New Database Architectures: Steps Towards Big Data Processing

CONFERENCE PAPER · JULY 2013

READS
397

1 AUTHOR:

Jaroslav Pokorný
Charles University in Prague
163 PUBLICATIONS **777** CITATIONS

SEE PROFILE

NEW DATABASE ARCHITECTURES: STEPS TOWARDS BIG DATA PROCESSING

Jaroslav Pokorný
Charles University, Faculty of Mathematics and Physics, - Malostranské nám. 25, 118 00 Praha 1, Czech Republic

ABSTRACT

Both research and practice indicate that traditional universal DBMS architecture hardly satisfies new trends in data processing, particularly in the context of cloud computing and Big Data problems. New database architectures and their basic features will be described, particularly their horizontal scalability and concurrency model, which is mostly weaker than ACID transactions in relational SQL-like database systems. We focus on so called NoSQL databases which support solving, at least partially, Big Data problems. Some features of NoSQL databases like data models and querying capabilities are presented in more detail. We will also mention an overview of some their representatives. Finally, we point out on actual problems associated with current database research at all.

KEYWORDS

NoSQL databases, Big Data, data mining

1. INTRODUCTION

D. Feinberg reported in (Feinberg, 2011) that worldwide information volume is growing at a minimum rate of 59% annually. A "Big Data movement" reflects problems resulting in large part from our inability to utilize vast amounts of information effectively. This concerns in part Big Data storage and processing at low-level and analytical tools on higher levels. It seems that from a user's point view just Big Analytics is the most important aspect of Big Data computing. Unfortunately, large datasets are expressed in different formats, for instance, relational, XML, textual, multimedia or RDF, which may cause difficulties in its processing, e.g., by data mining algorithms. Also, increasing either data volume in a repository or the number of users of this repository requires more feasible solution of scaling in such dynamic environments than it is offered by traditional database architectures.

Users have a number of options how to approach the problems associated with such data. For storing and processing large datasets they can use:

- traditional parallel database systems,
- Hadoop technologies,
- key-value datastores (so called NoSQL databases).

NoSQL databases are a relatively new type of databases which is becoming more and more popular mostly among Web companies today. Clearly, Big Analytics is done also on big amounts of transaction data as extension of methods used usually in technology of data warehouses (DW). But DW technology was always focused on structured data in comparison to much richer variability of Big Data as it is understood today. Consequently, analytical processing of Big Data requires not only new database architectures but also new methods for analysing the data. The paper follows-up the work (Pokorný, 2013) on NoSQL databases and focuses in more extent on challenges coming with Big Data. Its contribution is to relate principles of NoSQL databases and Hadoop technologies with Big Data problems and show some alternatives in this area.

Section 2 describes two types of database architectures: the traditional universal one going through the whole history of databases and Hadoop-like implementing MapReduce framework. In Section 3 we present a short overview of NoSQL technology, particularly data models, architectures, and some representatives of NoSQL databases. Section 4 is devoted to Big Data problems, i.e. their sources, characteristics, processing, and analysing. Section 5 concludes with trends and open problems associated with Big Data.

2. DATABASE ARCHITECTURES

In (Härder and Reuter, 1983) the authors described today already classical *universal DBMS architecture* based on a mapping model consisting from five abstraction layers (see Table 1). In the most general version the architecture is encapsulated together with use of the SQL language in L1. The same model can be used in the case of distributed databases for every node of a network together with a *connection layer* responsible for communication, adaptation, or mediation services. Also a typical shared-nothing parallel relational DBMS can be described by this architecture. Layers L1-L4 are present usually at each machine in a cluster. A typical property of the universal architecture is that its users can see only the outermost (SQL) layer.

Table 1. The five-layered DBMS mapping hierarchy

	Level of abstraction	**Objects**	**Auxiliary mapping data**
L5	non-procedural access or algebraic access	tables, views, rows	logical schema description
L4	record-oriented, navigational approach	records, sets, hierarchies, networks	logical and physical schema description
L3	records and access path management	physical records, access paths	free space tables, DB-key translation tables
L2	propagation control	segments, pages	buffers, page tables
L1	file management	files, blocks	directories

The number of layers in the universal architecture is often reduced to the well-known three-layer model (Härder, 2005). A lot of variations of the universal architecture like special-purpose servers for databases specialized on certain data type are discussed in (Pokorný, 2010).

In any case, associated database technologies both centralized and distributed was found not well-suited for Web-scale data management. Perhaps the most important problems is a hard scalability of traditional DBMSs in Web environment. A *vertical scaling* (called also *scale-up*), i.e. investments into new and expensive big servers, was replaced by database partitioning across multiple cheap machines added dynamically to a network. Such so called *horizontal scaling* (also *scale-out*) can apparently ensure scalability in a more effective and cheaper way. Data is distributed horizontally in the network that means, e.g., into groups of rows in the case of tabular data, but a vertical "shredding" or a combination of both styles are being used as well. Horizontal data distribution enables to divide computation into concurrently processed tasks.

Table 2. The three-layered Hadoop software stack

	Level of abstraction	**Data processing**
L5	non-procedural access	HiveQL/PigLatin/Jaql
L2-L4	record-oriented, navigational approach	Hadoop MapReduce Dataflow Layer
	records and access path management	HBase Key-Value Store
	propagation control	
L1	file management	Hadoop Distributed File System

A typical layered architecture of Hadoop software often used in NoSQL environment has also a three-layer model but it looks a little differently. The open source software Hadoop[1] is based on the framework MapReduce (Dean and Ghemawat, 2008) developed in Google for data processing and the Hadoop Distributed File System (HDFS)[2]. Hadoop is an entire framework that can be used with NoSQL DBMS. On the top of HDFS there is, e.g., the NoSQL database HBase[3]. Hadoop has quickly become a "gold-standard" in industry as a highly scalable data-intensive MapReduce platform. Table 2 based on (Borkar *et al*, 2012) documents this architecture.

[1] http://hadoop.apache.org/
[2] http://hadoop.apache.org/
[3] http://hbase.apache.org/

One remarkable difference of the Hadoop software stack from the universal DBMS architecture is that we can access data by three different sets of tools in particular layers. The middle layer Hadoop MapReduce system server serves for batch analytics (Input: Hadoop M/R jobs). HBase is available as a key-value layer (Input: Get/Put operations). Finally, high-level languages HiveQL (Facebook), PigLatin (Yahoo!), and Jaql (IBM) are for some users at disposal at the outermost layer. The use of declarative languages reduces code size by orders of magnitude and enables distributed or parallel execution.

Complexity of tasks for data processing in such alternative database architectures is minimized using programming languages, like MapReduce, occurring especially in context of NoSQL databases. Obviously the approach is not easily realizable for arbitrary algorithm and arbitrary programming language. Map-Reduce inspired by functional programming enables to implement, e.g., multiplication of sparse matrices by a vector in a natural way. On the other hand, it is worth to mention that computing in such languages does not enable effective implementation of the relational operation join.

3. NOSQL DATABASES

Typically, NoSQL means "not only SQL" or "no SQL at all", that makes this collection of databases very diverse. NoSQL solutions starting in development from late 90s provide simpler scalability and improved performance relative to traditional relational databases. Popularly said, NoSQL is used for *non-relational, distributed data stores that often do not attempt to provide ACID guarantees*. Particularly, these products are appropriate for storing semi-structured and unstructured data. NoSQL databases are also often used for storing Big Data.

3.1 Data Models Used in NoSQL Databases

What is principal in classical approaches to databases – a (logical) data model – is in approaches to NoSQL databases described rather intuitively, without any formal fundamentals. The NoSQL terminology is also very diverse and a difference between conceptual and database view of data is mostly blurred.

Most simple NoSQL databases called *key-value stores* (or *big hash tables*) contain a set of couples (key, value). A key uniquely identifies a value (typically string, but also a pointer, where the value is stored. A value for a given key (or row-ID) can be a collection of couples name (attribute name in relational databases or column name in SQL databases) and value attached to this name. The sequence of (name, value) couples are contained as a blob, usually in the format of a string. This means that data access operations, typically get and put, have only a key as the address argument. The approach key-value reminds simple abstractions as file systems or hash tables, which enables efficient lookups. However, it is essential here, that couples (name, value) can be of different types. In terms of relational data model – they may not "come" from the same table. Though very efficient and scalable, the disadvantage of too simple data models can be essential for such databases. On the other hand, NULL values are not necessary, since in all cases these databases are schema-less.

In a more complex case, NoSQL database stores combinations of couples (name, value) collected into collections, i.e. rows addressed by a key. Then we talk about *column NoSQL databases*. New columns can be added to these collections. There is a further level of structure (e.g., in CASSANDRA) called *super-columns*, where a column contains nested (sub)columns. Data access is improved by using column names in operations get, insert and delete.

The most general models are called (rather inconveniently) *document-oriented NoSQL databases*. The JSON[4] (JavaScript Object Notation) format is usually used to presentation of such data structures. JSON is a binary and typed data model which supports the data types list, map, date, Boolean as well as numbers of different precision. Document stores allow arbitrarily complex documents, i.e. subdocuments within subdocuments, lists with documents, etc. whereas column stores only allow a fixed format, e.g. strict one-level or two-level dictionaries. On a model level, they contain a set of couples (key, document).

We can observe that all the data models are in principle key-valued. The three categories considered distinguish mainly in possibilities of aggregation of (key, value) couples and accessing the values.

[4] http://www.json.org/

Often *graph databases* are considered as a category of NoSQL. They offer interesting issues, e.g., graph partitioning strategies in case of large data, efficient evaluation of graph queries (graph traversals, sub-graph and super-graph queries, and graph similarity evaluation), and graph query languages designed for particular types of graph data.

To the broad sense, NoSQL has seven categories. Also object-oriented, XML, and so called multivalued databases are mentioned. But the characteristics that are essential for the above four categories are not very obviously reflected in them.

Table 3. Representatives of NoSQL databases (see (Pokorny, 2013))

Name	Producer	Data model	Querying
key-valued			
SimpleDB	Amazon	set of couples (key, {attribute}), where attribute is a couple (name, {value})	restricted SQL; select, delete, GetAttributes, and PutAttributes operations
Redis	S. Sanfilippo	set of couples (key, value), where value is simple typed value, list, ordered (according to ranking) or unordered set, hash value	primitive operations for each value type
Memcached	B. Fitzpatrick	set of couples (key, value) in associative array	get, put and remove operations
Dynamo	Amazon	like Simple DB	simple get operation and put in a context
Voldemort	LinkeId	like Simple DB	similar to Dynamo
column-oriented			
BigTable	Google	set of couples (key, {value})	selection (by combination of row, column, and time stamp ranges)
HBase	Apache	groups of columns (a BigTable clone)	JRUBY IRB-based shell (similar to SQL)
Hypertable	Hypertable	like BigTable	HQL (Hypertext Query Language)
CASSANDRA	Apache (originally Facebook)	columns, groups of columns corresponding to a key (supercolumns)	simple selections on key, range queries, column or columns ranges
PNUTS	Yahoo	(hashed or ordered) tables, typed arrays, flexible schema	selection and projection from a single table (retrieve an arbitrary single record by primary key, range queries, complex predicates, ordering, top-k)
document-based			
MongoDB	10gen	object-structured documents stored in collections; each object has a primary key called ObjectId	manipulations with objects in collections (find object or objects via simple selections and logical expressions, delete, update)
CouchDB	Couchbase	document as a list of named (structured) items (JSON document)	views via Javascript and MapReduce

3.2 Architectures of NoSQL Databases

The category of NoSQL databases described in the list of well-maintained and structured Web site[5] includes at least 150 products. Some of these projects are more mature than others, but each of them is trying to solve

[5] http://nosql-database.org/

similar problems. Some of the products are in-memory databases. *In-memory* means data is stored in computer memory to make access to it faster. The fastest NoSQL data stores available today – Redis and Memcached entirely serve from memory, but a persistent variant MemcachedDB[6] exists also.

Table 3 presents some significant representatives of NoSQL databases. Another list of various opened and closed source NoSQL databases can be found in (Cattel, 2010). A very detailed explanation of NoSQL databases is presented in the work (Strauch, 2011).

NoSQL databases optimize processing massive amounts of data, while imposing a relatively weak consistency model (typically row-level locks). They use lightweight coordination mechanisms, which allow them to scale while keeping a partially centralized point of control. Typically, a support of scalability works against transactionality. In fact, due to the Brewer's theorem (Brewer, 2012), assuming a partitioning tolerance in our unreliable Web systems, mostly availability is prioritized over a consistency.

In the Big Data landscape NoSQL databases dominate rather for operational capabilities, i.e. interactive workloads where data is primarily captured and stored. Analytical Big Data workloads, on the other hand, tend to be addressed by parallel database systems and MapReduce. Nevertheless, NoSQL are also becoming as a dominant for Big Analytics, but with a lot disadvantages, e.g., a heavy computational model, low-level information about data processed, etc. Exclusions occur, e.g., Apache Mahout[7] implemented on top of Hadoop brings to bear automated machine learning to finding hidden trends and otherwise unthought-of or unconsidered ideas.

A new approach to database architectures supporting a combination of operational and analytical technologies can be found, e.g., in (Vinayak *et al*, 2012). Their ASTERIX system is fully parallel, it is able to store, access, index, query, analyze, and publish very large quantities of semi-structured data. Its architecture is similar to that one in Table 2, but with own Hyracks layer in the bottom to manage data-parallel computations, the Algebrics algebra layer in the middle, and the topmost ASTERIX system layer – a parallel information management system. The architecture includes also a Hadoop compatibility layer.

4. BIG DATA

Usually we talk about the Big Data when the dataset size is beyond the ability of the current software to collect, process, retrieve and manage this data. McKinsey, a leading research firm, describes in (Manyika *et al*, 2011) Big Data more functionally, as large pools of unstructured and structured data that can be captured, communicated, aggregated, stored, and analyzed which are now becoming part of every sector and function of the global economy. A specification of Big Data by some concrete numbers is offered, e.g., in (Morgan, 2012). Big Data is a system that has to collect more than 100TB of data annually, it has to grow at least 60% per year and it has to be deployed on scale-out architectures.

Besides traditional enterprise data (e.g., customer information from CRM systems, transactional ERP data, web store transactions, general ledger data), sources of Big Data include also sensor data, digital data streams (e.g. video, audio, RFID data) or e-science data coming from astronomy, biology, chemistry, and neuroscience, for example. In this context, related data driven applications are mentioned. As example we can use market research, smarter health-care systems, environment applications, posts to social media sites, water management, energy management, traffic management, but also astronomical data analysis (IBM Research, 2013).

Web plays a prominent role towards shifting to the Big Data paradigm. The textual Web content, a typical example of *Big Text* – is a source that people want to easily consult and search. Challenges in this area include, e.g., document summarization, personalized search, and recommender systems. The social structures formed over the Web, mainly represented by the online social networking applications such as Facebook, LinkedIn or Twitter, contribute intensively to Big Data. Typically, the interactions of the users within a social networking platform form graph structures, leading to the notion of *Big Graph*.

The area of *Linked Data* poses a different type of large data and thus different problems. The key aspect we need to ensure in this case is the quality of the data and their provenance. An emerging issue is in the area of *linked open data* (LOD), namely data that is in an open format and that is linked through universal

[6] http://memcachedb.org/
[7] http://mahout.apache.org/

resource names and published in RDF language. Users of LOD have a query language SPARQL at disposal; logics for automated reasoning are ensured by RDFS and OWL.

In general, Big Data come from four main contexts:

- large data collections in traditional data warehouses or databases,
- enterprise data of large, non-Web-based companies,
- data from large Web companies, including large unstructured data and graph data,
- data from e-science.

In any case, a typical feature of Big Data is the absence of a schema characterization, which makes difficulties when we want to integrate structured and unstructured datasets.

4.1 Big Data Characteristics

Big Data embodies new data characteristics created by our digitized world:

- *Volume* data at scale - size from TB to PB and more. Too much volume is a storage issue, but too much data is also a Big Analytics issue.
- *Velocity* data in motion – analysis of streaming data, structured record creation, and availability for access and delivery. Velocity means both how quickly data is being produced and how quickly the data must be processed to meet demand.
- *Variety* data in many formats/media types – structured, unstructured, semi-structured, text, media.
- *Veracity* uncertainty/quality – managing the reliability and predictability of inherently imprecise data.

The first three V have been introduced in (Gartner, 2011), the V from Veracity has been added by Dwaine Snow in his blog Dwaine Snow's Thoughts on Databases and Data Management[8] in 2012. IBM mentions the same step also in 2012[9]. Both Variety and Velocity are actually working against the Veracity of the data. They decrease the ability to cleanse the data before analysing it and making decisions. A fifth V occurred in (Gamble and Goble, 2011) belongs to

- *Value* worthwhile and valuable data for business.

Data value vision includes creating social and economic added value based on the intelligent use, management and re-uses of data sources with a view to increase business intelligence (BI) and efficiency of both private and business sectors as well as to support new business opportunities. Then, a major new trend in information processing is the trading of original and enriched data, effectively creating an *information economy*.

Sometimes another V is considered:

- *Visualization* visual representations and insights for decision-making (e.g., tag clouds, clustergrams, history flows, spatial information flows[10]).

4.2 Big Data Processing

A general observation is that as data is becoming more and more complex, also its analysis is becoming increasingly complex. To exploit this new resource, we need to scale up and scale out both infrastructures and standard techniques. Big Data and high performance computing (HPC) are playing essential roles in attacking the most important problems in this context. We may distinguish between the HPC and the Big Data in terms of combinatorial complexity of storing and the complexity of addressing the data. In the first case, the problem is not related to the amount of data but the combinatorial structure of the problem. In the second case, the problem is in the scalability by linearization rather than in the parallelism.

Big Data processing involves interactive processing and decision support processing of *data-at-rest* and real-time processing of *data-in-motion*. The latter is usually performed by *Data Stream Management Systems*. Hadoop based on MapReduce paradigm is appropriate rather for decision support. However, MapReduce is still very simple technique compared to those used in the area of distributed databases.

[8] http://dsnowondb2.blogspot.cz/2012/07/adding-4th-v-to-big-data-veracity.html
[9] http://anlenterprises.com/2012/10/30/ibms-4th-v-for-big-data-veracity/
[10] http://www.thbs.com/index.php/services/big-data-offerings/data-visualization

MapReduce is well suited for applications which analyse elements of a large dataset independently; however, applications whose data access patterns are more complex must be built using several invocations of the Map and Reduce steps. The performance of such design is dependent on the overall strategy as well as the nature and quality of the intermediate data representation and storage. For example, e-science applications involve complex computations which pose new challenges to MapReduce systems. As scientific data is often skewed and the runtime complexity of the reducer task is typically high, the resulted data processing M/R jobs may be not too effective. Similarly, MapReduce is not appropriate for ad hoc analyses but rather for organized data processing. On the contrary, NoSQL systems serve rather for interactive data serving environments.

Big Analytics is about turning information into knowledge using a combination of existing and new approaches. Related technologies include

- data management (uncertainty, query processing under near real-time constraints, information extraction),
- programming models,
- machine learning and statistical methods,
- systems architectures,
- information visualization.

A highly scalable platform supporting these technologies is called a *Big Data Management System* (BDMS) in (Vinayak *et al*, 2012). The ASTERIX mentioned in Section 3.2 belongs to the BDMS category.

Big Data is often associated with a cloud computing. But cloud computing means only computing resources that are delivered as a service, typically over the Internet. NoSQL databases used in such environments are mostly sufficient for storage and processing Big Data used there.

4.3 Big Analytics

Big Data are often mentioned only in context with BI. But not only BI developers also scientists analyse large collections of data. A challenge for computer specialists or data scientists is to provide these people with tools that can efficiently perform complex analytics that take into account the special nature of such data and their intended tasks. It is important to emphasise that Big Analytics involves not only the analysis and modelling phase. For example, noisy context, heterogeneity, and interpretation of results are necessary to be taken into account.

Feinleib lists eight "Big Data Laws"[11]. The Law #1 is: "The faster you analyze your data, the greater its predictive value ... Companies are moving away from batch processing to real time to gain competitive advantage. Therefore, a goal is not only manage and understand the data, but also acts on the data.

Besides these rather classical themes of mining Big Data, other interesting issues have appeared in last years, e.g., entity resolution and subjectivity analysis. The latter includes Sentiment Analysis and Opinion Mining as topics using Information Retrieval and Web data analysis. A particular problem is finding sentiment-based contradictions at a large scale and to characterize (e.g., in terms of demographics) and explain (e.g., in terms of news events) the identified contradictions.

5. CONCLUSION

In (Markl, 2013), three big trends in Big Data have been formulated:

- solving complex data analysis problems (predictive analytics, infinite data streams, distributed data dealing with uncertainty, bringing the human in the loop - visual analytics)
- producing results in near-real time (first results fast, low latency, exploiting new hardware e.g. multi-core, OpenGL/CUDA, FPGA, heterogeneous processors, NUMA, remote memory)
- hiding complexity (declarative data analysis programs, fault tolerance)

Toady's main task for research is improving the quality and scalability of data mining methods according to these trends. Indeed, the processes of query composition - especially in the absence of a schema - and the interpretation of the obtained answers may be non-trivial since the dataset returned as answer may be too big

[11] http://www.slideshare.net/bigdatalandscape/big-data-trends

to be easily human-readable. As much data toady is not natively in structured format, transforming the content into a structured format for later analysis is a challenge. Data mining techniques, already widely applied to extract frequent correlations of values from both structured and semi-structured datasets in BI, are interesting solutions for Big Analytics too, but they have to be extended and accommodated.

So far the mining process is guided by the analyst, whose knowledge of the application scenario determines the portion of data wherefrom the useful patterns can be extracted. A more advanced approach would be the automatic mining process and to extract extracts approximate, synthetic (intensional) information on both the structure and the contents of large datasets.

ACKNOWLEDGEMENT

This research has been supported by the grant of GACR No. P103/13/08195S.

REFERENCES

Borkar, V., Carey, M.-J., Chen Li, Ch.,2012. Inside "Big Data management": ogres, onions, or parfaits? *Proceedings of EDBT Conference*, Berlin, Germany, pp. 3-14.

Brewer, E.A., 2012. CAP twelve years later: how the 'rules' have changed, *In Computer*, Vol. 45, No. 2, pp. 22-9.

Cattell, R., 2010. Scalable SQL and NoSQL Data Stores. *In SIGMOD Record*, Vol. 39, No. 4, pp. 12-27.

Dean, D. and Ghemawat, S., 2008. MapReduce: Simplified Data Processing on Large Clusters. *In Communications the ACM*, Vol. 51, No. 1, pp. 107-113.

Feinberg, D., 2011.The End of the Database as We Know It: No DISK, No SQL and Very Cloudy. Gartner, Data Center Summit, 28-29 November 2011 London, U.K.

Feuerlicht, G. and Pokorny, J., 2012. Can Relational DBMS Scale-up to the Cloud? Pooley, R.J. et al (Eds.), *Information Systems Development - Reflections, Challenges and New Directions*, Springer-Verlag.

Gamble, M. and Goble, C., 2011. Quality, Trust and Utility of Scientific Data on the Web: Toward a Joint model. *Proceedings of WebSci'11 Conference*, Koblenz, Germany.

Gartner, 2011. Pattern-Based Strategy: Getting Value from Big Data. Gartner Group http://www.gartner.com/it/page.jsp?id=1731916

Härder, T., Reuter, A. ,1983. Concepts for Implementing and Centralized Database Management System. *Proceedings of Int. Computing Symposium on Application Systems Development*, Nürnberg, Germany, B.G., pp. 28-104.

Härder, T., 2005. DBMS Architecture – the Layer Model and its Evolution. *In Datenbank-Spektrum* 13, pp. 45-57. IBM Research, 2013. The Square Kilometer Array: The Ultimate Big Data Challenge. White Paper, IBM.

Manyika, J., Chui, M., Brown, B., Bughin, J., Dobbs, R., Roxburgh, C. and Byers, A.-H., 2011. Big data: the next frontier for innovation, competition, and productivity, McKinsey Global Inst.

Markl, V., 2013. Big Data - Analytics Chances and Challenges. Keynote Presentation at World Summit on Big Data and Organization Design, May 16-7, Paris, France.

Morgan, T. P., 2012. IDC: Big data biz worth $16.9 BILLION by 2015. *The Register*.

Pokorný, J., 2010. Databases in the 3rd Millennium: Trends and Research Directions. *In Journal of Systems Integration*, Vol. 1, No. 1-2, pp. 3-15.

Pokorny J., 2013. NoSQL Databases: a step to databases scalability in Web environment. *In International Journal of Web Information Systems*, Vol. 9, No. 1, pp. 69-82.

Strauch, Ch., 2011. NoSQL Databases, Lecture Selected Topics on Software-Technology Ultra-Large Scale Sites, Stuttgart Media University, manuscript, p.149, http://www.christof-strauch.de/nosqldbs.pdf (accessed 30 May 2013).

Vinayak, R., Borkar, V., Carey, M.-J., Chen Li, Ch., 2012. Big data platforms: what's next? *In ACM Cross Road*, No. 1 pp. 44-49.