Massive Data Analysis: Tasks, Tools, Applications and Challenges

Murali K. Pusala, Mohsen Amini Salehi, Jayasimha R. Katukuri, Ying Xie and Vijay V. Raghavan

Abstract In this study, we provide an overview of the state-of-the-art technologies in programming, computing, and storage of the massive data analytics landscape. We shed light on different types of analytics that can be performed on massive data. For that, we first provide a detailed taxonomy on different analytic types along with examples of each type. Next, we highlight technology trends of massive data analytics that are available for corporations, government agencies, and researchers. In addition, we enumerate several instances of opportunities that exist for turning massive data into knowledge. We describe and position two distinct case studies of massive data analytics that are being investigated in our research group: recommendation systems in e-commerce applications; and link discovery to predict unknown association of medical concepts. Finally, we discuss the lessons we have learnt and open challenges faced by researchers and businesses in the field of massive data analytics.

Murali K. Pusala, Mohsen Amini Salehi, Vijay V.Raghavan
Center of Advanced Computer Studies (CACS), University of Louisiana Lafayette, Louisiana, 70503 , e-mail: {mxp6168,amini,vijay}@cacs.louisiana.edu

Jayasimha R. Katukuri
Center of Advanced Computer Studies (CACS), University of Louisiana Lafayette, Louisiana, 70503, e-mail: jaykatukuri@gmail.com

Ying Xie
Department of Computer Science, Kennesaw State University, Kennesaw, GA 30144 e-mail: yxie2@kennesaw.edu

1 Introduction

1.1 Motivation

Growth of Internet usage in the last decade has been at an unprecedented rate from 16 million, which is about 0.4% of total population in 1995, to more than 3 billion users, which is about half of the world's population in mid-2014. This revolutionized the way people communicate and share their information. According to [60], just during 2013, 4.4 zettabytes (4.4×2^{70}bytes) of information have created and replicated, and it estimated to grow up to 44 zettabytes by 2020. Below, we explain few sources from such massive data generation.

Facebook[1] has an average of 1.39 billion monthly active users exchanging billions of messages and postings every day [6]. There is also a huge surge in multimedia content like photos and videos. For example, in popular photo sharing social network Instagram [2], on average, 70 million photos uploaded and shared every day [12]. According to other statistics published by Google on its video streaming service, YouTube[3], has approximately 300 hours of video uploaded every minute and billions of views generated every day [21].

Along with Individuals, organizations are also generating a huge amount of data, mainly due to increased use of networked sensors in various sectors of organizations. For example, by simply replacing traditional bar code systems with Radio Frequency Identification (RFID) systems organizations have generated 100 to 1000 times more data [58].

Organization's interest on customer behavior is another driver for producing massive data. For instance, Wal-Mart[4] handles more than a million customer transactions each hour and maintains a database that holds more than 2.5 petabytes of data [58]. Many businesses are creating a 360° view of a customer by combining transaction data with social networks and other sources.

Data explosion is not limited to individuals or organizations. With the increase of scientific equipment sensitivity and advancements in technology, the scientific and research, community is also generating a massive amount of data. Australian Square Kilometer Array Pathfinder radio telescope [3] has 36 antennas streams approximately 250 GB of data per second per antenna that collectively produces nine terabytes of data per second. In another example, particle accelerator, particle detector, and simulations at Large Hadron Collider (LHC) at CERN [13] generate approximately 15 petabytes of data per year.

[1] https://facebook.com

[2] https://instagram.com

[3] http://www.youtube.com

[4] http://www.walmart.com

1.2 Big Data Overview

The rapid explosion of data is usually referred as *"Big Data"*, which is a trending topic in both industry and academia. Big data (aka Massive Data) is defined as, data that cannot be handled or analyzed by conventional processing and storage tools. Big data is also characterized by features,known as *5V's*. These features are: *volume*, *variety*, *velocity*, *variability*, and *veracity* [26] [33].

Traditionally, most of the available data is structured data and stored in conventional databases and data-warehouses for supporting all kinds of data analytics. With the Big data, data is no longer necessarily structured. Instead, it contains a *variety* of data sources, including structured, semi-structured, and unstructured data [26]. It is estimated that 85% of total organizational data are unstructured data [58] and almost all the data generated by individuals (e.g., emails, messages, blogs, and multimedia) are unstructured data too. Traditional relational databases are no longer a viable option to store text, video, audio, images, and other forms of unstructured data. This creates a need for special types of NoSQL databases and advanced analytic methods.

Velocity of data is described as problem of handling and processing data at the speeds at which they are generated to extract a meaningful value. Online retailers store every attribute (e.g., clicks, page visits, duration of visits to a page) of their customers' visits to their online websites. There is a need to analyze customers' visits within a reasonable timespan (e.g., real-time) to recommend similar items and related items with respect to the item a customer is looking at. This helps companies to attract new customers and keep an edge over their competitors. Some organizations analyze data as a stream in order to reduce data storage. For instance, LHC at CERN [13] analyzes data before storing to meet the storage requirements. Smart phones are equipped with modern location detection sensors that enable us to understand the customer behavior while, at the same time, creating the need for real-time analysis to deliver location-based suggestions.

Data *variability* is the variation in data flow with time of day, season, events, etc. For example, retailers sell significantly more in November and December compared to rest of year. According to [1], traffic to retail websites surges during this period. The challenge, in this scenario, is to provide resources to handle sudden increases in users' demands. Traditionally, organizations were building in-house infrastructure to support their peak-estimated demand periods. However, it turns out to be costly, as the resources will remain idle during the rest of the time. However, the emergence of advanced distributed computing platforms, known as 'the cloud', can be leveraged to enable on-demand resource provisioning through third-party companies. Cloud provides efficient computational, storage and other services to organizations and relieves them from the burden of over-provisioning resources [53].

Big data provides advantage in decision-making and analytics. However, among all data generated in 2013 only 22% of data are tagged, or somehow characterized as useful data for analysis, and only 5% of data are considered valuable or "Target Rich" data. The quality of collected data, to extract a value from, is referred as *veracity*. The ultimate goal of an organization in processing and analyzing data is

to obtain hidden information in data. Higher quality data increases the likelihood of effective decision-making and analytics. A McKinsey study found that retailers using full potential from Big data could increase the operating margin up to 60% [48]. To reach this goal, the quality of collected data needs to be improved.

1.3 Big Data Adoption

Organizations have already started tapping into the potential of Big data. Conventional data analytics are based on structured data, such as the transactional data, that are collected in a data warehouse. Advanced massive data analysis helps to combine traditional data with data from different sources for decision-making. Big data provides opportunities for analyzing customer behavior patterns based on customer actions inside (e.g., organization website) and outside (e.g., social networks).

In a manufacturing industry, data from sensors that monitor machines' operation are analyzed to predict failures of parts and replace them in advance to avoid significant down time [11]. Large financial institutions are using Big data analytics to identify anomaly in purchases and stop frauds or scams [23].

In spite of the wide range of emerging applications for Big data, organizations are still facing challenges to adopt Big data analytics. A report from AIIM [4], identified three top challenges in the adoption of Big data, which are lack of skilled workers, difficulty to combine structured and unstructured data, and security and privacy concerns. There is a sharp rise in the number of organizations showing interest to invest in Big data related projects. According to [8], in 2014, 47% of organizations are reportedly investing in Big data products, as compared to 38% in 2013. IDC predicted that the Big data service market has reached 11 billion dollars in 2009 [20] and it could grow up to 32.4 billion dollars by end of 2017 [15]. Venture capital funding for Big data projects also increased from 155 million dollars in 2009 to more than 893 million dollars in 2013 [20].

1.4 The Chapter Structure

From the late 90's, when Big data phenomenon was first identified, until today, there has been many improvements in computational capabilities, storage devices have become more inexpensive, thus, the adoption of data-centric analytics has increased. In this study, we provide an overview of Big data analytic types, offer insight into Big data technologies available, and identify open challenges.

The rest of this paper is organized as following. In section 2, we explain different categories of Big data analytics, along with application scenarios. Section 3 of the chapter describes Big data computing platforms available today. In section 4, we provide some insight into the storage of huge volume and variety data. In that section, we also discuss some commercially available cloud-based storage services. In

section 5, we present two real-world Big data analytic projects. Section 6 discusses open challenges in Big data analytics. Finally, we summarize and conclude the main contributions of the chapter in section 7.

2 Big Data Analytics

Big data analytics is the process of exploring Big data, to extract hidden and valuable information and patterns [52]. Big data analytics helps organizations in more informed decision-making. Big data analytics applications can be broadly classified as *descriptive*, *predictive*, and *prescriptive*. Figure 1 illustrates the data analytic classes, techniques, and example applications. In the rest of this section, with reference to Figure 1, we elaborate on these Big data analytic types.

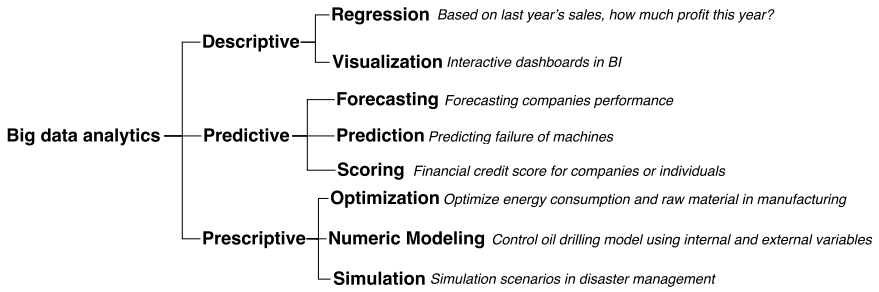

Fig. 1 Types of Big data analytics: The second level in the hierarchy is the categorization of analytics. The third level,explains the typical techniques, and provides example in the corresponding analytic category.

2.1 Descriptive Analytics

Descriptive analytics mines massive data repositories to extract potential patterns existing in the data. Descriptive analytics drills down into historical data to detect patterns like variations in operating costs, sales of different products, customer buying preferences, etc.

Typically it is the first step of analytics in decision-making, answering the question of "what has happened? ". It summarizes raw data into a human understandable format. Most of the statistical analysis used in day-to-day Business Intelligence (BI) regarding a company's production, financial operations, sales, inventory and customers come under descriptive analytics [62]. Analytics involve simple techniques, such as regression to find correlation among various variables and drawing charts,

to identify trends in the data, and visualize data in a meaningful and understandable way, respectively.

For example, Dow Chemicals used descriptive analytics to identify under-utilized space in its offices and labs. As a result, they were able to increase space utilization by 20% and save approximately $4 million annually [22].

2.2 Predictive Analytics

With descriptive analytics organizations can understand what happened in the past. However, at a higher level of decision-making is to address the question of "what could happen? ". Predictive analytics helps to combine massive data from different sources with the goal of predicting future trends or events. Predictive analytics evaluates the future, by forecasting trends, by generating prediction models, and by scoring.

For example, industries use predictive analytics to predict machine failures using streaming sensor data [11]. Organizations are able to forecast their sales trends or overall performance [45]. Financial institutions devote a lot of resources to predict credit risk scores for companies or individuals. Eventhough predictive analytics cannot predict with 100% certainty, but it helps the companies in estimating future trends for more informed decision-making.

Southwest airlines has partnered with National Aeronautics and Space Administration (NASA) to work on a Big data-mining project [14]. They apply text-based analysis on data from sensors in their planes in order to find patterns that indicate potential malfunction or safety issues.

Purdue University uses Big data analytics to predict academic and behavioral issues [18]. For each student, the system predicts and generates, a risk profile indicating how far a student succeeds in a course and labels the risk levels as green (high probability of success), yellow (potential problems), and red (risk of failure) by using data from various sources, such as student information and course management systems for this analytics.

E-commerce applications apply predictive analytics on customer purchase history, customer behavior online, like page views, clicks, and time spend on pages, and from other sources [59] [27]. Retail organizations are able to predict customer behavior to target appropriate promotions and recommendations [38]. They use predictive analysis to determine the demand of inventory and maintain the supply chain accordingly. Predictive analysis also helps to change price dynamically to attract consumers and maximize profits [2].

2.3 Prescriptive Analytics

Descriptive and predictive analytics helps to understand the past and predict the future. The next stage in decision-making is "how can we make it happen?" - the answer is prescriptive analytics. The goal of prescriptive analytics is to assist professionals in assessing the impact of different possible decisions. It is a relatively new analytic method. According to Gartner [9], only 3% of companies use prescriptive analytics in their decision-making. Prescriptive analytics involves techniques such as optimization, numerical modeling, and simulation.

Oil and Gas exploration industries use prescriptive analytics to *optimize* the exploration process . Explorers are using massive datasets from different sources in the exploration process and use prescriptive analytics to optimize drilling location [16]. They use earth's sedimentation characteristics, temperature, pressure, soil type, depth, chemical composition, molecular structures, seismic activity, machine data, and others to determine the best possible location to drill [5] [31]. This helps to optimize selection of drilling location, and avoid the cost and effort of unsuccessful drills.

Health care is one of the sectors benefiting from applying Big data prescriptive analytics. Prescriptive analytics can recommend diagnoses and treatments to a doctor by analyzing patient's medical history, similar conditioned patient's history, allergies, medicines, environmental conditions, stage of cure, etc. According to [7], the Aurora Health Care Center saves six million USD annually by using Big data analytics and recommending best possible treatment to doctors.

3 Big Data Analytics Platforms

There are several Big data analytics platforms available. In this section, we discuss recent advances in Big data analytics platforms.

3.1 MapReduce

MapReduce framework represents a pioneering schema for performing Big data analytics. It has been designed for a dedicated platform (such as a cluster). There are three implementations of the MapReduce framework. The first implementation with a proprietary license was developed by Google [30]. The other two implementations: Hadoop [43] and Spark [66] are open-source.

The MapReduce works based on two input functions- Map and Reduce. Programmers implement these functions. Each of these functions utilizes the available resources to process Big data in parallel. The MapReduce storage functionality for storing input, intermediate, and output data is supported by distributed file systems,

such as Hadoop Distributed File System (HDFS) [56] and Google File System (GFS), developed specifically for this framework [32].

Every MapReduce workflow contains three subsequent phases that are Map, Shuffle, and Reduce. In the Map phase, the Map function implemented by the user is executed on the input data across the computational resources. The input data are divided into partitions and stored in a distributed file system (DFS). Each Map task works on a partition of data from the distributed file system and produces intermediate data that are stored locally on the worker machines. Then, the intermediate data are used by the Reduce phase.

Distributing the intermediate data across computational resources to perform parallel Reduce processing is termed Shuffling. The distribution of intermediate data is accomplished in an all-to-all manner that includes a communication bottleneck. Once the intermediate data are distributed, the Reduce function is run and the output is produced. It is also possible to have a chain of MapReduce jobs (aka multi-stage MapReduce), such as Yahoo! WebMap [24]. In these jobs, the output of a MapReduce job is the intermediate data for the next MapReduce job.

3.2 Apache Hadoop

Hadoop [43] framework was developed as an open-source product by Yahoo! and widely adopted for Big data analytics by the academic and industrial communities. The main design advantage of Hadoop is its fault-tolerance. In fact, Hadoop has been designed with the assumption of failure as a common issue in distributed systems. Therefore, it is robust against failures commonly occur during different phases of execution.

Hadoop Distributed File System (HDFS) and MapReduce are two main building blocks of Hadoop. The former is the storage core of Hadoop (see Section 4.1 for details). The latter, MapReduce engine, is above the file system and takes care of executing the application by moving binaries to the machines that have the related data.

For the sake of fault-tolerance, HDFS replicates data blocks in different racks; thus, in case of failure in one rack, the whole process would not fail. A Hadoop cluster includes one master node and one or more worker nodes. The master node includes four components namely, JobTracker, TaskTracker, NameNode, and DataNode. The worker node just includes DataNode and TaskTracker. The JobTracker receives user applications and allocates them to available TaskTracker nodes, while considering data-locality. JobTracker assures about the health of TaskTrackers based on regular heartbeats it receives from them. Although Hadoop is robust against failures in a distributed system, its performance is not the best amongst other available tools because of frequent disk accesses [55].

3.3 Spark

Spark is a more recent framework developed at UC Berkeley [66]. It is being used for research and production applications. Spark offers a general-purpose programming interface in the Scala programming language for interactive, in-memory data analytics of large datasets on a cluster.

Spark provides three data abstractions for programming clusters namely, *resilient distributed datasets (RDDs)*, *broadcast variables*, and *accumulators*. RDD is a read-only collection of objects partitioned across a set of machines. It can reconstruct lost partitions or recover in the event of a node failure. RDD uses a restricted shared memory to achieve fault-tolerance. Broadcast variables and accumulators are two restricted types of shared variables. Broadcast variable is a shared object wrapped around a read-only value, which ensures it is only copied to each worker once. Accumulators are shared variables with an *add* operation. Only workers can perform an operation on an accumulator and only users' driver programs can read from it. Eventhough, these abstractions are simple and limited, they can be used to develop several cluster-based applications.

Spark uses master/slave architecture. It has one master instance, which runs a user-defined driver program. At run-time, the driver program launches multiple workers in the cluster, which read data from the shared filesystem (e.g., Hadoop Distributed File System). Workers create RDDs and write partitions on RAM as defined by the driver program. Spark supports RDD transformations (e.g., map, filter) and actions (e.g., count, reduce). Transformations generate new datasets and actions return a value, from the existing dataset.

Spark has proved to be 20X faster than Hadoop for iterative applications, was shown to speed up a real-world data analytics report by 40X, and has been used interactively to scan a 1 TB dataset with 57 seconds latency [65].

3.4 High Performance Computing Cluster

LexisNexis Risk Solutions originally developed High Performance Computing Cluster (HPCC)[5], as a proprietary platform, for processing and analyzing large volumes of data on clusters of commodity servers more than a decade ago. It was turned into an open-source system in 2011. Major components of an HPCC system include a Thor cluster and a Roxie cluster, although the latter is optional. Thor is called the data refinery cluster, which is responsible for extracting, transforming, and loading (ETL), as well as linking and indexing massive data from different sources. Roxie is called the query cluster, which is responsible for delivering data for online queries and online analytical processing (OLAP).

Similar to Hadoop, HPCC also uses a distributed file system to support parallel processing on Big data. However, compared with HDFS, the distributed file system

[5] http://hpccsystems.com

used by HPCC has some significant distinctions. First of all, HPCC uses two types of distributed file systems; one is called Thor DFS that is intended to support Big data ETL in the Thor cluster; the other is called Roxie DFS that is intended to support Big data online queries in the Roxie cluster. Unlike HDFS that is key-value pair based, the Thor DFS is record-oriented, which is flexible enough to support data sets of different formats, such as CSV, XML, fixed or variable length of records, and records with nested structures. Thor DFS distributes a file across all nodes in the Thor cluster with an even number of records for each node. The Roxie DFS uses distributed B+ tree for data indexing to support efficient delivery of data for user queries.

HPCC uses a data-centric, declarative programming language called Enterprise Control Language (ECL) for both data refinery and query delivery. By using ECL, the user specifies what needs to be done on data instead of how to do it. The data transformation in ECL can be specified either locally or globally. Local transformation is carried out on each file part stored in a node of the Thor cluster in a parallel manner; whereas global transformation processes the global data file across all nodes of the Thor cluster. Therefore, HPCC not only pioneers the current Big data computing paradigm that moves computing to where the data is, but also maintains the capability of processing data in a global scope. ECL programs can be extended with C++ libraries and compiled into optimized C++ code. A performance comparison of HPCC with Hadoop shows that, on a test cluster with 400 processing nodes, HPCC is 3.95 faster than Hadoop on the Terabyte Sort benchmark test [51]. One of the authors of this chapter is currently conducting a more extensive performance comparison of HPCC and Hadoop on a variety of Big data analysis algorithms. More technical details on HPCC can be found in [19] [10] [50] [51].

4 Distributed Data Management Systems for Big Data Analytics

As we discussed earlier in this chapter, huge volumes and a variety of data create a need for special types of data storage. In this section, we discuss recent advances in storage systems for Big data analytics and some commercially available cloud-based storage services.

4.1 Hadoop Distributed File System

The Hadoop Distributed File System (HDFS)[6] is a distributed file system designed to run reliably and to scale on commodity hardware. HDFS achieves high fault-tolerance by dividing data into smaller chunks and replicating them across several

[6] http://hadoop.apache.org

nodes in a cluster. It can scale up to 200 PB in data, and 4500 machines in single cluster. HDFS is a side-project of Hadoop and works closely with it.

HDFS is designed to work efficiently in batch mode, rather than in interactive mode. Characteristics of typical applications developed for HDFS, such as write once and read multiple times, and simple and coherent data access, increases the throughput. HDFS is designed to handle large file sizes from Gigabytes to a few Terabytes.

HDFS follows the master/slave architecture with one NameNode and multiple DataNodes. NameNode is responsible for managing the file system's meta data and handling requests from applications. DataNodes physically hold the data. Typically, every node in the cluster has one DataNode. Every file stored in HDFS is divided into blocks with default block size of 64 MB. For the sake of fault-tolerance, every block is replicated user-defined number of times (recommended to be a minimum of 3 times) and distributed across different data nodes. All meta data about replication and distribution of the file are stored in the NameNode. Each DataNode sends a heartbeat signal to NameNode. If it fails to do so, the NameNode marks the DataNode as failed.

HDFS maintains a Secondary NameNode, which is periodically updated with information from NameNode. In case of NameNode failure, HDFS restores a NameNode with information from the Secondary NameNode, which ensures fault-tolerance of the NameNode. HDFS has a built-in balancer feature, which ensures uniform data distribution across the cluster, and re-replication of missing blocks to maintain the correct number of replications.

4.2 NoSQL Databases

Conventionally, Relational Database Management Systems (RDBMS) are used to manage large datasets and handle tons of requests securely and reliably. Built-in features, such as data integrity, security, fault-tolerance, and ACID (atomicity, consistency, isolation, and durability) have made RDBMS a go-to data management technology for organizations and enterprises. In spite of RDBMS' advantages, it is either not viable or is too expensive for applications that deal with Big data. This has made organizations to adopt a special type of database called "NoSQL" (Not an SQL), which means database systems that do not employ traditional "SQL" or adopt the constraints of the relational database model. NoSQL databases cannot provide all strong built-in features of RDBMS. Instead, they are more focused on faster read/write access to support ever-growing data.

According to December 2014 statistics from Facebook [6], it has 890 Million average daily active users sharing billions of messages and posts every day. In order to handle huge volumes and a variety of data, Facebook uses a Key-Value database system with memory cache technology that can handle billions of read/write requests. At any given point in time, it can efficiently store and access trillions of items. Such operations are very expensive in relational database management systems.

Scalability is another feature in NoSQL databases, attracting large number of organizations. NoSQL databases are able to distribute data among different nodes within a cluster or across different clusters. This helps to avoid capital expenditure on specialized systems, since clusters can be built with commodity computers.

Unlike relational databases, NoSQL systems have not been standardized and features vary from one system to another. Many NoSQL databases trade-off ACID properties in favor of high performance, scalability, and faster store and retrieve operations. Enumerations of such NoSQL databases tend to vary, but they are typically categorized as Key-Value databases, Document databases, Wide Column databases and Graph databases. Figure 2 shows a hierarchical view of NoSQL types, with two examples of each type.

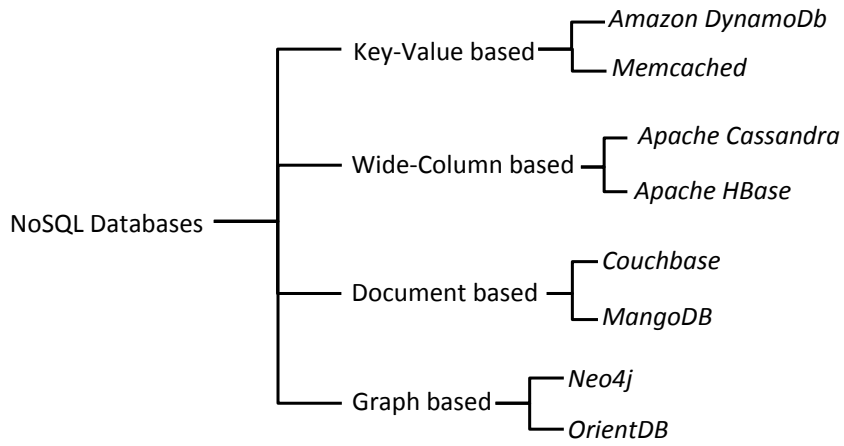

Fig. 2 Categorization of NoSQL databases: The first level in the hierarchy is the categorization of NoSQL. Second level, provides examples for each NoSQL database type.

4.2.1 Key-Value Database

As the name suggests, Key-Value databases store data as Key-Value pairs, which makes them schema-free systems. In most of Key-Value databases, the key is functionally generated by the system, while the value can be of any data type from a character to a large binary object. Keys are typically stored in hash tables by hashing each key to a unique index.

All the keys are logically grouped, eventhough data values are not physically grouped. The logical group is referred to as a *'bucket'*. Data can only be accessed with both a bucket and a key value because the unique index is hashed using the bucket and key value. The indexing mechanism increases the performance of storing, retrieving, and querying large datasets.

There are more than 40 Key-Value systems available with either commercial or open-source licenses. Amazon's DynamoDB[7], which is a commercial data storage system, and open-source systems like Memcached[8], Riak[9], and Redis[10] are most popular examples of Key-Value database systems available.These systems differ widely in functionality and performance.

Key-Value databases are appropriate for applications that require one to store or cache unstructured data for frequent and long-term usages, such as chat applications, and social networks. Key-Value databases can also be used in applications that require real-time responses that need to store and retrieve data using primary keys, and do not need complex queries. In consumer-faced web applications with high traffic, Key-Value systems can efficiently manage sessions, configurations, and personal preferences.

4.2.2 Wide Column Database

A column-based NoSQL database management system is an advancement over a Key-Value system and is referred to as a Wide Column or column-family database. Unlike the conventional row-centric relational systems [34], Wide Column databases are column-centric. In row-centric RDBMS, different rows are physically stored in different places. In contrast, column-centric NoSQL databases store all corresponding data in continuous disk blocks, which speeds up column-centric operations, such as aggregation operations. Eventhough Wide Column is an advancement over Key-Value systems, it still uses Key-Value storage in a hierarchical pattern.

In a Wide-Column NoSQL database, data are stored as name and value pairs, rather than as rows, which are known as columns. Logical grouping of columns is named as column-family. Usually the name of a column is a string, but the value can be of any data type and size (character or large binary file). Each column contains timestamp information along with a unique name and value. This timestamp is helpful to keep track of versions of that column. In a Wide-Column database, the schema can be changed at any time by simply adding new columns to column-families. All these flexibilities in the column-based NoSQL Systems are appropriate to store sparse, distributed, multidimensional, or heterogeneous data. A Wide Column database is appropriate for highly scalable applications, which require built-in versioning and high-speed read/write operations. Apache Cassandra[11] (Originated by Facebook) and Apache HBase[12] are the most widely used Wide Column databases.

---

[7] http://aws.amazon.com/dynamodb/

[8] http://memcached.org/

[9] http://basho.com/riak/

[10] http://redis.io/

[11] http://cassandra.apache.org/

[12] http://hbase.apache.org/

4.2.3 Document Database

A Document database works in a similar way as Wide Column databases, except that it has more complex and deeper nesting format. It also follows the Key-Value storage paradigm. However, every value is stored as a document in JSON [13], XML [14] or other commonly used formats. Unlike Wide Column databases, the structure of each record in a Document database can vary from other records. In Document databases, a new field can be added at anytime without worrying about the schema. Because data/value is stored as a document, it is easier to distribute and maintain data locality. One of the disadvantages of a Document database is that it needs to load a lot of data, even to update a single value in a record. Document databases have built-in approach of updating a document, while retaining all old versions of the document. Most Document database systems use secondary indexing [37] to index values and documents in order to obtain faster data access and to support query mechanisms. Some of the database systems offer full-text search libraries and services for real-time responses.

One of the major functional advantages of document databases is the way it interfaces with applications. Most of the document database systems use JavaScript (JS) as a native scripting language because it stores data in JS friendly JSON format. Features such as JS support, ability to access documents by unique URLs, and ability to organize and store unstructured data efficiently, make Document databases popular in web-based applications. Documents databases serve a wide range of web applications, including blog engines, mobile web applications, chat applications, and social media clients.

Couchbase[15] and MongoDB[16] are among popular document-style databases. There are over 30 document databases. Most of these systems differ in the way data are distributed (both partition and replications), and in the way a client accesses the system. Some systems can even support transactions [35].

4.2.4 Graph Databases

All NoSQL databases partition or distribute data in such a way that all the data are available in one place for any given operation. However, they fail to consider the relationship between different items of information. Additionally, most of these systems are capable of performing only one-dimensional aggregation at a time.

A Graph database is a special type of database that is ideal for storing and handling relationship between data. As the name implies Graph databases use a graph data model. The vertices of a graph represent entities in the data and the edges represent relationships between entities. Graph data model, perfectly fits for scaling

[13] http://json.org

[14] http://www.w3.org/TR/2006/REC-xml11-20060816/

[15] http://couchbase.com/

[16] http://mangodb.org/

out and distributing across different nodes. Common analytical queries in Graph databases include finding the shortest path between two vertices, identifying clusters, and community detection.

Social graphs, World Wide Web, and the Semantic Web are few well-known use-cases for graph data models and Graph databases. In a social graph, entities like friends, followers, endorsements, messages, and responses are accommodated in a graph database, along with relationships between them. In addition to maintaining relationships, Graph databases make it easy to add new edges or remove existing edges. Graph databases also support the exploration of time-evolving graphs by keeping track of changes in properties of edges and vertices using time stamping.

There are over 30 graph database systems. Neo4j [17] and Orient DB[18] are popular examples of graph-based systems. Graph databases found their way into different domains, such as social media analysis (e.g., finding most influential people), e-commerce (e.g., developing recommendations system), and biomedicine (e.g., to analyze and predict interactions between proteins). Graph databases also serve in several industries, including airlines, freight companies, healthcare, retail, gaming, and oil and gas exploration.

4.2.5 Cloud-Based NoSQL Database Services

Amazon DynamoDB: DynamoDB[19] is a reliable and fully managed NoSQL data service, which is a part of Amazon Web Services (AWS). It is a Key-Value database that provides a schema-free architecture to support ever-growing Big data in organizations and real-time web applications. DynamoDB is well optimized to handle huge volume of data with high efficiency and throughput. This system can scale and distribute data, virtually, without any limit. DynamoDB partitions data using a hashing method and replicates data three times and distributes them among data centers in different regions in order to enable high availability and fault-tolerance. DynamoDB automatically partitions and re-partitions data depending on data throughput and volume demands. DynamoDB is able to handle unpredictable workloads and high volume demands efficiently and automatically.

DynamoDB offers eventual and strong consistency for read operations. Eventual consistency does not always guarantee that a data read is the latest written version of the data, but significantly increases the read throughput. Strong consistency guarantees that values read are the latest values after all write operations. DynamoDB allows the user to specify a consistency level for every read operation. DynamoDB also offers secondary indexing (i.e., local secondary and global secondary), along with the indexing of the primary key for faster retrieval.

---

[17] http://neo4j.org/

[18] http://www.orientechnologies.com/orientdb/

[19] http://aws.amazon.com/dynamodb/

DynamoDB is a cost efficient and highly scalable NoSQL database service from Amazon. It offers benefits such as reduced administrative supervision, virtually unlimited data throughput, and the handling of all the workloads seamlessly.

Google BigQuery: Google uses massively parallel query system called as 'Dremel' to query very large datasets in seconds. According to [54], Dremel can scan 35 billion rows in ten seconds even without indexing. This is significantly more efficient than querying a Relational DBMS. For example, on Wikipedia dataset with 314 million rows, Dremel took 10 seconds to execute regular expression query to find the number of articles in Wikipedia that include a numerical character in the title [54]. Google is using Dremel in web crawling, Android Market, Maps, and Books services.

Google brought core features of this massive querying system to consumers as a cloud-based service called 'BigQuery'[20]. Third party consumers can access BigQuery through either a web-based user interface, command-line or through their own applications using the REST API. In order to use BigQuery features, data has to be transferred into the Google Cloud storage in JSON encoding. The BigQuery also returns results in JSON format.

Along with an interactive and fast query system, Google cloud platform also provides automatic data replication, on-demand scalability, and handles software and hardware failure without administrative burdens. In 2014, using BigQuery, scanning one terabyte of data only cost $5, with additional cost for storage [21].

Windows Azure Tables: Windows Azure Tables[22] is a NoSQL database technology with a Key-Value store on the Windows Azure platform. Azure Tables also provides, virtually, unlimited storage of data. Azure Tables is highly scalable and supports automatic partitioning. This database system distributes data across multiple machines efficiently to provide high data throughput and to support higher workloads. Azure Tables storage provides the user with options to select a Partition-Key and a Row-Key upfront, which may later be used for automatic data partitioning. Azure Tables follows only the strong consistency data model for reading data. Azure Tables replicates data three times among data centers in the same region and additional three times in other regions to provide a high degree of fault-tolerance.

Azure Tables is a storage service for applications with huge volume of data, and needs schema-free NoSQL databases. Azure Tables uses primary key alone and it does not support secondary indexes. Azure Tables provides the REST-based API to interact with its services.

[20] http://cloud.google.com/bigquery/

[21] https://cloud.google.com/bigquery/pricing

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

5 Examples of Massive Data Applications

In this section, a detailed discussion of solutions proposed by our research team for two real-world Big data problems are presented.

5.1 Recommendations in e-Commerce

Recommender systems are gaining wide popularity in e-commerce, as they are becoming major drivers of incremental business value and user satisfaction [40] [38]. In this section, we will describe the architecture behind a recommendation engine for eBay, a large open marketplace [39]. In an e-commerce system, there are two major kinds of recommendation scenarios: pre-purchase and post-purchase.

In the pre-purchase scenario, the system recommends items that are good alternatives for the item the user is viewing. In the post-purchase scenario, the recommendation system recommends items complementary or related to an item, which the user has bought recently.

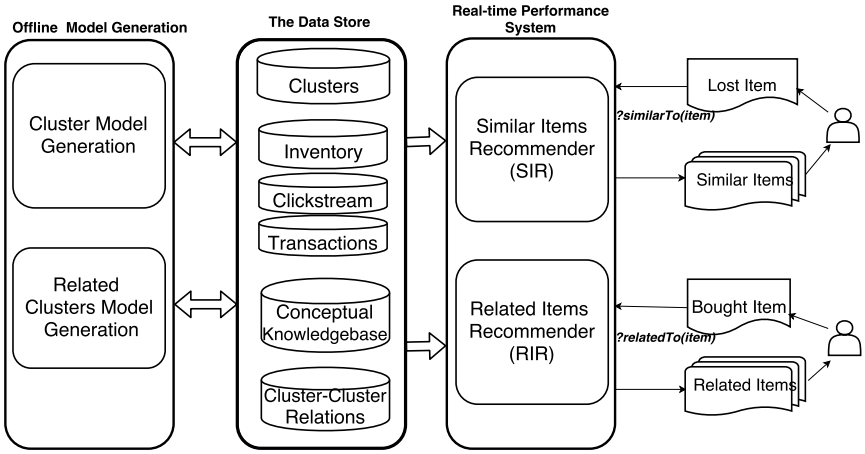

Fig. 3 The recommendation system architecture with three major groups: The offline model; The Data Store; The Real-time Performance System.

5.1.1 Architecture

The recommendation architecture consists of a number of components that can be partitioned in three major groups: The Data Store contains data about the active and temporarily changing state of the web site as well as models learned using that infor-

mation, the Real-time Performance System generates recommendations at the site given a session state and information in the Data Store, and the offline model generation component creates models by conducting computationally intensive offline analyses. Next we describe these components in more detail.

Data Store: The Data Store is the glue between the computationally intensive Offline Model Generation and the Real-time Performance components. Since both systems use the information in the Data Store, it often provides two versions of similar services. For example, the Offline Modeling component has access to the complete history of changes in the item inventory but does not provide efficient keyword search on item properties, while the Real-time Performance System can search an indexed version of the current inventory but does not have access to changes over time. Next, we describe, the Data Sources that our system uses as input and output.

- *Input Information Sources:* The Data Store contains two kinds of information sources: continuously accumulated raw data about the individual actions along with the resulting state on the website, and models, which contain generalized knowledge that can apply to new situations.Like in many e-commerce sites, our Data Store can be categorized into inventory data, which contains a set of items and their static attributes; clickstream data, which stores actions and dynamic state of the site; and transaction data, which store a history of purchases. Note that, while transactions could be recreated from clickstream data, it is typically stored separately for efficient access. The Data Store also contains a conceptual knowledge base including category tree, which is a hierarchical ontology that organizes items contained in the inventory, language-specific knowledge sources like stop words, spell correction rules, etc., and term dictionary that lists important terms/phrases in a given category. Some of these knowledge sources are learned from raw data, but we will not describe this process here due to space limitations.
- *Output Cluster Model:* This architecture generates two main output models. The first model contains cluster definitions, which groups a set of conceptually similar items. One of the most important architectural commitments of our approach is to represent clusters, as an explicit bag of phrases. One major advantage of this representation is that they can be used as search queries. For example, given a search engine indexing an inventory of items, we can retrieve an ordered set of items that best match a cluster expression. This representation also allows us to calculate term similarity and item-coverage overlap between clusters.
  The second knowledge structure our system generates is the Related Cluster model, which is used for post-purchase recommendations of complimentary items. This model represents a sparse graph between clusters. A strong link from one cluster to another indicates that the likelihood of purchase of an item from the second cluster increases after a purchase of an item from the first cluster. Next, we describe how the cluster models are used in the runtime system and how they are learned during offline processing.

Real-time Performance System: The Real-time Performance System consists of two components, Similar Items Recommender (SIR) and Related Items Recom-

mender (RIR). Both of these components take a seed item as input, and return a set of items that are similar or related to that seed item. Since the Real-time Performance System is interacting with the user in real-time, it is crucial that any relevant complicated decision process is compiled to the offline model. However, pre-caching all processing is also not feasible since under the changing inventory assumption we are making, the seed item can be an item the Real-time Performance System encounters the first time. As a result, we require that any retrieval from a data source occur through an efficient indexing service and the computation done after retrieval is kept limited.

Both SIR and RIR start the recommendation process by calling a cluster assignment service that returns the best matching clusters given an item. To achieve that, the service compiles normalized versions of cluster expressions in a Lucene[23] index and runs the input item title and features (e.g., category, attribute-value pair) through a similar normalization to return the best matching clusters. SIR shows few best items returned from the search service as its recommendations. On the other hand, RIR returns one item per the query it has constructed to ensure that each recommendation is related to the seed item in a different way.

Offline Model Generation:

- *Clusters Generation:* An online marketplace like eBay has a huge inventory of transient items ranging in the hundreds of millions. Moreover, the inventory covers a very broad spectrum of variety, ranging from regular electronic items to unique collectible items. Given the scale and variedness of the inventory, global clustering is not feasible. The system uses parallel clustering of inventory on a Hadoop Map-Reduce cluster.
- *Cluster-Cluster Relations Generation:* The cluster-related cluster pairs are computed using the transactional data. From the transactional data set, an item-to-item co-purchase matrix is built. Cluster-related clusters are generated using the item-item co-purchase matrix on a Hadoop Map-Reduce cluster.

5.1.2 Experimental results

We conducted A/B tests to compare the performance of our Similar and Related Items Recommender systems described in this section over the legacy recommendation system developed by Chen & Canny [28]. The legacy system clusters the items using generative clustering and later it uses a probabilistic model to learn relationship patterns from the transaction data. One of the main differences, is the way these two recommendation system generate the clusters. The legacy system uses item data (auction title, description, price), whereas our system uses user queries to generate clusters.

A test was conducted on Closed View Item Page (CVIP) in eBay to compare our Similar Items Recommender algorithm with the legacy algorithm. CVIP is a page that is used to engage a user by recommending similar items after an unsuccess-

[23] http://lucene.apache.org/

ful bid. We also conducted a test to compare our Related Items Recommender with legacy algorithm [28]. Both the test results show significant improvement in user engagement and site-wide business metrics with 90% confidence. As we are not permitted to publish actual figures representing system performances, we are reporting relative statistics. Relative improvements in user engagement (Click Through Rate) with our SIR and RIR, over legacy algorithms, are 38.18% and 10.5%, respectively.

5.2 Link Prediction in Biomedical Literature

Predicting the likelihood of two nodes associating in the future, which do not have direct association between them in the current timestep, is known as the link prediction problem. Link prediction is widely used in social network analysis. Link prediction has wide range of applications such as identifying missing information, identifying spurious interactions, and studying the evolution of the network. In e-commerce, link prediction is used for building recommendation systems, and in bio-informatics, it is used to predict protein-protein interactions.

Katukuri at el., [41], proposed a supervised link prediction method, to predict unknown association of medical concepts using bio-medical publication information from Medline[24]. Medline is a National Institute of Health (NIH)'s citation database with more than 21 million publication citations. Figure 4 illustrates different stages in the proposed supervised link prediction approach. A temporal concept network is generated using relevant medical concepts extracted from publications. In the concept network, each node represents a medical concept and an edge between two nodes represents relationship that two medical concepts co-occurred at least in one publication. Document frequency of a given concept is a weight of node and co-occurrence frequency of two concepts is edge weight. Now, link prediction problem is formulated as a process of identifying whether a pair of concepts, which are not directly connected in the current duration concept network, will be connected directly in the future.

This link prediction problem is formulated as a supervised classification task. Training data is automatically labeled by comparing concept network snapshots of two consecutive time periods. This automatic labeling approach helps to avoid need for domain experts.

In automatic labeling method, concept pairs, which are not directly connected in the first snapshot, are labeled based on its possible connection strength in the second snapshot. Connections strength is categorized as follows (S is edge weight in second snapshot, and minimum_support and margin (ranges between 0 to 1) are user-defined values):

- Connection as strong: S $\geq$ minimum_support
- Connection as emerging: $margin \times minimum\_support \leq$ S $<$ minimum_support.
- Connection as weak: S $< margin \times minimum\_support$.

[24] http://www.ncbi.nlm.nih.gov/pubmed/

- No connection: S=0.

Given a pair of nodes that has no direct connection in first snapshot is assigned with positive class label if this pair is strongly connected in the second snapshot and is assigned negative class label if it has weak connection or no connection the second snapshot, and the pairs with intermediate values of strength are labeled as emerging.

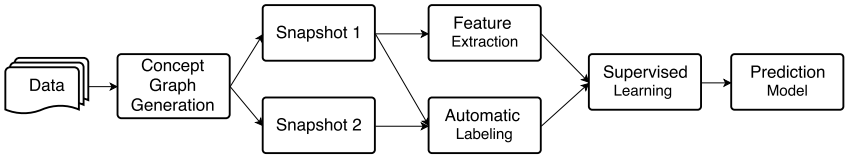

Fig. 4 Implementation flow diagram for supervised link prediction in biomedical literature

For each of labeled concept pairs, a set of topological features (random-walk based and neighborhood-based) is extracted from the first snapshot of the concept network. The topological feature set also includes Common neighbors, Adamic/Adar, Jaccard Co-efficient, and Preferential Attachment [36]. Along with topological features, semantically-enriched features, like Semantic CFEC [41] are extracted. Combining labeled concept pairs with the corresponding feature set generates the training instances. Supervised classification algorithms, such as SVM, and C4.5 decision tree are used to generate prediction models. The classification accuracy of prediction models is calculated using cross-validation, which is on average 72% [41].

Implementing such a computationally intensive phase to extract features needed for generating the predictive model on massive data needs large computational resources. For example, a snapshot of the concept network for years 1991-2010 has nearly 0.2 million nodes and nearly 44 million edges. Processing millions of publications to extract medical concepts, generating class labels and extracting features from large well-connected network is computationally heavy. To handle such computations on large graphs, the prediction system is developed by using the MapReduce framework and a Hadoop cluster environment. We implemented the MapReduce fuctions to extract the medical concepts from millions of publications in Medline dataset, to generate the labeled data, and to extract structual features from the large concept graph.

Implementing such a graph computation method on MapReduce framework has its own limitations. One of the drawbacks of MapReduce framework is its inability to retain the state of a graph across multiple iterations [17]. One approach to retain the state in a MapReduce pipeline is by explicitly writing the graph to disk after one iteration and reading it back from disk in the next iteration. This approach proves to be inefficient due to the huge increase in I/O operation and bandwidth [17].

Google proposed a Bulk Synchronous Parallel processing model called Pregel [47], which is a message passing model. Unlike MapReduce framework, this model helps by retaining the graph state across the iterations. Apache Giraph is open source

alternative to Pregel, which is built on top of MapReduce framework. In such distributed graph systems, a graph is partitioned and distributed among different cluster nodes. Each vertex has information about itself along with its neighbors. In our link prediction, features like the Jaccard Coefficient, can be extracted in parallel since such calculations depend only on information local to each vertex. However, other features, like the Semantic CFEC, need to be calculated by exploring all the paths between given pair of nodes, which can be formulated as an all-pairs-path problem. There are several frameworks [47], [46] that can calculate the all-pairs-path problem by passing information through edges, but between just a pair of nodes at a time. However, these frameworks cannot support the operation of finding all paths between all pairs in parallel. In our case, there is a need to extract such features for millions of concept pairs. To the best of our knowledge, there is no algorithm or framework that can support such a problem to run in a distributed environment. This is one of the open computational challenges in graph analytics that needs to be investigated by the research community.

6 Current Issues of Big Data Analytics

In this section, we discuss several open challenges relating to computation, storage, and security in Big data analytics.

6.1 Data Locality

One prominent aspect in efficient Big data processing is the ability to access data without a significant latency. Given the transfer-prohibitive volume of Big data, accessing data with low latency can be accomplished only through data locality. In fact, data movement is possible in computations with moderate to medium volume of data where the data transfer to processing time ratio is low. However, this is not the case for Big data analytics applications. The alternative approach to alleviate the data transfer problem is moving the computation to where the data resides. Thus, efficient data management policies are required in the Big data analytics platforms to consider issues such as maximizing data locality and minimizing data migration (i.e., data transfer) between cloud data centers [64].

One of the key features of the Hadoop framework is its ability to take the effects of data locality into account. In Hadoop, the JobTracker component tries to allocate jobs to nodes where the data exists. Nonetheless, there are cases in which all the nodes that host a particular data node are overloaded. In this situation, JobTracker has to schedule the jobs on machines that do not have the data.

To expedite data processing, Spark keeps the data in main memory, instead of on disk. Spark's data locality policy is similar to Hadoop. However, in Spark, the Reduce tasks are allocated to machines where the largest Map outputs are gener-

ated. This reduces data movement across the cluster and improves the data locality further.

6.2 Fault-Tolerance of Big Data Applications

In long-running, Big data analytics applications, machine failure is inevitable. Both transient (i.e., fail-recovery) and permanent (i.e., fail-stop) failures can occur during the execution of such applications [42]. Google reports experiencing on average, 5 machine crashes during a MapReduce job in March 2006 [29] and at a minimum one disk failure in each execution of a MapReduce job with 4000 tasks. Because of the criticality of failure, any resource allocation method for Big data jobs should be fault-tolerant.

MapReduce was originally designed to be robust against faults that commonly happen at large-scale resource providers with many computers and devices such as network switches and routers. For instance, reports show that during the first year of a cluster operation at Google there were 1000 individual machine failures and thousands of hard-drive failures.

MapReduce uses logs to tolerate faults. For this purpose, the output of Map and Reduce phases create logs on the disk [49]. In the event that a Map task fails, it is re-executed with the same partition of data. In case of failure in Reducer, the key/value pairs for that failed Reducer are re-generated.

6.3 Replication in Big Data

Big data jobs either do not replicate the data or do this automatically through distributed file system (DFS). Without replication, the failure of a server storing the data causes the re-execution of the affected tasks. Though, the latter approach offers more fault-tolerance, it is not efficient due to network overhead and increase in the execution time of the job.

In Hadoop, users can manually determine the number of replicas that they wish to have created. Such a static replication approach imposes a remarkable storage overhead to the underlying system and slows down the job execution. A solution to this problem is to adjust the replication rate based on the usage rate of the data. This will reduce the storage and processing costs on the resources [63]. Cost-effective incremental replication [44] is a method, for cloud-based jobs, that is capable of predicting when a job needs to be replicated.

There are four major data-replication schemes for Big data on clouds namely, synchronous and asynchronous replication, rack-level replication, and selective replication. These replication schemes can be applied on input, intermediate, or output data of a job.

In synchronous data replication (e.g., HDFS), producer tasks in a job are blocked until replication finishes. Synchronous replication leads to a high consistency because, if a producer of block A returns, all the replicas of block A are guaranteed to be identical and any consumer tasks of block A can read any replica. Nonetheless, the drawback of this approach is that the performance of producers can get affected, as they have to be blocked. In asynchronous data replication [42] producers proceed without waiting for a replication to complete. The asynchronous data replication consistency is not as precise as the synchronous scheme because even if a producer of block A returns, a replica of block A may still perform replication. However, performance of the producer improves due to the non-blocking nature of this scheme. When asynchronous replication is used in Hadoop, Map and Reduce tasks can continue without being blocked.

Rack-level replication enforces replication on the same rack in a data center. In data centers, servers are structured in racks with a hierarchical topology. A two-level architecture, with a rack-level switch and a central switch, is a common topology in these data centers. In this topology, the central switch can become the bottleneck as many racks share it. This means that, there is heterogeneity in network bandwidth where inter-rack bandwidth is scarce relative to intra-rack bandwidth. One instance of bandwidth bottleneck is in the Shuffling step of MapReduce. In this case, the central switch becomes over-utilized whereas rack-level switches are underutilized. Rack-level replication reduces the traffic that goes through the central switch. However, the rack-level replication cannot tolerate rack-level failures. According to recent studies, rack-level failures are infrequent which supports the adoption of rack-level replication.

In selective replication, the data generated by the previous step of the Big data job are replicated on the same server, in which they are generated. For example, in a chained MapReduce, once there is a failure at the Map phase, the affected Map tasks can be restarted instantly, if the data generated by the previous Reduce tasks were replicated on the same machine. Therefore, the intermediate data that needs to be replicated in the Map phase is reduced. However, it is not very effective for the Reduce step, because Reduce data are mostly consumed locally.

As mentioned earlier, replicating input data or intermediate data on distributed file systems is expensive for Big data jobs. The overhead is due to data replication, disk I/O, network bandwidth, and serialization, which can potentially dominate the job execution time [65]. To avoid these overheads, in Pregel [47], a system for iterative graph computation, intermediate data are kept in memory. Resilient Distributed Datasets (RDDs) [65] are parallel data structures within the Spark [66] framework that enable users to keep intermediate data in memory and manipulate them using various operators. They also control the partitioning of the data to optimize data placement.

6.4 Big Data Security

In spite of the advantages offered by Big data analytics on clouds and the idea of Analytics as a Service, there is an increasing concern over the confidentiality of the Big data in these environments [25]. This concern is more serious as increaseing amount of confidential user data are migrated to the cloud for processing. Genome sequences, health information, and feeds from social networks are few instances of such data.

A proven solution to the confidentiality concerns of sensitive data on cloud is to employ user-side cryptographic techniques for securing the data [25]. However, such techniques limit the cloud-based Big data analytics in several aspects. One limitation is that the cryptographic techniques usually are not transparent to end-users. More importantly, these techniques restrict functionalities, such as searching and processing, that can be performed on the users' data. Numerous research works are being undertaken to address these limitations and enable seamless Big encrypted data analytics on the cloud. However, all of these efforts are still in their infancy and not applicable to Big data scale processing.

Another approach to increase data confidentiality is to utilize multiple cloud storage units simultaneously [61]. In this approach, user data are sharded based on a user-side hashing function, and then the data for each cloud is encrypted and uploaded across multiple clouds. It is noteworthy that sharding and distributing of the data are achieved based on some lightweight user-side processing. Therefore, the control and distribution of the data is determined merely at the user-side. Although such sharding approach seems interesting for Big data analytics, challenges, such as processing sharded data across multiple clouds, still remains unsolved.

Hybrid clouds have proven to be helpful in increasing the security of Big data analytics. In particular, they can be useful for cloud-based Big data analytics where a portion of the data is sensitive and needs specific trusted resources for execution. One approach is to label the data and treat them differently based on their labels [67]. As such, non-sensitive data are pushed to a public cloud for processing and sensitive data are processed in a private cloud. The coordination is accomplished through a scheduler placed within local resources that determines where a data should be processed depending on its label.

6.5 Data Heterogeneity

One of the major challenges researchers are facing is "How to integrate all the data from different sources to maximize the value of data". In the World Wide Web (WWW), there is a huge amount of data created by social network sites, blogs, and websites. However, every source is different in data structure, semantics, and format. Structure of data from these sources varies from well-structured data (e.g., databases) to unstructured data (e.g., heterogeneous documents).

Vivek Singh at el., [57], developed a framework to detect situations (such as epidemics, traffic jams) by combining information from different streams like Twitter, Google Insights, and satellite imagery. In this framework, heterogeneous real-time data streams are combined by converting selected attributes and unified across all streams. There are several other proposed frameworks that can combine different data sources for various chosen domain-specific applications. Most of these solutions use a semantics-based approach. Ontology matching is a popular semantics-based method, which finds the similarity between the ontologies of different sources.

Ontology is a vocabulary and description of a concept and its relationship with others in the respective domain. In the Web example, ontology is used to transform unstructured or partially structured data from different sources. Most of them are human readable format (e.g., HTML) and are hard for the machine to understand. One of the most successful ontology integration projects is Wikipedia, which is essenstially integrated with human intervention. Semantic Web tools are used to convert the unstructured Web to a machine understandable structure. Semantic Web adds ontology constructs into web pages and enables machines to understand the contents of a webpage. It helps to automate integration of heterogeneous data from different sources on the Web using ontology matching.

7 Summary and Discussion

Size of data in the digital universe is almost doubling every year and has reached to a stage that they cannot be processed by the conventional programming, computing, storage, visualization, and analytical tools. In this study, we reviewed different types of analytic needs arising in research and industry. We broadly categorized the current analytical applications as descriptive, predictive, and prescriptive and identified several real-world applications of each type.Then, we provided an overview of the state-of-the-art on platforms, tools, and use cases for massive data analytics.

In particular, we discussed that cloud services are helpful platforms in alleviating many of the massive data processing challenges. MapReduce compute resources, NoSQL databases, virtually unlimited storages, and customized filesystems, amongst many others, are useful cloud services for massive data analytics.

We provided two use-cases that were investigated within our research team. The first one, recommendation in e-commerce applications, consists of a number of components that can be partitioned into three major groups: *The Data Store* that contains data about the active and temporarily changing state of an e-commerce web site; *The Real-time Performance System* that generates recommendations in real-time based on the information in the Data Store; and the *offline model generation* that conducts computationally intensive offline analyses. The Real-time Performance System consists of two components, similar items recommender (SIR) and related items recommender (RIR). Both of these components take a seed item as input, and return a set of items that are similar or related to that seed item.

The second use case addresses the problem of link prediction that proposes associations between medical concepts that did not exist in earlier published works. In this project, we model biomedical literature as a concept network, where each node represents a biomedical concept that belongs to a certain semantic type, and each edge represents a relationship between two concepts. Each edge is attached with a weight that reflects the significance of the edge. Based on the constructed massive graph, a machine-learning engine is deployed to predict the possible connection between two indirectly connected concepts.

In the course of our research on massive data analytics tools and projects, we have learnt key lessons and identified open challenges that have to be addressed by researchers to further advance efficient massive data analytics. Below, we highlight some of the lessons and challenges:

- In many analytical applications (e.g., recommendation system in e-commerce), even with availability of state-of-the-art Big data technologies, treating customer data as a data stream is not yet viable. Therefore, some steps (e.g., model building in recommendation systems), have to be performed offline.
- It is difficult, if not impossible, to come up with a generic framework for various types of analytics. For instance, in the recommendation system, which is an example of predictive data analytics in e-commerce, there are many subtle nuances. Thus, a specific architecture is required based on the merits of each application. Accordingly, in the eBay application, we noticed that Related Items Recommendation (RIR) needs a different architecture compared to Similar Items Recommendation (SIR).
- Many Big data analytics (e.g., biomedical link prediction) process massive graphs as their underlying structure. Distributed graph techniques need to be in place for efficient and timely processing of such structures. However, to the best our knowledge, there is not yet a comprehensive distributed graph analytic framework that can support all conventional graph operations (e.g., path-based processing in distributed graphs).
- Data locality and replication management policies ought to be cleverly integrated to provide robust and fault-tolerant massive data analytics.
- As massive data are generally produced from a great variety of sources, novel, semantics-based solutions should be developed to efficiently support data heterogeneity.

References

1. 5 must-have lessons from the 2014 holiday season. http://www.experian.com/blogs/marketing-forward/2015/01/14/five-lessons-from-the-2014-holiday-season/ March 6 2015.
2. 6 uses of big data for online retailers. http://www.practicalecommerce.com/articles/3960-6-Uses-of-Big-Data-for-Online-Retailers Feb 28 2015.
3. Australian square kilometer array pathfinder radio telescope. http://www.atnf.csiro.au/projects/askap/index.html Feb 28 2015.

4. Big data and content analytics: measuring the ROI. http://www.aiim.org/Research-and-Publications/Research/Industry-Watch/Big-Data-2013 Feb 28 2015.
5. Enhancing exploration and production with big data in oil gas. http://www-01.ibm.com/software/data/bigdata/industry-oil.html Feb 28 2015.
6. Facebook. http://newsroom.fb.com/company-info/ March 14 2015.
7. The future of big data? three use cases of prescriptive analytics. https://datafloq.com/read/future-big-data-use-cases-prescriptive-analytics/668 March 02 2015.
8. Gartner survey reveals that 73 percent of organizations have invested or plan to invest in big data in the next two years. http://www.gartner.com/newsroom/id/2848718 Feb 28 2015.
9. Gartner taps predictive analytics as next big business intelligence trend. http://www.enterpriseappstoday.com/business-intelligence/gartner-taps-predictive-analytics-as-next-big-business-intelligence-trend.html Feb 28 2015.
10. HPCC vs Hadoop. http://hpccsystems.com/Why-HPCC/HPCC-vs-Hadoop March 14 2015.
11. IBM netfinity predictive failure analysis. http://ps-2.kev009.com/pccbbs/pc_servers/pfaf.pdf March 14 2015.
12. Instagram. https://instagram.com/press/ Feb 28 2015.
13. The Large Hadron Collider. http://home.web.cern.ch/topics/large-hadron-collider Feb 28 2015.
14. NASA applies text analytics to airline safety. http://data-informed.com/nasa-applies-text-analytics-to-airline-safety/ Feb 28 2015.
15. New IDC worldwide big data technology and services forecast shows market expected to grow to $32.4 billion in 2017. http://www.idc.com/getdoc.jsp?containerId=prUS24542113 Feb 28 2015.
16. The oil gas industry looks to prescriptive analytics to improve exploration and production. https://www.exelisvis.com/Home/NewsUpdates/TabId/170/ArtMID/735/ArticleID/14254/The-Oil–Gas-Industry-Looks-to-Prescriptive-Analytics-To-Improve-Exploration-and-Production.aspx Feb 28 2015.
17. Processing large-scale graph data: A guide to current technology. http://www.ibm.com/developerworks/library/os-giraph/ september 08 2015.
18. Purdue university achieves remarkable results with big data. https://datafloq.com/read/purdue-university-achieves-remarkable-results-with/489 Feb 28 2015.
19. Resources:HPCC systems. http://hpccsystems.com/resources March 14 2015.
20. VC funding trends in big data (IDC report). http://www.experfy.com/blog/vc-funding-trends-big-data-idc-report/ Feb 28 2015.
21. Youtube statistics. http://www.youtube.com/yt/press/statistics.html Feb 28 2015.
22. Descriptive , predictive , prescriptive : Transforming asset and facilities management with analytics. October 2013.
23. Ahmed Abbasi, Conan Albrecht, Anthony Vance, and James Hansen. Metafraud: A meta-learning framework for detecting financial fraud. *MIS Q.*, 36(4):1293–1327, December 2012.
24. Özge Alaçam and Mustafa Dalcı. A usability study of webmaps with eye tracking tool: The effects of iconic representation of information. In *Proceedings of the 13th International Conference on Human-Computer Interaction. Part I: New Trends*, pages 12–21. Springer-Verlag, Sep 2009.
25. Mohsen Amini Salehi, Thomas Caldwell, Alejandro Fernandez, Emmanuel Mickiewicz, David Redberg, Eric W. D. Rozier, and Saman Zonouz. RESeED: Regular Expression Search over Encrypted Data in the Cloud. In *Proceedings of the 7th IEEE Cloud conference*, Cloud '14, pages 673–680, June 2014.
26. Marcos D Assunção, Rodrigo N Calheiros, Silvia Bianchi, Marco AS Netto, and Rajkumar Buyya. Big data computing and clouds: Trends and future directions. *Journal of Parallel and Distributed Computing*, 2014.
27. Wouter Buckinx, Geert Verstraeten, and Dirk Van den Poel. Predicting customer loyalty using the internal transactional database. *Expert systems with applications*, 32(1):125–134, 2007.

28. Ye Chen and John F Canny. Recommending ephemeral items at web scale. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 1013–1022. ACM, 2011.
29. Jeffrey Dean. Experiences with MapReduce, an abstraction for large-scale computation. In *Proceedings of the 15th International Conference on Parallel Architectures and Compilation Techniques*, PACT '06, 2006.
30. Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, Jan 2008.
31. Adam Farris. How big data is changing the oil & gas industry. *Analytics Mag., Dec*, 2012.
32. Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. The google file system. In *Proceedings of the 19th ACM Symposium on Operating Systems Principles*, SOSP '03, pages 29–43, Oct 2003.
33. Venkat N Gudivada, Ricardo Baeza-Yates, and Vijay V Raghavan. Big data: Promises and problems. *Computer*, (3):20–23, 2015.
34. Venkat N Gudivada, Dhana Rao, and Vijay V Raghavan. NoSQL systems for big data management. In *Services (SERVICES), 2014 IEEE World Congress on*, pages 190–197. IEEE, 2014.
35. Venkat N Gudivada, Dhana Rao, and Vijay V Raghavan. Renaissance in data management systems: SQL, NoSQL, and NewSQL. *IEEE Computer*, "In press".
36. MohammadAl Hasan and MohammedJ. Zaki. A survey of link prediction in social networks. In Charu C. Aggarwal, editor, *Social Network Data Analytics*, pages 243–275. Springer US, 2011.
37. M. Indrawan-Santiago. Database research: Are we at a crossroad? reflection on NoSQL. In *Network-Based Information Systems (NBiS), 2012 15th International Conference on*, pages 45–51, Sept 2012.
38. Jayasimha Katukuri, Tolga Konik, Rajyashree Mukherjee, and Santanu Kolay. Recommending similar items in large-scale online marketplaces. In *Big Data (Big Data), 2014 IEEE International Conference on*, pages 868–876. IEEE, 2014.
39. Jayasimha Katukuri, Rajyashree Mukherjee, and Tolga Konik. Large-scale recommendations in a dynamic marketplace. In *Workshop on Large Scale Recommendation Systems at RecSys*, volume 13, 2013.
40. Jayasimha Katukuri, Rajyashree Mukherjee, and Tolga Konik. Subjective similarity: Personalizing alternative item recommendations. In *WWW Workshop: Ad Targeting at Scale*, 2015. accepted.
41. Jayasimha R Katukuri, Ying Xie, Vijay V Raghavan, and Ashish Gupta. Hypotheses generation as supervised link discovery with automated class labeling on large-scale biomedical concept networks. *BMC genomics*, 13(Suppl 3):S5, 2012.
42. Steven Y. Ko, Imranul Hoque, Brian Cho, and Indranil Gupta. Making cloud intermediate data fault-tolerant. In *Proceedings of the 1st ACM Symposium on Cloud Computing*, SoCC '10, pages 181–192, June 2010.
43. Chuck Lam. *Hadoop in Action*. Manning Publications Co., Greenwich, CT, USA, 1st edition, 2010.
44. Wenhao Li, Yun Yang, and Dong Yuan. A novel cost-effective dynamic data replication strategy for reliability in cloud data centres. In *Proceedings of the Ninth IEEE International Conference on Dependable, Autonomic and Secure Computing*, DASC '11, pages 496–502, Oct 2011.
45. Steve Lohr. The age of big data. *New York Times*, 11, 2012.
46. Yucheng Low, Joseph Gonzalez, Aapo Kyrola, Danny Bickson, Carlos Guestrin, and Joseph M Hellerstein. Graphlab: A new framework for parallel machine learning. arxiv preprint. *arXiv preprint arXiv:1006.4990*, 1, 2010.
47. Grzegorz Malewicz, Matthew H. Austern, Aart J.C Bik, James C. Dehnert, Ilan Horn, Naty Leiser, and Grzegorz Czajkowski. Pregel: A system for large-scale graph processing. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, SIGMOD '10, pages 135–146, June 2010.

48. James Manyika, Michael Chui, Brad Brown, Jacques Bughin, Richard Dobbs, Charles Roxburgh, Angela Hung Byers, and McKinsey Global Institute. Big data: The next frontier for innovation, competition, and productivity. 2011.
49. Andre Martin, Thomas Knauth, Stephan Creutz, Diogo Becker, Stefan Weigert, Christof Fetzer, and Andrey Brito. Low-overhead fault tolerance for high-throughput data processing systems. In *Proceedings of the 31st International Conference on Distributed Computing Systems*, ICDCS '11, pages 689–699, May 2011.
50. AnthonyM. Middleton, DavidAlan Bayliss, and Gavin Halliday. ECL/HPCC: A unified approach to big data. In Borko Furht and Armando Escalante, editors, *Handbook of Data Intensive Computing*, pages 59–107. Springer New York, 2011.
51. Author Anthony M Middleton, Ph D Lexisnexis, and Risk Solutions. White Paper HPCC Systems : Data Intensive Supercomputing Solutions. *Solutions*, 2011.
52. Philip Russom et al. Big data analytics. *TDWI Best Practices Report, Fourth Quarter*, 2011.
53. Mohsen Salehi and Rajkumar Buyya. Adapting market-oriented scheduling policies for cloud computing. In *Algorithms and Architectures for Parallel Processing*, volume 6081 of *ICA3PP' 10*, pages 351–362. Springer Berlin / Heidelberg, 2010.
54. Kazunori Sato. An inside look at google bigquery. *White paper, URL: https://cloud. google. com/files/BigQueryTechnicalWP. pdf*, 2012.
55. Avraham Shinnar, David Cunningham, Vijay Saraswat, and Benjamin Herta. M3r: Increased performance for in-memory hadoop jobs. *Proceedings of VLDB Endowment*, 5(12):1736–1747, Aug 2012.
56. Konstantin Shvachko, Hairong Kuang, Sanjay Radia, and Robert Chansler. The hadoop distributed file system. In *Proceedings of the 26th IEEE Symposium on Mass Storage Systems and Technologies*, MSST '10, pages 1–10, 2010.
57. Vivek K. Singh, Mingyan Gao, and Ramesh Jain. Situation recognition: An evolving problem for heterogeneous dynamic big multimedia data. In *Proceedings of the 20th ACM International Conference on Multimedia*, MM '12, pages 1209–1218, New York, NY, USA, 2012. ACM.
58. Mark Troester. Big data meets big data analytics. page 13, 2012.
59. Dirk Van den Poel and Wouter Buckinx. Predicting online-purchasing behaviour. *European Journal of Operational Research*, 166(2):557–575, 2005.
60. David Reinsel Stephen Minton Vernon Turner, John F. Gantz. The Digital Universe of Opportunities: Rich Data and Increasing Value of the Internet of Things. (April), 2014.
61. Jianzong Wang, Weijiao Gong, Peter Varman, and Changsheng Xie. Reducing storage overhead with small write bottleneck avoiding in cloud raid system. In *Proceedings of the 2012 ACM/IEEE 13th International Conference on Grid Computing*, GRID '12, pages 174–183, Washington, DC, USA, 2012. IEEE Computer Society.
62. Seth Wolpin. An exploratory study of an intranet dashboard in a multi-state healthcare system. 2006.
63. Dong Yuan, Lizhen Cui, and Xiao Liu. Cloud data management for scientific workflows: Research issues, methodologies, and state-of-the-art. In *10th International Conference on Semantics, Knowledge and Grids (SKG)*, pages 21–28, Aug 2014.
64. Dong Yuan, Yun Yang, Xiao Liu, and Jinjun Chen. A data placement strategy in scientific cloud workflows. *Future Generation Computer Systems*, 26(8):1200 – 1214, 2010.
65. Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauley, Michael J. Franklin, Scott Shenker, and Ion Stoica. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*, NSDI'12, pages 2–12. USENIX Association, 2012.
66. Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, and Ion Stoica. Spark: Cluster computing with working sets. In *Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing*, HotCloud'10, pages 10–15, June 2010.
67. Chunwang Zhang, Ee-Chien Chang, and R.H.C. Yap. Tagged-MapReduce: A general framework for secure computing with mixed-sensitivity data on hybrid clouds. In *Proceedings of*

14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, pages 31–40, May 2014.