Vietnam J Comput Sci (2014) 1:3–14
DOI 10.1007/s40595-013-0001-6

REGULAR PAPER

Big data as the new enabler in business and other intelligence

Gottfried Vossen

Received: 19 September 2013 / Accepted: 23 September 2013 / Published online: 12 November 2013

Abstract The term “big data” will always be remembered as the big buzzword of 2013. According to the Wikipedia, big data “is a collection of data sets so large and complex that it becomes difficult to process using on-hand database management tools”. In other perceptions, the “3 Vs” that characterize it (i.e., volume, velocity, and variety) or the “4 Vs” (adding veracity to the previous three) are responsible for the fact that it exceeds an organization’s own data as well as its storage or compute capacity for accurate and timely decision-making. In essence, big data refers to the situation that more and more aspects and artifacts of everyday life, be it personal or professional, are available in digital form, e.g., personal or company profiles, social network and blog postings, buying histories, health records, to name just a few, that increasingly more data gets dynamically produced especially on the Internet and on the Web, and that nowadays the tools and techniques are available for evaluating and analyzing all that data in various combinations. Numerous companies already foresee the enormous business effects that analytical scenarios based on big data can have, and the impacts that it will hence have on advertising, commerce, and business intelligence (BI). This paper reviews the issues, techniques, and applications of big data, with an emphasis on future BI architectures.

Keywords Big data · Business intelligence · Business analytics

G. Vossen (✉)
University of Münster, European Research Center for Information Systems (ERCIS), Leonardo-Campus 3, 48149 Münster, Germany
e-mail: vossen@uni-muenster.de; vossen@waikato.ac.nz

G. Vossen
Department of Management Systems, The University of Waikato Management School, Private Bag 3105, Hamilton, New Zealand

1 Introduction

Ever since the beginning of the digital age, data in digital form has received a growing importance, first primarily in the business domain and later also in the private domain. Think back, for example, to the early beginnings of email usage in the late 1970s and early 1980s: it took a while to set up a connection, typically via a slow modem, then type a message on a black and white screen using a line editor, next sending the message off, and finally shutting down the connection again. While the actual data, i.e., the number of characters or bytes making up the message, was small compared to what we can put in an email today, at the time nobody would have believed that the same could at some point be done from an “intelligent” phone with much higher speed and considerably bigger content. For another example, think of early digital cameras and their resolution and compare that to what is currently the standard! But besides digitalization and the fact that digital objects have become larger and larger over time, technology has also enabled faster transportation of data and—thanks to the Web 2.0 developments [26]—both increased automatic as well as human production of data. The result is so overwhelming that the term “big data” seems appropriate; this paper is about the issues, techniques, and applications of big data, with an emphasis on future BI architectures.

In a recent statistics,[1] Intel reported that in a single Internet minute, 639,800 GB of global IP data gets transferred over the Internet, which can be broken down into emails, app downloads, e-commerce sales, music listening, video viewing, or social network status updates, and this number will increase significantly over the next couple of years. This already is representative of one dimension of big data, its volume or

[1] http://www.intel.com/content/www/us/en/communications/internet-minute-infographic.html.

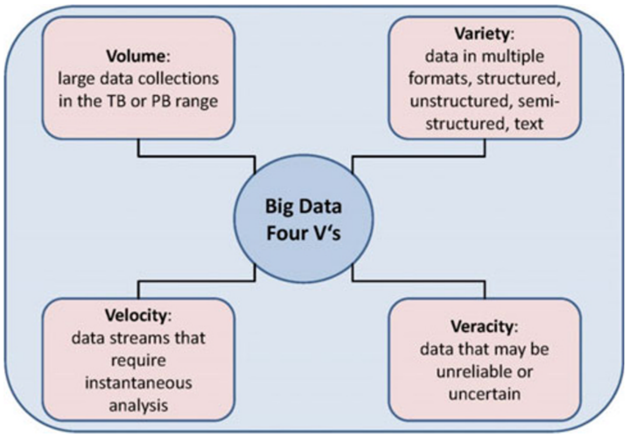
Volume:
large data collections in the TB or PB range
Variety:
data in multiple formats, structured, unstructured, semi-structured, text
Big Data Four V's
Velocity:
data streams that require instantaneous analysis
Veracity:
data that may be unreliable or uncertain

Fig. 1 The defining "4 Vs" of big data

size: data is considered big if it has reached TB or PB in size, and is typically so large that it exceeds a single organization's storage capacity. Other dimensions which have become common for characterizing big data, and which together with volume are called the "4 Vs of big data",[2] are the velocity or the speed with which data is produced and needs to be consumed, the variety data can have, and the veracity the data comes with (We note that the first three of these Vs are attributed to analyst Doug Laney[3] who now works for Gartner). Velocity refers to the fact that data often comes in the form of streams which do not give the respective consumer a chance to store them for whatever purpose, but to act on the data instantly. Variety means that data can come in different forms such as unstructured (e.g., text), semi-structured (e.g., XML documents), or structured (e.g., as a table), and veracity refers to the fact the data may or may not be trustworthy or uncertain. These characteristic properties of big data are summarized in Fig. 1.

We consider the transition from Web 1.0 to Web 2.0 as one of the major drivers that have led to big data. Indeed, as we have written in my book on Web 2.0 [26], this transition was determined by three parallel streams of development: the applications stream that has brought along a number of services anybody can nowadays use on the Internet and the Web; the technology stream which has provided the underlying infrastructure groundwork for all of this with fast moving and comprehensive advances in networking and hardware technology and quite a bit of progress regarding software; and finally the user participation and contribution stream (which we might also call the socialization stream) which has changed the way in which users, both private and professional ones, perceive the Web, interact with it, contribute to it, and in particular publish their own or their private information on it.

[2] http://www.ibmbigdatahub.com/infographic/four-vs-big-data.

[3] http://blogs.gartner.com/doug-laney/.

These three streams have brought along a number of techniques, technologies, and usage patterns that at present converge, and the result is what has received the term "Web 2.0". While initially content was mostly read from the Web, content is nowadays constantly written to the Web; hence the term "read/write Web". An immediate consequence of the fact that more and more people publish on the Web through blogs, instant messaging, social networks, and otherwise is that increasing amounts of data arise. Additionally, data arises from commercial sites, where each and every user or customer transaction leaves a trace in a database. Several years back, this made companies start employing data warehouse technology for online analytical processing or the application of data mining tools to large data collections to generate new knowledge. Especially, these tools have reached a new maturity, so that besides stored data it is now possible to process, or to incorporate into processing, data streams which cannot or need not be stored. We indeed consider "big" data as a consequence of the Web 2.0 developments, and it remains to be seen how to exploit this data in a fruitful way.

As can be done for other developments in computer science, big data can be viewed from various perspectives and in various dimensions; these are summarized in Fig. 2. As my goal in this paper is to give a brief survey of the current state of the big data area, we will first look at several use cases in Sect. 2 which indicate the enormous potential that can be seen in big data processing through a variety of examples and use cases; this touches the economical dimension. Section 3 covers the technological dimension and hence the technology available for handling big data, in particular technology that has made it to the center of attention recently. Section 4 takes an organizational perspective and describes how to exploit big data in an enterprise environment where a data warehouse has been the tool of choice until now; as it will turn out, a data warehouse architecture can straightforwardly be augmented to allow for big data. Section 5 concludes the paper and tries to give an outlook into what will happen next. Due to a lack of expertise of the author, the legal dimension will not be dealt with in this paper.

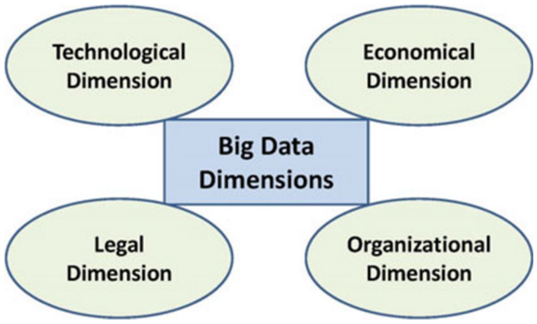

Fig. 2 Big data dimensions

2 Big data use cases

In this section, we describe several use cases for big data which are intended to indicate that this is indeed a development that is different from what we have seen in the past. As will be seen, they stem from vastly distinct areas, and it has to be kept in mind that these examples do not represent an exhaustive list.

One of the oldest examples of what data, at the time not yet called "big" data, can do when properly collected and analyzed is from the area of sports and refers to the Oakland Athletics baseball team and their coach Billy Beane, who was able to use statistics and player data to revamp the team from an unsuccessful one into a pretty successful one within a limited time span. The story is well documented in [12] and a movie based on that book. The already mentioned Doug Laney gives a more recent example from sports, namely from the Indy 500 race happening in the USA every year on Memorial Day weekend. According to Laney, a present-day Indy 500 race car is on the inside "smattered with nearly 200 sensors constantly measuring the performance of the engine, clutch, gearbox, differential, fuel system, oil, steering, tires, drag reduction system, and dozens of other components, as well as the drivers' health. These sensors spew about 1 GB of telemetry per race to engineers poring over them during the race and data scientists crunching them between races. According to McLaren, "its computers run a thousand simulations during the race. After just a couple laps they can predict the performance of each subsystem with up to 90 % accuracy. And since most of these subsystems can be tuned during the race, engineers, pit crews and drivers can proactively make minute adjustments throughout the race as the car and conditions change". Further details can be found on Laney's blog,[4] and it is obvious that the situation for Formula 1 cars[5] or the NASCAR series is similar.

An example from emergency response occurred in connection with hurricane Sandy, a gigantic storm which hit the Caribbean as well as the US east coast in the fall of 2012; company Direct Relief applied big data technology to coordinate rescue activities. On their website, they state: "using analytics and mapping software from technology partners, Palantir and Esri, Direct Relief was able to better understand needs on the ground and deploy appropriate resources to those areas. Beginning with preparedness activities driven by social vulnerability and health risk analysis, and extending through meteorological investigations, rapid scrutiny of shipping histories and continual monitoring of clinic status, shelters, pharmacies, and power outages within a common framework, Direct Relief connects clinics with essential medical resources by using the best insights available to assess needs, scale problems and track the rapid pace of events."[6]

A third area that will finally come to life with big data is home automation, a field that has been under development for more than 10 years now, but which so far has not taken off on a large scale (at least not in Europe). It is to be expected that this is now going to change, with the technical ability to process data from air conditioning, heating, lighting, or household devices such as washers, dryers, and refrigerators in conjunction with personal information from the people living in a house, to create living conditions optimally adapted to a particular age- or health-related situation. The latter remark carries over to the domain of health care, which is also increasingly supported by or based upon data gathered about a patient's medical condition, daily activity, nutrition, as well as other input, e.g., from drug manufacturers and its appropriate processing. This area will particularly boom in the future due to an availability of personal sequence or genome data and an increasing understanding of which portions of it (i.e., genes) are responsible for what disease or defect. The increasing dissemination of personal tracking devices such as the Fitbit,[7] the Nike+ Fuelband[8] or the Jawbone Up[9] will deliver another source of data that will be welcomed by health experts as well as the users themselves.

Other areas that are already big on big data analytics include market research, traffic management (e.g., in countries like Singapore) or autonomous cars which can drive by themselves and communicate with other cars. In the entertainment industry, Disney Parks and Resorts has developed the MyMagic+ system, which through the My Disney Experience website and the corresponding mobile app can deliver up-to-date information on current offerings to prospective guests planning a trip to one of the Disney parks. Disney's MagicBand can be used by the guest as a room key, a ticket for the theme park, access to FastPass+ selection, or to make a purchase. Participating visitors can skip queues, reserve attractions in advance and later change them via their smartphone, and they will be greeted by Disney characters by their name. The system behind MagicBand is that it collects data about the visitor, his or her current location, purchase history, and which attractions have been visited.

To close our brief survey of big data applications, we mention that social media sites or search engines are also intensively analyzing the data that they can get hold of. Indeed, Twitter analyzes the tweets its users are generating, for example, to identify and compare user groups, to analyze user habit, or to perform sentiment analyses on the text of tweets.

[4] http://blogs.gartner.com/doug-laney/the-indy-500-big-race-bigger-data/.

[5] http://www.quantumblack.com/formula-1-race-strategy-2/.

[6] http://www.directrelief.org/emergency/hurricane-sandy-relief-and-recovery/.

[7] http://www.fitbit.com.

[8] http://www.nike.com/cdp/fuelband/us/en_us/.

[9] https://jawbone.com/up.

Similarly, Facebook is interested in the number of “likes” a page gets over time and keeps a counter for recommended URLs, to make sure it takes less than 30 s from a click to an update of the respective counter. Google performs text clustering in Google News and ties to show similar news next to each other; moreover, they classify e-mails in Gmail and perform various other analytic tasks, e.g., in connection with their AdWords business.

3 Technology for handling big data

To cope with big data, a variety of techniques, methods, and technology have been developed in recent years, which are surveyed next. In particular, when data comes in such large quantities that local or in-house storage and processing is not an option anymore, it is not a surprise that “traditional” technology focusing around a central database is no longer apt. To determine what is needed and what fits in well, we first look at requirements for big data processing and then review technologies satisfying these requirements.

In a nutshell, these requirements can be characterized as follows:

- considerable processing power for complex computations;
- scalable, distributed and fault-tolerant data processing capabilities, including temporary or even permanent storage;
- parallel programming and processing paradigms suitable for handling large collections of data;
- appropriate implementations and execution environments for these programming models and paradigms.

Regarding hardware solutions for processing big data, we refer the reader to [20]. Also relevant in this context is a revival of main memory or in-memory database technology, a development that was first studied in the 1980s [8] and that has finally become available in commercial products [13,17] thanks to considerable technological advances during the last 30 years. The database field has furthermore brought along not only SQL (“NoSQL”) databases for coping with the requirements of big data applications such as scalability, wide distribution, and fault tolerance, which come in various flavors including key value stores (e.g., Amazon’s SimpleDB or Dynamo, LinkedIn’s Voldemort), column stores (e.g., Google’s BigTable [5], Apache’s Hbase, or Cassandra, Yahoo! PNUTS), document databases (e.g., MongoDB or Couchbase), and more recently graph databases (e.g., Neo4J or Allegro) [19]. In addition, “NewSQL” databases such as Clustrix, NuoDB, VoltDB, and Google’s spanner promise transactional guarantees in addition to NoSQL’s scalability.

If data can no longer be exclusively stored locally, it is near at hand to refer to cloud storage as an extension of local or in-house capabilities, or to stream processing systems that can vastly do without considerable local storage. For the sake of completeness, the difference between a database system and a data stream system is illustrated in Fig. 3 for the aspects of querying: A database query can be sent to a database sys-

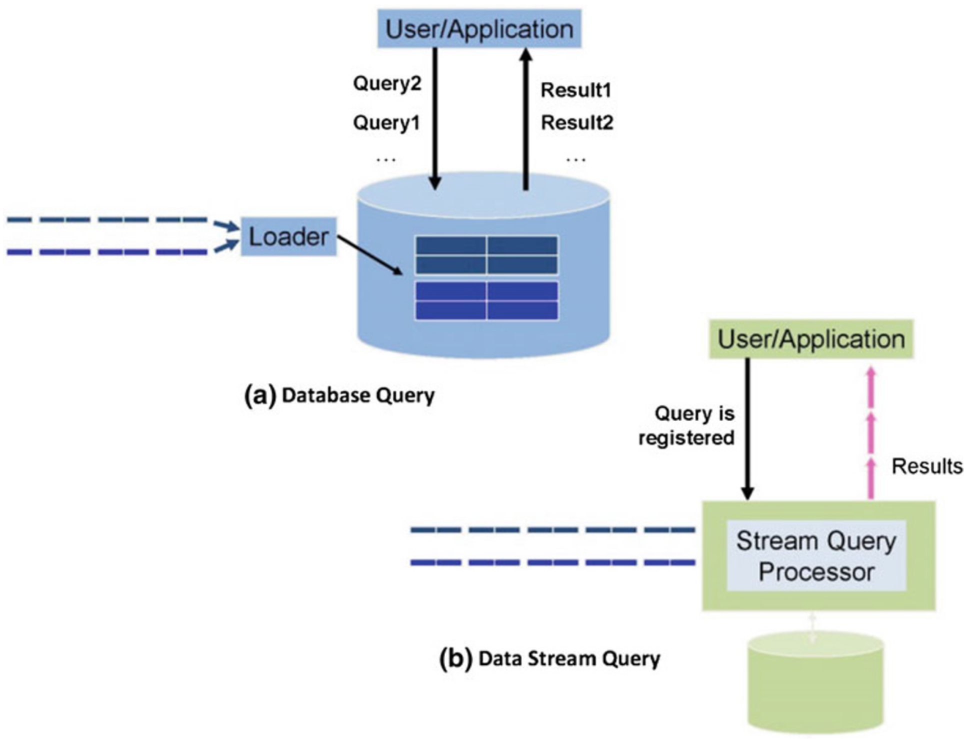

Fig. 3 Database query vs. data stream query

tem in an ad hoc manner and each query will be processed and produce a result individually (Fig. 3a), due to the fact that data is loaded and then permanently stored. In a data stream system, on the other hand, the data is streamed to a query processor continuously and without the option of being available for long periods of time; so the query processor can only respond to queries that have previously been registered with it and can produce result for the data stream by looking at a portion of the stream available within a certain window (Fig. 3b). The figure is, however, incomplete in that a stream processing system is often complemented by local storage or even part of a regular database system.

The ability to process data that is only available as a stream (e.g., data from temperature or pressure sensors in a weather station), but occurs at high frequency obviously requires certain processing power. This aspect of big data processing is not considered a major problem anymore, due to the availability of multi-core processors, GPU computing, in-memory computing, main memory database systems, and the widespread provisioning of high-performance data centers.

So for both computing and storage, cloud sourcing has become a typical scenario, which according to the US National Institute for Standards and Technology (NIST) is defined as follows—cloud sourcing is the utilization of IT capabilities from a cloud service provider based on the cloud paradigm with the following five characteristics: resource pooling, rapid elasticity, on-demand self-service, broad network access, and measured service. NIST defines three service models: Software-, Platform- and Infrastructure-as-a-Service, abbreviated as SaaS, PaaS and IaaS, respectively, which represent different types of services and, in a sense, different levels of abstraction from the underlying physical IT infrastructure. All three service models are used when it comes to big data: often IaaS for simple access to "unlimited" computing and/or storage capabilities, PaaS to establish one's own linguistic or algorithmic paradigm for processing big data, and SaaS when it comes to simply using a service or a combination of services for big data business analytics.

Cloud providers in this area typically base their processing power on large collections of commodity hardware, including conventional processors ("compute nodes") connected via Ethernet or inexpensive switches, which are arranged in clusters and which are replicated within as well as across data centers. Replication as a form of redundancy is the key to hardware reliability and fault-tolerant processing, and in just the same way data is protected against losses via replication. The result is either a distributed file system such as the Hadoop distributed file system (HDFS, see below) or a globally distributed database such as Google's Spanner [6]. Besides fault tolerance and availability, distribution can enhance parallel processing of the given data, in particular when computing tasks can be executed independently on distinct subsets of the data. In such a case, data is often partitioned over several clusters or even data centers; Fig. 4 illustrates the difference between partitioning and replication.

In the example shown in Fig. 4, data from a relational database about customer orders is partitioned over three different sites in such a way that each site is assigned distinct customer numbers, while products data is replicated over the sites (i.e., identically copied). Queries and updates can now go to a particular partition or to multiple partitions at the same time. If an organization like the one shown is run by a cloud provider underneath an SaaS product, a user does not need to care about proper data handling.

Data management in the cloud has its specific challenges when it comes to balancing consistency against availability and resiliency to partitioning failures [3]. Keeping a distributed data collection, file system, or database consistent at all times such that an access to any fraction of it will never see inconsistent or invalid data is hard to maintain, in particular since in a distributed system both hardware and software failures are frequent. Since the good news is that not every application running in the cloud permanently needs full consistency (in the sense of serializability [27]), consistency can often be relaxed into what is known as eventual consistency: when no updates occur for a long period of time, eventually all updates will propagate through the system and all the nodes will be consistent; for a given accepted update and a given node, eventually either the update reaches the node or the node is removed from service. Eventual consistency is

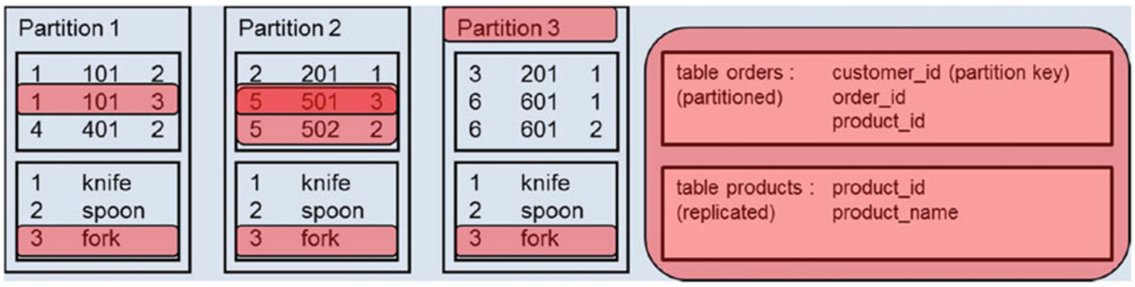

Fig. 4 Partitioning vs. replication

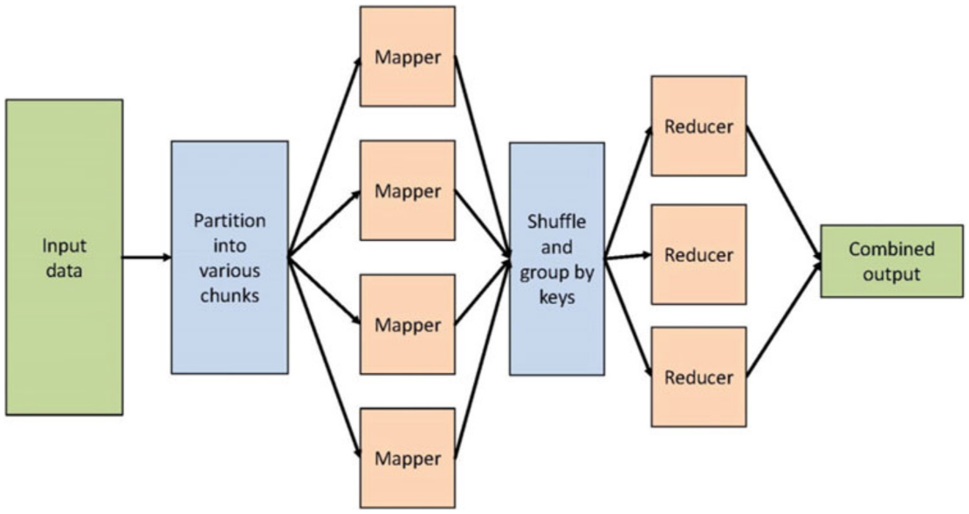

Fig. 5 Principle of a map-reduce computation

used, for example, by Amazon[10] within several of their data storage products or in CouchDB. An observation first made by Eric Brewer and later proved in [14] is that of consistency, availability, and partition tolerance, only two properties can be achieved simultaneously; this result has become known as the "CAP theorem". In other words, if an application wants to be immune against partition failures and needs to guarantee high availability, it has to compromise in consistency. Conversely, an application that needs consistency along with high availability cannot expect to be partition tolerant and hence has to take measures for handling partition failures.

The NoSQL systems mentioned above have reacted and responded to the CAP theorem in various ways, most often by allowing for relaxed notions of consistency, yet more recent developments such as Google's Spanner [6] and F1 [25] claim to be able to go back to strict forms of consistency.

While replication is a measure to enhance data availability, since if one copy fails another might still be available, partitioning turns out to be the key to tackling many large data problems algorithmically. Partitioning essentially follows the "old" principle of divide and conquer, which has a long tradition in computer science and its algorithms. If data can be split into various independent partitions (as in the example in Fig. 4 above), processing of that data can exploit parallelism, for example by keeping multiple cores of a processor or multiple CPUs in a cluster busy at the same time. The results obtained by these cores or CPUs may need to be combined to form a final processing result. This is the basic idea of Google's map-reduce [7] (US Patent 7,650,331, granted in January 2010) which employs higher-order functions (well known from the functional programming paradigm) for specifying distributed computations on massive amounts of data.

[10] http://www.allthingsdistributed.com/2008/12/eventually_consistent.html.

Map-reduce is a combination of two functions, map and reduce, which work on key–value pairs. A map-reduce computation essentially works as shown in Fig. 5: input data is made available in a number of data chunks, which typically come from a distributed file system. These chunks are fed into map tasks executed by components called mappers. Mappers turn their given chunk into a sequence of key–value pairs; exactly how these key–value pairs are generated from the input data depends on the particular computing task and is determined by the code written by the user for the map function. Next, mapper intermediate outputs are collected by a master controller and grouped by their key values. The keys and their associated value groups are then given to reduce tasks in such a way that all key–value pairs with the same key end up at the same reducer component. Finally, reducers work on one key at a time, and combine all the values associated with that key in a task-dependent way again specified by the code written by the user for the reduce function.

Essentially, a map-reduce computation centers around two functions that resemble SQL's group-by followed by aggregation:

1. $\text{Map} : (K_1, V_1) \rightarrow \text{list}(K_2, V_2)$
2. $\text{Reduce} : [K_2, \text{list}(V_2)] \rightarrow \text{list}(K_3, V_3)$

As an example, we consider the analysis of weather data coming as long string from weather stations; our interest is in an overview of the maximum temperature per year. Input data in this case might look like the sample shown in Fig. 6. The weather station regularly sends long strings that have to be interpreted appropriately; every string contains, among other information, the ID of the station, the date of the measurement, longitude and latitude of the station's location, and the actual temperature.

Now, suppose the following input is received, where the parts relevant for determining the maximum temperature are highlighted (and temperature values are rounded to integers):

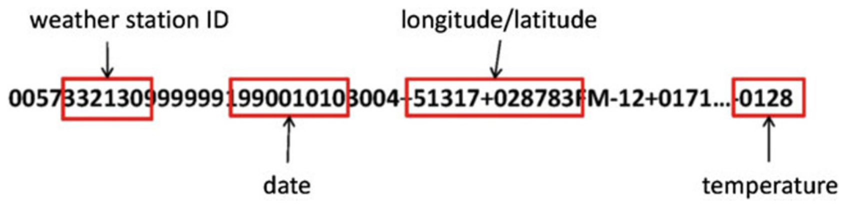

Fig. 6 Sample weather input data

0067011990999991**1990**051507004 + 51317
  + 028783FM − 12 + 0171. . .+ **0000**
0043011990999991**1990**051512004 + 51317
  + 028783FM − 12 + 0171. . .+ **0022**
0043011990999991**1990**051518004 + 51317
  + 028783FM − 12 + 0171. . .− **0011**
0043011990999991**1989**032412004 + 51317
  + 028783FM − 12 + 0171. . .+ **0111**
0043011990999991**1989**032418004 + 51317
  + 028783FM − 12 + 0171. . .+ **0078**

Suppose this is the chunk of data given to a mapper, then the latter will extract year (as key) and temperature (as value) as desired:

(1990, 0)
(1990, 22)
(1990, −11)
(1989, 111)
(1989, 78)

Shuffling and grouping this by key values will result in

(1989, [111, 78])
(1990, [0, 22, −11]),

from which a reducer can determine maxima as

(1989, 111)
(1990, 22).

It should be obvious that a task like this, which will in reality be based on huge amounts of weather station data, all of which can be processed independently, is a perfect candidate for a map-reduce computation. Other such tasks include counting the occurrences of words in a text collection (relevant to index creation and maintenance for a search engine), matrix–vector multiplication (relevant to PageRank computations for ordering search results), or operations from relational algebra (including joins and aggregate operations relevant to query optimization in databases) [18].

Clearly, several issues need to be addressed to make a map-reduce computation work, including the following:

- How do we actually write the code for a particular map-reduce task?
- How do we decompose a given problem into smaller chunks which can be processed in parallel?
- How do we adequately assign tasks to compute nodes (executing a mapper or a reducer)?
- How do we coordinate synchronization between the different compute nodes involved in a computation?
- How do we make such a scenario robust against failures?

The first question needs to be answered by a user who writes the map and reduces functions, say, in a high-level programming language such as Java. For our weather example above (and assuming that Hadoop will be used for executing the code, see below), the code for map could be as follows, which implements the map function as a Java class:

```
public class MaxTemperatureMapper
      extends MapReduceBase
      implements Mapper<LongWritable, Text, Text, IntWritable> {

   public void map(LongWritable key, Text value,
      OutputCollector<Text, IntWritable> output, Reporter reporter) {

      String line = value.toString();
      String year = line.substring(15, 19);
      int temperature = Integer.parseInt(line.substring(87, 92);

      output.collect(new Text(year), new IntWritable(temperature));
   }
}
```

This resembles what has been informally described above: from a given input string, year and temperature are extracted and written to an (intermediate) output file. Next, the reduce function can be written as another Java class:

```
public class MaxTemperatureReducer
                    extends MapReduceBase
                    implements Reducer<Text, IntWritable, Text, IntWritable> {

          public void reduce(Text key, Iterator<IntWritable> values,
                    OutputCollector<Text, IntWritable> output, Reporter reporter) {

                    int maxValue = Integer.MIN_VALUE;
                    while (values.hasNext()) {
                              maxValue = Math.max(maxValue, values.next().get());
                    }
                    output.collect(key, new IntWritable(maxValue));
          }
}
```

Here, variable `maxValue` is first initialized to the smallest integer and then repeatedly updated whenever a larger value for the same year is encountered.

The other questions listed above have been answered in recent years in various ways, the best known of which is the software library Hadoop.[11] Hadoop [28] supports scalable computations across distributed clusters of machines. Its core components are the MapReduce Engine and the HDFS. The MapReduce Engine is responsible for execution and control of map-reduce jobs; HDFS is a distributed file system in which large datasets can be stored, read, and output. User data is divided into blocks, which get replicated across the local disks of cluster nodes. HDFS is based on a master–slave architecture, where a namenode as master maintains the file namespace including the file-to-block mapping and the location of blocks, and datanodes as slaves manage the actual blocks. This is shown in Fig. 7, which is taken from the HDFS architecture guide.[12] Besides these main components, there are numerous extensions of Hadoop by specific functionality which together are considered the Hadoop ecosystem. Meanwhile, there has also been a host of suggestions for Hadoop alternatives (e.g., Disco, Skynet, Twister, or FileMap) as well as an evolution into Hadoop YARN. For an overview of Hadoop users, we refer the reader to the Apache website.[13]

We mention that the map-reduce paradigm and its Hadoop implementation not only have spawned a host of development in recent years, which has resulted in a variety of commercial offerings, but also a lot of research; as a starting point, we refer the reader to [1,2,9,18,21,22]. A typical extension of the basic map-reduce paradigm is PACT, a programming model described in [4] that generalizes the map-reduce model by adding more functions as well as ways to specify behavior guarantees for functions.

We conclude our survey of the technological dimension of big data by mentioning various other fields that are relevant here: The first is the area of statistical computing or computational statistics, which lies on the border of statistics and computer science and is concerned with the development of statistical programming languages such as R and with the design and implementation of statistical algorithms (such as those available in packages like SPSS, SAS, or STATA). The second relevant area is data mining, i.e., the process of discovering patterns (such as association rules or clusters) in large datasets; data mining has since the 1990s become popular as a collection of techniques to unleash previously unknown knowledge from raw data [10]. The third is visualization, which is concerned with the construction of visual representations of numerical, textual, or geographic data to reinforce human cognition and to ease interpretation of data or computation results stemming from that data. See, for example [16], for a survey of visualization techniques used in connection with social network data.

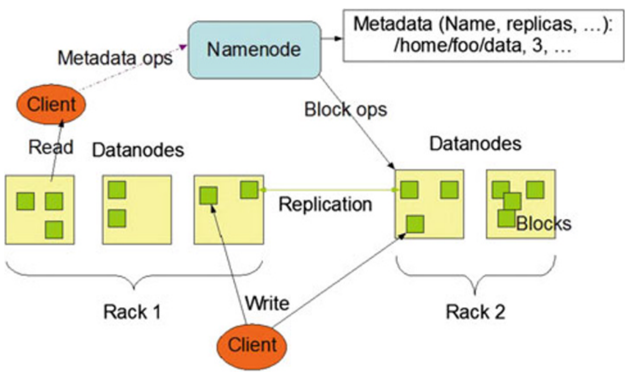

Fig. 7 HDFS architecture

11 http://hadoop.apache.org/.

12 http://hadoop.apache.org/docs/stable/hdfs_design.html.

13 http://wiki.apache.org/hadoop/PoweredBy.

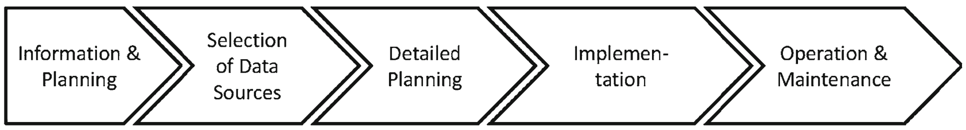

Fig. 8 An adoption strategy for big data

4 How do we exploit big data?

We now look at the organizational dimension of big data and consider the situation where a company or institution wants to make use of it. What does it take to do so, and what needs to change if the company has previously set up a data warehouse for its data analytics purposes? In particular, we briefly look at strategy development and then present a modification of the "classical" data warehouse architecture that is intended to accommodate big data requirements.

As has been the case for many other IT adoption decisions that have arisen over the years, it makes sense to base a decision of whether to start a big data project or to adopt big data technology on well-grounded considerations. To this end, techniques such as a SWOT analysis can help, which may be able to reveal the strengths, weaknesses, opportunities, and threads of a particular technology or project. Another tool that could be used in decision making is context analysis, which looks at objectives, added values, and the general context and environment into which a project should fit. Both SWOT and context analysis are popular and have proven successful, for example, in business process modeling [23].

More comprehensive than specific analyses is the development of a strategy for big data, which may look like the one shown in Fig. 8.

It starts with information gathering and planning, which could involve either a SWOT analysis or a context analysis or both. If a decision is made in favor of a big data project or of a general adoption of big data technology, relevant data sources need to be selected, which in an enterprise could be a variety of in-house source, e.g., databases, but could also be a variety of external sources, e.g., from the Web, which may provide relevant data for free or for a cost (as in the case of a data marketplace, see Sect. 5). The third phase of detailed planning includes a selection of the technology to be employed, e.g., the selection of a specific Hadoop implementation. Then the implementation can take place; finally, the system or project is in operation and may need regular or ad hoc maintenance.

We do not delve into further details of strategy development here, but mention that it may help even non-IT enterprises to take advantage of data analytics and intelligence that is nowadays available. When it comes to business intelligence, what has been at the center of attention for many years is the data warehouse [11], traditionally understood as a database separate from operational systems that is built via an ETL process extracting, transforming, and loading data from the various sources into the warehouse, and that is then the basis for online analytical processing, planning and reporting, ad hoc querying, spreadsheet and dashboard as well as data mining applications.

The basic architecture of a data warehouse can also be recognized from the right half of Fig. 9, yet the figure also indicates how to extend a traditional data warehouse architecture for big data. Indeed, what is new in this figure is a wider selection of external data sources than typically considered and the extension by a map-reduce engine such as Hadoop on the left side. Various ways of communication need to be made available between these old and new building blocks, but in the end the setup might look as shown in the figure.

We mention that running business intelligence and analytics applications does not necessarily require the existence of a data warehouse. Many tools are nowadays available that can be operated as add-ons to the operational systems and databases that an enterprise is already running and using. In that case, an explicit architecture design is not needed, and the same remarks apply to big data applications. However, experience shows that strategy as well as architectural considerations, in particular when they are well documented, can help an enterprise prevent project failures.

5 Conclusions and outlook

In this paper we have tried to survey various dimensions that are relevant to the field of big data that has emerged in recent years. Essentially, big data refers to the concept that data is nowadays available in an abundance that was never known before, that data-processing technology is capable of handling huge amounts of data efficiently, and that therefore there are large and primarily economic opportunities for exploiting this data. The notion of business intelligence that was "invented" in the context of (early) data mining as a circumscription of the fact that business can improve or enhance their "intelligence" regarding customers and revenues by analyzing and "massaging" their data to discover the unknown will now enter the next level. Indeed, a consequence of the fact that more and more data is made available in digital form not only allows businesses to gain new insights, but also renders new discoveries possible in areas such as physics or health care which are not necessarily of primary type "business". So not only regarding business, big data can indeed be seen as the new intelligence enabler, since the broadness of data available today (not just its sheer size!)

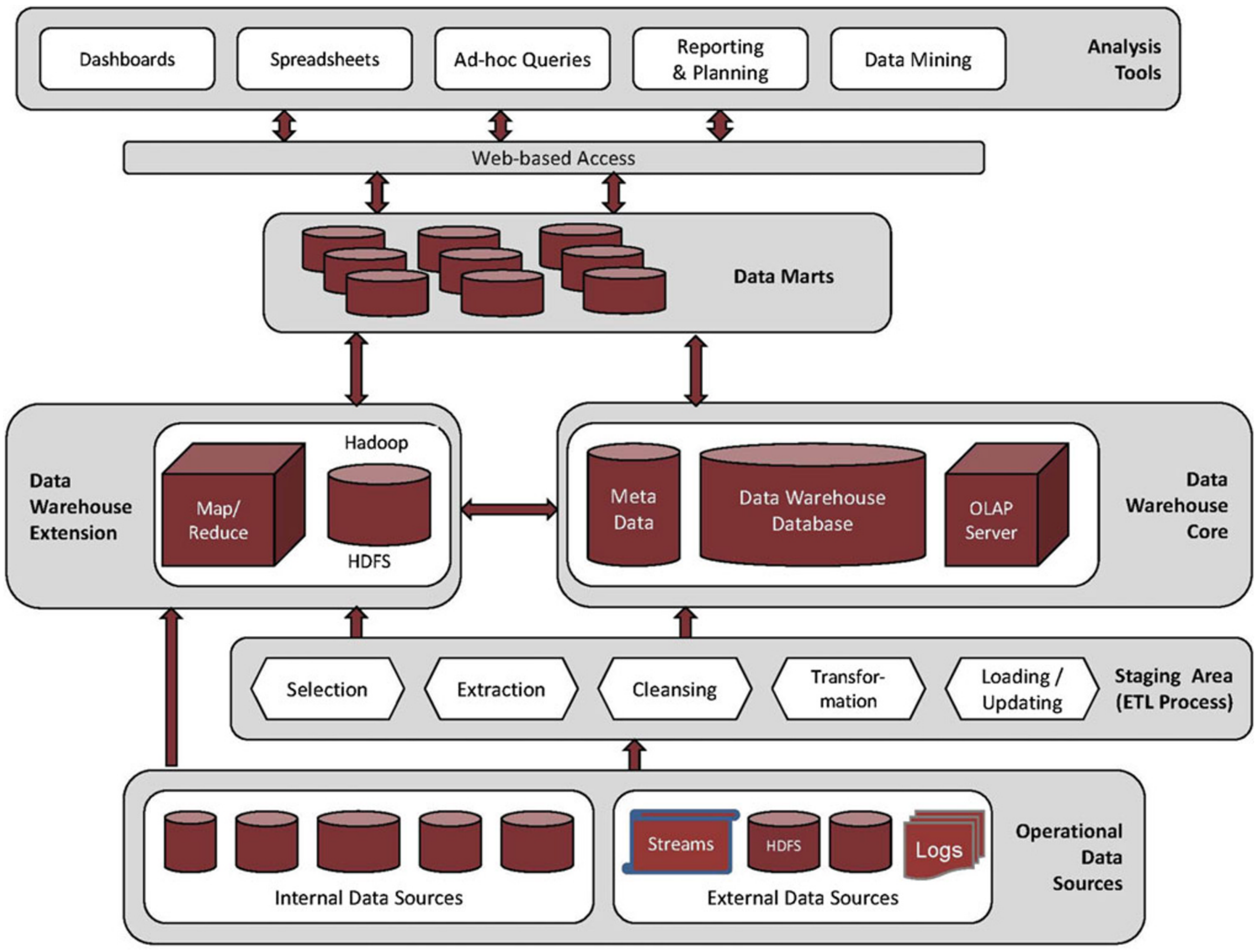

Fig. 9 Data warehouse architecture enhanced for big data processing

and the available technology enable us to perform analytics, to see connections, and to make predictions unthinkable only a short while ago.

We should mention that there is also a downside to all of this, best illustrated by the recent discovery how comprehensively and deeply the American NSA has been spying on people, companies, and even countries worldwide.[14] While security breaches and data misuse have always been a challenge in computer science, this reaches a new level with big data. Website io9 lists a number of ways in which big data is creating the “science fiction future”,[15] among them that dating sites that can predict when you are lying, that surveillance gets really Orwellian, really fast, or that scientists and doctors can make sense of your genome and so can insurers. We should hence be aware that big data does not just require the right technology, but also needs an appropriate governance and protection.

To conclude, we mention two developments that are foreseeable in the near future. The first is the fact that big data will have an impact on academic education. Indeed, a number of schools, so far primarily in the USA, have already launched programs for educating “data scientists”. We expect this trend to continue, at the borderline of computer science, statistics, machine learning, and possibly other fields such as communication and social sciences or medicine.

Second, as has happened with other goods in the past, when data becomes a commodity, we will see the emergence of (virtual) marketplaces for data just as the past has seen the creation of marketplaces, say, for stock. The stock market is characterized by the fact that it not only sells shares in companies, but offers a variety of other products that may or may not be derived from basic stock. In a similar way, a data marketplace will offer raw data, say, on a certain topic, and will also offer a variety of ways in which this data can

[14] https://www.eff.org/nsa-spying, http://www.theguardian.com/world/2013/sep/09/nsa-spying-brazil-oil-petrobras, http://www.bloomberg.com/news/2013-09-10/nsa-phone-records-spying-violated-court-rules-for-years.html.

[15] http://io9.com/5877560/10-ways-big-data-is-creating-the-science-fiction-future.

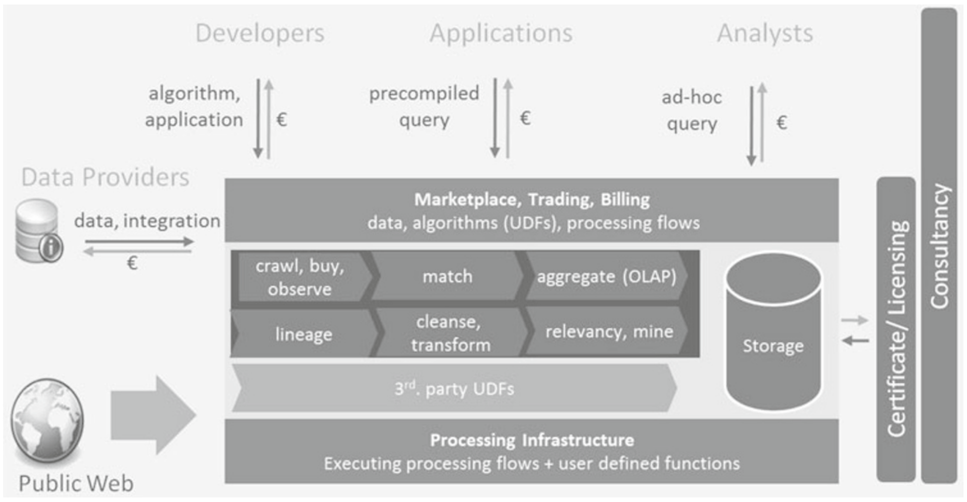

Fig. 10 Concept of a data marketplace

be processed prior to being sold. Different from the stock market, however, data marketplace may be open to anyone, i.e., users can act as sellers or buyers or both.

Figure 10, which originally appeared in [15], shows the general schema of a data marketplace for integrating public Web data with other data sources. In analogy to a data warehouse architecture, the schema includes components for data extraction, transformation and loading, as well as meta data repositories describing data and algorithms. In addition, the data marketplace offers interfaces for uploading data and methods for optimizing data, e.g., by employing operators with user-defined-functionality, as well as components for trading and billing the usage of these operators. In return, the provider of the user-defined function retrieves a monetary consumption (indicated by the euro symbol) from buyers. Moreover, in the case of large data volumes from the Web, the marketplace relies on a scalable infrastructure for processing and indexing data. A survey of the state of the-art in this field can be found in [24].

Acknowledgments The author is grateful to Jens Lechtenbörger for constructive comments on an earlier version of this paper.

References

1. Afrati, F., Das Sarma, A., Salihoglu, S., Ullman, J.D.: Vision paper: towards an understanding of the limits of map-reduce computation. CoRR abs/1204.1754 (2012)
2. Afrati, F., Das Sarma, A., Salihoglu, S., Ullman, J.D.: Upper and lower bounds on the cost of a map-reduce computation. PVLDB **6**(4), 277–288 (2013)
3. Agrawal, D., Das, S., El Abbadi, A.: Data management in the cloud: challenges and opportunities. Synth. Lect Data Manag. **4**(6), 1–138 (2012)
4. Battré, D., et al.: Nephele/PACTs: a programming model and execution framework for web-scale analytical processing. Proc. 1st ACM Symp. Cloud. Comput. (SoCC) 119–130 (2010)
5. Chang, F., et al.: Bigtable: a distributed storage system for structured data. ACM Trans. Comput. Syst. **26**(2), 1–26 (2008)
6. Corbett, J.C., et al.: Spanner: Google's globally-distributed database. In: Proceedings of the 10th USENIX Symposium on Operating Systems Design and Implementation (OSDI) (2012)
7. Dean, J., Ghemawat, S.: MapReduce: simplified data processing on large clusters", proceedings of the 6th symposium on operating system design and implementation (OSDI) (2004) and communication. ACM **51**(1), 107–113 (2008)
8. Eich, M.H.: Main memory database research directions. In: Proceedings of the International Workshop on Database Machines, pp. 251–268 (1989)
9. Fedak, G., et al.: Special issue of mapreduce and its applications. Concurr. Comput. Pract. Exp. **25**(1), (2013)
10. Han, J., Kamber, M., Pei, J.: Data Mining: Concepts and Techniques, 3rd edn. Morgan Kaufmann Publishers, Burlington (2011)
11. Inmon, W.H.: Building the Data Warehouse, 4th edn. Wiley, New York (2005)
12. Lewis, M.: Moneyball: The Art of Winning an Unfair Game. Norton & Company, USA (2004)
13. Loos, P., et al.: In-memory databases in business information systems. Bus. Inf. Syst. Eng. **6**, 389–395 (2011)
14. Lynch, N., Gilbert, S.: Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. ACM SIGACT News **33**(2), 51–59 (2002)
15. Muschalle, A., et al.: Pricing approaches for data markets. In: Castellanos, M. (ed.) BIRTE 2012 (Proceedings of the 6th International Workshop on Business Intelligence for the Real Time Enterprise 2012, Istanbul), pp. 129–144. Springer LNBIP, New York (2013)
16. Pflanzl, N.: State-of-the-Art of Social Network Visualization. Master thesis, University of Münster, Department of Information Systems (2012)
17. Plattner, H., Zeier, A.: In-Memory Data Management—An Inflection Point for Enterprise Applications. Springer, Berlin (2011)
18. Rajaraman, A., Lescovec, J., Ullman, J.D.: Mining of Massive Datasets; downloadable from http://infolab.stanford.edu/ullman/mmds.html (2013)
19. Redmond, E., Wilson, J.R.: Seven Databases in Seven Weeks: A Guide to Modern Databases and the NoSQL Movement. Pragmatic Programmers, Dallas, TX, USA (2012)
20. Saecker, S., Markl, V.: Big Data Analytics on Modern Hardware Architectures: A Technology Survey; European Business Intelligence Summer School (eBISS), pp. 125–149. Spinger LNBPI, New York (2012)

21. Sauer, C., Härder, T.: Compilation of query languages into mapreduce. Datenbank-Spektrum **13**(1), 5–15 (2013)
22. Shim, K.: MapReduce algorithms for big data analysis. Spinger LNCS **7813**, 44–48 (2013)
23. Schönthaler, F., Vossen, G., Oberweis, A., Karle, T.: Business Processes for Business Communities. Springer, Berlin (2012)
24. Schomm, S., Stahl, F., Vossen, G.: Marketplaces for data: an initial survey. ACM SIGMOD Rec. **42**(1), 15–26 (2013)
25. Shute, J., et al.: F1: A Distributed SQL Database That Scales. Proc. VLDB Endowment **6**(11), 1068–1079
26. Vossen, G., Hagemann, St.: Unleashing Web 2.0—From Concepts to Creativity. Morgan Kaufmann Publishers, Burlington (2007)
27. Weikum, G., Vossen, G.: Transactional Information Systems—Theory, Algorithms, and the Practice of Concurrency Control and Recovery. Morgan Kaufmann Publishers, San Francisco (2002)
28. White, T.: Hadoop: The Definitive Guide, 3rd edn. O'Reilly Media, Sebastopol (2012)