Opportunities in Big Data Management and Processing

Bela STANTIC [a,1] and Jaroslav POKORNY [b]
[a] *School of Information and Communication Technology, Griffith University, Australia*
[b] *Faculty of Mathematics and Physics Charles University, Prague, Czech Republic*

Abstract. Every day we witness new forms of data in various formats. Some example include structured data from transactions we make, unstructured data as text communications of different kinds, varieties of multimedia files and video streams. To ensure efficient processing of this data, often called 'Big Data', the use of highly distributed and scalable systems and new data management architectures, e.g. distributed file systems and NoSQL database, has been widely considered. However, volume, variety and velocity of Big Data and data analytics demands indicate that these tools solve Big Data problems partially only. To make effective use of such data, novel concepts of management is required, in particular, being able to efficiently access data within a tolerable time not only from the users own domain, but also from other domains which the user is authorized to access. This paper elaborates on some issues related to the efficient management of Big Data, discusses current trends, identifies the challenges and suggests possible research directions.

Keywords. Big Data, Efficient Management, NoSQL, NewSQL

Introduction

The fundamental concept of generating data has changed recently, in the past, several main sources have been generating data and all others have been consuming data. However, today all of us are both generating data and also consumers of this shared data. Every day there new sources of data arise from which valuable information can be extracted, including meteorology, genomics, complex physics simulations, biological and environmental research, different sensory technologies, software logs, cameras, etc. These data sets that are so large and complex, often termed as 'Big Data', is beyond the ability of traditional database management systems to be handled within a reasonable time-frame. There are many challenges including how to capture, curate, store, efficiently search, share, transfer, analyze and visualize the data. To overcome these challenges, there is a need for new architectures, techniques, algorithms, and analytics not only to manage it, but also to extract the value and the hidden knowledge. This new concept of data generation has caused a significant increase in volume as well as in the number of users, and therefore, requires more feasible solutions of scaling in such dynamic environments than is offered by traditional database architectures and has to be replaced using options such

[1]Corresponding Author: Bela Stantic; School of Information and Communication Technology Griffith Sciences, Griffith University, QLD 4222, Australia ; E-mail: B.Stantic@griffith.edu.au.

as massively parallel systems running on tens, hundreds, or even thousands of servers. Additionally, to obtain valuable information, these systems need to manage various data sources and types of data which results in all of this data needing to be linked together. For storing and processing large datasets, users have a number of options associated with the problems mentioned above, they can use:

- traditional legacy and parallel database systems,
- distributed file systems such as Hadoop technologies,
- key-value datastores (so called NoSQL databases),
- new database architectures (e.g., NewSQL databases).

Distributed file systems such as the Hadoop Distributed File System (HDFS), for example, use file partitioning and replications. NoSQL databases are a relatively new type of database which were initiated by Web companies in the early 2000s. Finally, NewSQL databases are aiming to provide the scale-out advantages of NoSQL databases often on commodity hardware. It is also used to maintain the transactional data consistency of traditional RDBMs, in addition to also being compatible with SQL.

Progress and innovation is no longer hindered by the ability to collect data but by the ability to manage, analyze, summarize, visualize, and discover knowledge from the collected data in both an efficient manner and in a scalable fashion. This paper focuses on the approaching challenges with Big Data with the goal of relating the principles of today's data processing tools and methods, and to show some alternatives in this area as well as suggest future directions.

The reminder of this paper is organized as follow: In Section 2 we briefly elaborate on how traditional database management systems handle Big Data. Section 3 identifies and group sources of Big Data and elaborates what has been done to ensure efficient management of data in each group. It also addresses problems not only about the architectures, but also about the indexing methods. In Section 4, we evaluate various approaches to the Big Data Processing and Big Analytics, and in Section 5 we conclude the paper by discussing and identifying open problems and challenges associated with Big Data as well as discuss possible future directions.

1. Traditional Database Management Systems and Big Data

Traditional database management systems, which are commercially available and are in place in most organizations, are disk based SQL oriented Database systems. The data inside these systems is stored in disk blocks, which are heavily encoded for a variety of technical reasons. The performance is improved by the use of a main memory buffer pool and letting the database system move blocks between memory cache and disk. SQL is used for interactions with the data. Query plans are designed to find the best way to execute SQL commands by optimizing CPU usage and disk I/Os. The fundamental operation is normally a read or an update of a table row.

Indexing of the data is normally done by using B-trees in the form of clustered or unclustered indexes. To ensure concurrency control and crash recovery, dynamic row-level locking (ACID) and also write-ahead logs are normally in place (Aries) [1]. Replication, in the case of more nodes, is basically done by first updating the primary node and then moving the log over the network to other sites and rolling forward at the secondary

node(s). This technology, due to the proven capabilities, starting in the nineties when it took over the database market, is considered to fit all requirements.

However, the significant increase in the form of application domains, database sizes, as well as the variety of data began to cause problems for this technology as it started to be too robust and not able to answer the requirements of new demands. A while ago, Stonebraker in his 2006 paper identified this by saying one size does not fit all [2]. He claimed that the three decades of commercial DBMS development can be summed up in a single phrase: "One size fits all". A traditional DBMS architecture, which is originally designed and optimized for business data processing, has been used to support many data-centric applications with widely varying characteristics and requirements. He strongly argues that this concept is no longer applicable to the whole database market and that the commercial world will venture into a collection of independent database engines, some of which may be unified by a common front-end parser.

2. Big Data

Every day we witness new types and sources of data emerge that have the ability to provide useful information. In the early days, the focus was on legacy systems as they were the main source of data, and therefore, database management systems have been tailored for their requirements. However, with the emergence of new sources and data, as well as the need for other types of information, it started to be evident that the traditional database management systems can not satisfy the ever demanding requirements and response times. Big Data is most often characterized by several *V*s:

- *Volume*: data scale in the range of Petabytes, Exabytes and even more. It is estimated that the amount of available data is more than doubling every two years [3]. It is important to highlight that the the amount of information individuals create themselves in form of writing documents, taking pictures, downloading music, etc, is far less than the amount of information being created about them.
- *Velocity*: relates both on how quickly data is being produced and how quickly the data must be processed to meet the demand for extracting useful information. One of the largest sources of high-velocity data are user devices such as smartphones where everything you do is/can be logged. Another form of massive amounts of real-time data is Social media (Twitter, Facebook, or any other social data streams), however, the value of such data degrades over time.
- *Variety*: data is in many format types - structured, unstructured, semi-structured, text (even in different languages), media, etc. A single application can be generating/collecting many types of data. This variety of unstructured data creates problems not only for storage but also for mining and analyzing data. To extract this knowledge, all these types of data need to be linked together.
- *Veracity*: relates to managing the reliability and predictability of inherently imprecise data with noise and abnormality. It also refers to the fact that the data that is being stored, and mined must be meaningful to the problem being analyzed. Veracity in data analysis is the biggest challenge when compared with volume and velocity as it is harder to solve problems to keep your data clean.

A fifth *V* was introduced by Gamble and colleagues and it associates with the *Value*, and indicates if the data is worthwhile and has value for business [4]. Several other *V*s

have been mentioned in the literature, such as *Variability* indicating the different meanings associated with a given piece of data, and *Volatility* addressing the problem of how long data is valid and how long should it be stored or is it no longer relevant to the current analysis and decisions.

When talking about today's data based on the type of data, we can generally divide it into three main groups:

- Online Transaction Processing (OLTP), which is basically all forms of legacy systems driven by ACID properties.
- Online Analytical Processing (OLAP) or Data Warehousing, this group also relies on DBMS however the request for ACID properties can be relaxed due to mostly read only operations and the main attention is given to speed and therefore OLAP's are mostly based on indexes.
- Real-Time Analytic Processing (RTAP), new types of data as well as huge volumes and velocities of such data pose significant demands and it is evident that the traditional database management systems are unable to answer this demand in real time. We are witnessing that this group starts to dominate and is taking over with regard to the size of collected data as well as the useful information it can provide.

In the following sections we will provide some details on what can be done to ensure efficient access to data and information in the above mentioned groups.

2.1. Efficient Management of Data in OLTP

As briefly explained above, OLTP databases rely on data stored in the form of rows in relations. Each relation has a primary key which uniquely identifies the record and also can have one or more secondary indexes, which also can be on one attribute or be composite and consist of more attributes. Most often these indexes are based on B-tree structures with internal and leaf nodes [5]. Despite being proposed more than 30 years ago, the overall performance of the B-tree is considered the best and in most commercial relational databases management systems some forms of B-tree structure is considered as a default index. The B-tree has excellent performance on range queries as well as good performance in exact match queries. Complexity of the B-tree is $log_b\ N$ where b is the blocking factor (the number of records per block) and N is the number of records. It is evident why performance is excellent as in case of blocking factor of 100, which is common, and with million records, it is required to have only 3 reads to find the required record. Additionally, hash based indexes are being used for exact queries as they perform better because there is no need to navigate through internal nodes but just to allocate the appropriate bucket based on hashing function.

OLTP databases also store spatial data which is most often managed by family of R-tree indexes [6]. Efficient Management of temporal data presents a challenge due to the need to manage intervals and several methods, which can be implemented within the commercial databases management systems, have been proposed in the literature such as the RI-tree [7] and VB-tree [8]. Also, attention has been given to the efficient management of multidimensional data such as UB-tree [9] and VG-Curve [10].

2.2. *Efficient Management of Data in OLAP*

Online Analytical Processing (OLAP) including Data Warehousing (DW), which focuses on data storage, is a broader category of business intelligence which also encompasses relational database, report writing and data mining. OLAP relies on a multidimensional data model, which allows complex analytical and ad hoc queries with a rapid execution time. As the performance of DW is very important they are usually entirely built on indexes, in addition to B-tree indexes Bitmap indexes are also utilized [11]. The bitmap representation is an alternate method of the row ID's representation. It uses less space and it is less CPU demanding than the row IDs particularly when the number of distinct values of the indexed column is low. Additionally, the bitmap index improves complex query performance by applying low-cost Boolean operations such as OR, AND, and NOT in the selection predicate to reduce the search space before even going to the primary source of data. To improve the response time, different methods of identifying and materializing views have been also proposed in literature [12].

To further improve the efficiency, it has been suggested to use column stores as they can perform faster than the row stores by the factor of 50 or even 100. Explanation for this is as most warehouses have a central *fact* table which stores details for example "How many items have been sold in particular stores during the certain time period". This fact table has joins with *dimension* tables, which provide details about products, customers, stores, etc. These tables are organized in star or snowflake schema. Performance of the DW can be improved if column store is in place as typical warehouse query reads only several attributes out of maybe 100 that the fact table might have. To answer a query, a *Row* store needs to read all attributes of all rows that satisfy the query criteria. Regardless of how efficient the access method is, it is evident that significant amounts of the data fetched from disk into memory is irrelevant, and therefore, will slow down the system significantly. Due to the longer record size, the blocking factor is smaller and more data blocks need to be accessed. In contrast, *Column* store reads only columns of interest for the particular query and therefore can answer the same query more efficiently as it will be required less physical disk reads. Also, compression is much easier and more productive in a column store. In column store each block has only one kind of attribute. Advantages of column stores have been identified and the first commercially available column oriented database was developed in 1993 followed in 1995 by Sybase IQ[2]. However, since then we witnessed many open source and commercial implementations, including Amazon [3] which is a hosted data warehouse product part of the larger cloud computing platform Amazon Web Services, Google BigQuery[4], Vertica[5], Teradata[6], etc.

The importance of column stores to improve efficiency was also identified by one of the major commercial vendors in relational databases, IBM, which in 2012 released DB2 (code name Galileo) with row and column access control, which enables 'fine-grained' control of the database and data management that moves data to cost effective storage based on how frequently they are accessed. Subsequently, to further improve performance, it added in-memory columnar processing technology with other innovations such as Parallel Vector Processing, Actionable Compression, and Data Skipping.

[2]http://www.sybase.com/products/datawarehousing/sybaseiq
[3]Redshifthttp://aws.amazon.com/redshift/
[4]https://cloud.google.com/products/bigquery/
[5]http://www.vertica.com/
[6]http://www.teradata.com.au/

2.3. *Efficient Management of Data in RTAP*

Real time Analytics is about turning information into knowledge using a combination of existing and new approaches. Volume, velocity and variety of new types of data pose significant demands and it is evident that the traditional database management systems cannot answer this demand in real time. Data from different sources and different formats start to dominate both in volume as well as in content of useful information it can provide. Also, such data is becoming more and more complex, therefore its analysis is also becoming increasingly complex and computational demanding. To exploit this new resource, we need to scale both the infrastructures and techniques.

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*. Time is an integral dimension of data in a stream which influences its processing, i.e. the analyst cannot re-analyze the data after it is streamed. Also, velocity can be a problem, since the value of the analysis (and often the data) decreases with time. If several passes are required, the data has to be put into a DW where additional analysis can be performed. The former can be warehoused in a relatively traditional way, as explained in section 2.2 or processed by systems like NoSQL or NewSQL databases.

2.4. *NoSQL Databases*

NoSQL databases has been utilized for Real-Time Analytic Processing. NoSQL means ”not only SQL” or ”no SQL at all”, which makes this category of databases very diverse. NoSQL solutions are not new as the concept date back to the 1990s. NoSQL provides simpler scalability and improved performance comparing to traditional relational databases. What is principal in classical approaches to databases, a (logical) data model, is described in NoSQL databases rather intuitively, without any formal fundamentals. The NoSQL terminology is also very diverse and the differences between conceptual and database view of data is mostly blurred.

The simplest NoSQL databases called key-value stores (or big hash tables) contain a set of couples (key, value). A key uniquely identifies a value, which is (typically string, but also a pointer to where the value is stored) and a value can be a collection of couples (name, value), e.g. in Redis[7]. This means that the data access operations, typically *get* and *put*, have only a key as the address argument. Though very efficient and scalable, the disadvantage of this very simple data model can be a problem 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 databases store combinations of couples (name, value) collected into collections, i.e. rows addressed by a key, which are basically column NoSQL databases. New columns can be added to these collections. There is even further level of structure (e.g., in CASSANDRA[8]) called super-columns, where a column contains nested (sub)columns. Data access is improved by using column names in operations *get*, *insert* and *delete*.

[7] http://redis.io/
[8] http://cassandra.apache.org/

The most general data models belong to document-oriented NoSQL databases. They are same as key-value stores but pair each key with an arbitrarily complex data structure similar to a document. The JSON (JavaScript Object Notation) format is usually used for 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 precisions. JSON is similar to XML but it is smaller, faster and easier to parse than XML. CouchDB[9] is based on JSON. MongoDB[10] uses BSON (Binary JSON). Its types are a superset of JSON types. Querying document data is possible by other means than just a key (selection and projection over results are possible). In general, NoSQL databases can be classified in the following groups:

- key-value stores: SimpleDB[11], Redis, Memcached[12], Dynamo[13], Voldemort[14]
- column-oriented: BigTable [13], HBase, Hypertable[15], CASSANDRA, PNUTS [14]
- document-oriented: MongoDB, CouchDB

To improve performance, some NoSQL databases are in-memory databases, which means that the data is stored in computer's memory to achieve faster access. The fastest NoSQL data stores such as Redis and Memcached are entirely served from memory. Additionally, graph databases are often considered as a category of NoSQL. They offer graph partitioning strategies for large data, graph query languages designed for particular types of graph data, and efficient evaluation of queries (graph traversals, sub-graph and super-graph queries, and graph similarity evaluation).

Transaction processing in traditional RDBMS is based on ACID properties, which can be considered as a strong consistency. In the context of NoSQL databases, ACID properties are not implemented fully, databases can be only be eventually consistent or weakly consistent. In principle, if we abandon strong consistency, we can reach better availability which will highly improve database scalability. It can be in accordance to database practice where ACID transactions are also required only in certain use cases.

NoSQL databases described in the list of well-maintained Web site[16] includes at least 150 products including object-oriented, XML, and others.

A more general category of parallel DBMSs called NewSQL[17] databases are designed to scale out horizontally on shared nothing machines while ensuring transparent partitioning, still guaranteeing ACID properties, and employing lock-free concurrency control. Applications interact with the database primarily using SQL. NewSQL provides performance and scalability not comparable with traditional DBMS and with Hadoop as well. For example, a comparison of Hadoop and Vertica has shown that query times for Hadoop were a lot slower (1-2 orders of magnitude). A lot of success of some NewSQL solutions lie in the new database architectures, particularly in the in-memory approach. For example, MemSQL can be considered the fastest NewSQL both in data loading and

[9] http://couchdb.apache.org/
[10] https://www.mongodb.org/
[11] http://aws.amazon.com/simpledb/
[12] http://memcached.org/
[13] http://aws.amazon.com/dynamodb/
[14] http://www.project-voldemort.com/voldemort/
[15] http://hypertable.org/
[16] http://nosql-database.org/
[17] http://451research.com/report-long?icid=1651

query execution times. In the context of Big Analytics, most representatives of NewSQL are suitable for real-time analytics (e.g., ClustrixDB, Vertica, and VoltDB) or columnar storage (Vertica). But in general, their performance is still a problem.

2.5. Usability of NoSQL Databases and Hahoop

As NoSQL databases have little or no use of data modelling, developers generally do not create a logical model, thus the database design is rather query driven, data is unconstrained, and there is no standard query language. Some NoSQL databases are more mature than others, however all of them are trying to solve similar problems. Also, there is an issue with migration from one NoSQL system to another, as such, conversion is complicated. Although belonging to the same category, two NoSQL tools can be very different. For example, CouchDB uses a low-level query language and scalability through replications. Mongo owns a rich declarative query language and its scalability is achieved through sharding. NoSQL databases can be useful in applications which do not require transactional semantics such as what is found in address books, blogs, or CMS. They are appropriate for indexing a large number of documents, serving pages on high-traffic websites, delivering streaming media, managing data such as one typically occurring in social networking applications.

Therefore, NoSQL is inadvisable for applications requiring enterprise-level functionality (ACID, security and other features of RDBMS technology). However, overall, NoSQL databases could be a good platform for Big Analytics such as analyzing high-volume, real time data. For example, Web log analysis and Web-site click streams analysis belong to highly parallelizable problems which can be efficiently addressed with NoSQL databases.

On the other hand, e.g. column-oriented NoSQL databases are not suitable for the majority of DW/BI applications or even many traditional web-based applications. They have only a few facilities for ad-hoc query and analysis and even a simple query requires significant programming expertise. HBase albeit enables fast analytical queries, but only on column level.

In relation to the Hadoop, many NoSQL databases are built on top of the Hadoop core, i.e. their performance depends on MapReduce jobs. However, MapReduce is still a very simple technique compared to those used in the area of distributed databases. MapReduce is well suited for applications which analyze 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.

Problems with NoSQL concern the whole architecture. For example, M. Stonebraker[18] points out, that the Hadoop stack has a poor performance except where the application is “embarrassingly parallel”. The reasons for it having a background in no indexing, very inefficient joins in Hadoop layer, “sending the data to the query” and not “sending the query to the data”. Another example is Couchbase[19] which replicates the whole document if only a small part is changed.

[18] http://istc-bigdata.org/index.php/no-hadoop-the-future-of-the-hadoophdfs-stack/
[19] http://www.couchbase.com/

In the Big Data world, 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. 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 of Big Data used there. However, as we consider more complex cloud architectures, NoSQL should not be the only option in the cloud.

2.6. NoSQL - Indexing

NoSQL databases are generally based on the key-value store model which efficiently supports the single-key index and can respond to queries in milliseconds. To achieve this performance, different NoSQL databases utilize different indexing methods. For example, CouchDB has a B-tree index which is a bit different to the original. While it maintains all of the important properties, it adds *Multi-Version Concurrency Control* and an append-only design. B-trees are used to store the main database file as well as view indexes. One database is one B-tree, and one view index is one B-tree. MongoDB provides different types of indexes for different purposes and various content types. Single field indexes only include data from a single field of the documents in a collection and supports single field indexes on fields at the top level of a document and on fields in sub-documents, in addition to *Compound* indexes. *Multikey* indexes are also supported which reference an array and records a match if a query includes any value in the array. To improve efficiency, an index can be created as sparse, and in this case, the indexes will not include documents that do not have the indexed field.

However, numerous applications require multi-dimensional queries, which NoSQL databases do not support efficiently. Some solutions of NoSQL database that respond to multi-dimensional query are MapReduce and Table-Scan, but they are inefficient and costly, especially when the selectivity of the query request is low. Multi-dimensional index technology has been extensively studied in traditional DBMSs. However, they cannot meet the increasing requirements of scalability and high I/O throughput in the context of big data, therefore distributed multi-dimensional index for big data management on cloud platform management has been introduced. They can be generally grouped into two categories according to the associated distributed storage architecture:

- Peer-to-Peer system: Several methods have been proposed in the literature including Wu and colleagues who proposed a general index framework with global and local indexes. The local index is built on local data while the index framework selects nodes from local indexes by a respective cost model and organizes them together to form the global index [15]. There is the RT-CAN proposal which employs R-trees for local indexes and is based on CAN [16] protocol, [17] while QT-Chord has a local index based on a quad tree [18].
- Master-slave style: EMINC is a two-layered index with R-Tree for a global index which is stored in master node, while slave nodes local data are indexed with K-d tree [19]. MD-HBase utilizes Z-order curve and K-d tree or Quad-Tree [20]. EDMI in the global layer employs K-d tree to partition entire space into many subspaces and the local layer contains a group of Z-order prefix R-trees related to one subspace [21].

3. Big Analytics

Big data analytics is the process of analyzing large amounts and different types of data to uncover hidden patterns, correlations and other useful information. Big analytics could accomplish much more than what can be done with smaller datasets. For example, Big Analytics allows move beyond linear approximation models towards complex models of greater sophistication because small datasets often limit our ability to make accurate predictions and assessments. Also, it allows predictive analyses utilizing techniques such as data mining and statistical modeling. The use of such techniques coupled with Big Data vastly improves predictability and scoring. Additionally, Big Data significantly improves the ability to locate and analyze the impact of rare events that might escape detection in smaller data sets. Access to larger datasets, affordable high-performance hardware, and new powerful analytical tools provide means for better accuracy in predictions. Big analytics is a genuine leap forward and a clear opportunity to realize enormous gains in efficiency, productivity, revenue, and profitability [22]. 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, explicitly managed time dimension),
- new programming models,
- machine learning and statistical methods,
- complex systems architectures,
- information visualization.

Big Data is often mentioned only in context with Business Intelligence (BI), however, not only BI developers but also e-scientists analyze 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 considering the special nature of Big Data. It is important to emphasize that Big Analytics does not only involve the analysis and modelling phase. For example, noisy context, heterogeneity, and interpretation of results are also necessary to be taken into account. All these aspects influence scalable strategies and algorithms, therefore, more effective preprocessing steps (filtering and integration) and advanced parallel computing environments are needed.

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 them. Graph pattern matching is commonly used in social network analysis, where the graph involves, e.g. a billion of users and hundreds billions links. Traditional analytic methods are able only to cope with small data sets and require expensive configurations of hardware and software to run complex analyses. In any case the main problems of current data mining techniques applied on Big Data come from their inadequate scalability and parallelization.

4. Concluding Remarks

It is evident that ”one size does not fit all”. Different types of data and different requirements pose different challenges and needs, which can be only efficiently answered with different techniques and different concepts. This is an interesting time in databases, there is a lot of new ideas as well as products. Commercial traditional database management systems in the past successfully managed to add features to address different trends and needs, and as a result they became too large and too resource demanding. However, despite all the challenges, none of the proposed concepts will in the near future guarantee requirements that most OLTP application domains require. If a new product satisfies all requirements, it will be most likely also too complex and resource demanding and at the same time not yet proven enough, and as a result, most likely will not be widely accepted. Therefore, traditional database management systems are here to stay for OLTP applications, at least for a while. However, they should move toward the main memory systems and therefore current indexing methods will need to evolve to main memory search.

In relation to OLAP, answering needs for Business Intelligence, data warehouses will be bigger and bigger. To be able to efficiently work with this volume of data, DW will need to be column store based and able to run on dynamic scaling architectures. We are already witnessing this change. Such a concept could successfully compete in the data warehousing market.

In relation to Real-Time Analytic Processing, it is expected that the changes will be driven by requests of complex analytics, which are able to predict future not just provide valuable information from data. This will be possible as more and more data from different sources that contains useful information will be captured. One of the new sources will be the “internet of things” along with all other sources of Big Data. The challenge will be to efficiently process all that data and to extract useful information in real time. NoSQL databases will only be utilized in applications with schema-later concepts and obviously where ACID properties are not essential. However, we have already witnessed main NoSQL databases such as Cassandra and MongoDB are moving to SQL. They are also moving toward ACID. Initially NoSQL used to mean “No SQL”, then meant “Not only SQL”, and at present most accurate definition would be “Not yet SQL”. Hadoop stack will need to change to something different as it is only good for very small fraction of application, which Stonebraker calls “embarrassingly parallel”, it will need to look different. It is not likely that the current form HDFS will survive as it is very inefficient. New data management architectures, e.g. distributed file systems and NoSQL databases, can solve Big Data problems only partially due to their complexity and analytics demands. NewSQL systems are promising to address these shortcomings. Several promising systems in this category are VoltDB, Hana(SAP), and SQLFire.

To efficiently obtain information in RTAP, efficient distributed multi-dimensional indexing methods will need to overcome communication overhead which is excessive for routing query requests between computer nodes. Also, the current index framework cannot meet the velocity of Big data because communication overhead to synchronize the local index and global index is huge. To overcome these problems most likely completely new concept need to be introduced.

References

[1] Mohan C. Aries/lhs: A concurrency control and recovery method using write-ahead logging for linear hashing with separators. In: *ICDE*, 1993. 243–252.

[2] Stonebraker M. "One Size Fits All": An Idea Whose Time Has Come and Gone. In: *Proceeding ICDE '05 Proceedings of the 21st International Conference on Data Engineering*, 2006. 2–11.

[3] Gantz J., Reinsel D. Extracting value from chaos. In: *IDC iView*, 2011. 1–12.

[4] Gamble M., Goble C. Quality, Trust and Utility of Scientific Data on the Web: Toward a Joint model. In: *Proceedings of the ACM WebSci11*, 2011. 1–8.

[5] Comer D. The Ubiquitous B-Tree. *Computing Surveys*, 1979, 11(2), 123–137.

[6] Guttman A. R-Trees: a Dynamic Index Structure for Spatial Searching. In: *Proceedings of the 1984 ACM SIGMOD international conference on Management of data*, 1984. 47–57.

[7] Kriegel H.P., Potke M., Seidl T. Managing intervals efficiently in object-relational databases. *Proceedings of the 26th International Conference on Very Large Databases*, 2000, 407–418.

[8] Stantic B., Terry J., Topor R., Sattar A. Advanced indexing technique for temporal data. *Computer Science and Information Systems - ComSIS*, 2010, 7(4), 679–703.

[9] Ramsak F., Markl V., Fenk R., Zirkel M., Elhardt K., Bayer R., Forschungszentrum B., Mnchen T. Integrating the ub-tree into a database system kernel. In: *VLDB'00 Proceedings of the 26th International Conference on Very Large Data Bases*, 2000. 263 – 272.

[10] Terry J., Stantic B. Indexing Method for Multidimensional Vector Data. *Computer Science and Information Systems Journal*, 2013, 10(3), 1077–1104.

[11] Spiegler I., Maayan R. Storage and retrieval considerations of binary data bases. *Information Processing and Management: an International Journal*, 1985, 21(3), 233–254.

[12] Derakhshan R., Dehne F., Korn O., Stantic B. Simulated annealing for materialized view selection in data warehousing environment. In: *Proceedings of the 24th IASTED International Conference on Database and Applications*, 2006. 89–94.

[13] Chang F., Ghemavat J.D.S., et al. Bigtable: A Distributed Storage System for Structured Data. In: *Journal ACM Transactions on Computer Systems (TOCS)- Article No. 4.* , volume 26, 2008.

[14] Cooper B., Ramakrishnan R., et al. PNUTS: Yahoo!'s hosted data serving platform. *Journal Proceedings of the VLDB Endowment*, 2008, 1(2), 1277–1288.

[15] Wu S., Wu K. An indexing framework for efficient retrieval on the cloud. In: *IEEE Data Engineering Bulletin*, 2009. 75–82.

[16] Ratnasamy S., Francis P., Handley M., Karp R.M., Shenker S. A scalable contentaddressable network. In: *SIGCOMM*, 2001. 161–172.

[17] Wang J., Wu S., Gao H., Li J., Ooi B. Indexing multi-dimensional data in a cloud system. In: *SIGMOD*, 2010. 591–602.

[18] Ding L., Qiao B., Wang G., Chen C. An efficient quad-tree based index structure for cloud data management. In: *LNCS, vol. 6897*, 2011. 238–250.

[19] Zhang X., Ai J., Wang Z., Lu J., Meng X. An efficient multi-dimensional index for cloud data management. In: *CloudDB*, 2009. 17–24.

[20] Nishimura S., Das S., Agrawal D., Abbadi A. A scalable multidimensional data infrastructure for location aware services: MD-HBase. In: *CloudDB*, 2011. 7–16.

[21] Wang J., et al. Distributed Multi-dimensional Index for Big Data Management. In: *WAIM 2013, LNCS 7923*, 2013. 130–141.

[22] Nie N.H. The rise of big data spurs a revolution in big analytics. *Revolution Analytics Executive Briefing*, 2011, 1–8.