations like joins but also support specialised operators like arbitrary path-finding. At the same time, formal knowledge representation (based on Ontologies) formats could also be used to create Graphs in a semantically coherent and structured representation (RDF, RDFS). The term knowledge graph was popularised in 2012 by Google with the slogan ""things not strings"" with an argument that the strings can be ambiguous but in the Knowledge Graphs, the entities (the nodes in a Knowledge Graph) can be disambiguated more easily by exploiting their relationships (edges/properties) with other entities. Numerous definitions of Knowledge Graphs have been proposed in the literature, and a recent and generic definition portrays the "knowledge graph as a graph of data intended to accumulate and convey knowledge of the real world, whose nodes represent entities of interest and whose edges represent relations between these entities" [199]. A high number of public, open, cross-domain knowledge graphs have been created and published online. Examples include DBPedia, Wikidata or YAGO, which are either created by the community or extract knowledge from Wikipedia. Domain dependent open knowledge graphs have also been published covering areas like geography, life sciences, and tourism. At the same time, numerous enterprise knowledge graphs (mostly in-house) are created by e.g. IBM, Amazon, Facebook, LinkedIn and many others. The creation of these knowledge graphs mainly involves three methods.
Manual Curation e.g. Cyc, Wikidata, Freebase etc.
Creation using Semi-structured sources e.g. Wikipedia (from Wikipedia infoboxes), YAGO (WordNet, Wikipedia etc.) BableNet etc.
Creation from Unstructured Sources e.g. NELL (free text), WebIsA (free

text)
As briefly discussed above, such graphs could be created schema-agnostically, as well as using a formal ontology that defines the set of concepts and categories in a given domain alongside their properties and the relations. The knowledge contained in the knowledge graphs can be characterized around two main dimensions: a) Coverage of a single domain, which can be defined by the number of Instances. The instances depict the details covered in a given knowledge graph in a particular area, and more instances mean more details. Coverage could further be defined by the number of assertions, i.e. the relationships contained in the graph. Also, the link degree (average, median) can also assist in estimation of the coverage of the graph. For b) Knowledge Coverage (multiple domains), one can consider the number of classes in the schema, the number of relations, the class hierarchy (depth and width), or the complexity of schema can help in assessing the breadth and depth of the knowledge covered by a given knowledge graph. In practice, the graphs can differ in their sizes in orders of magnitude, but the complexity (linkage) of smaller graphs could still be higher. Similarly, the underlying schema could either be simple or rather deep and detailed. The number of instances per class could vary; on the contrary, there could be fewer instances per class, covering more classes in total. In conclusion, the knowledge graphs differ strongly in size, coverage, and level of detail.

4.1 Graph Stores

In order to handle large sizes of this relatively new-hyped knowledge representation format, several tools have been created which can be categorised into two types, one more general and simple, like graphs, and other relatively formal for RDF data named as Triple Stores.

Graph Databases Graph databases are based on graph theory and store data in graph structures using nodes and edges connecting each other through relations. These databases are designed for data containing elements which are interconnected, with an undetermined number of relations between them. Graph databases usually provide index-free adjacency, i.e. every element contains a direct pointer to its adjacent elements and no index lookups are necessary. Examples: Neo4J, FlockDB, HyperGraphDB

Triple Stores Triple stores are database management systems for the data modelled using RDF. RDF data can be thought of as a directed labelled graph wherein the arcs start with subject URIs, are labelled with predicate URIs, and end up pointing to object URIs or scalar values. This RDF data can be queried using SPARQL query language. Triple stores can be classified into three categories: Native triple stores - Triple stores implemented from scratch exploiting the RDF data model to efficiently store and access the RDF data. Examples: Stardog, Sesame, OWLIM RDBMS-backed triple stores - Triple stores built by

adding an RDF specific layer to an existing RDBMS. Example: OpenLink Virtuoso NoSQL triple stores - Triple stores built by adding an RDF specific layer to existing NoSQL databases. Example: CumulusRDF (built on top of Cassandra).

Efficient handling of large-scale knowledge graphs requires the use of distributed file systems, distributed data stores, and partitioning strategies. Apart for several centralised systems, many recent graph processing systems have been built using existing distributed frameworks, e.g. Jena-HBase [241] and H2RDF [341], H2RDF+ [342] make use of HBase, Rya [363] makes use of Accumulo, D-SPARQ [320] works using MongoDB. S2RDF [385], S2X [384], SPARQLGX [168] and SparkRDF [78] handle RDF data using Apache Spark. The main idea behind representing data as a graph is not only querying the data, but also efficient knowledge retrieval including reasoning, knowledge base completion, enrichment (from other sources), entity linking and disambiguation, path mining, and many other forms of analytics. It can be seen from many recent surveys [192, 473, 235] that several systems have been proposed in the literature to deal with one or a few of the many aspects of large-scale knowledge graph processing. It is important to realize this gap and the need for a scalable framework that caters for different tasks for large-scale knowledge graphs.

5 Conclusion

This chapter connects the term big data and knowledge graphs. The first section of this chapter provides an overview of big data, its major enabling technologies, the key characteristics of big data, the challenges that it poses, and the necessary activities to create a big data value chain. In the second section, we cover the big data architectures and provide a taxonomy of big data processing engines. In the last section, we connect the big data with large-scale knowledge graphs covered in chapter 1 and 2 of this book. We discuss a few key technologies and cover the possibilities and key challenges to harness large-scale knowledge graphs.