# MapReduce: Review and open challenges

**Article** *in* Scientometrics · April 2016

DOI: 10.1007/s11192-016-1945-y

CITATIONS
109

READS
4,525

**6 authors**, including:

Ibrahim Abaker Targio Hashem
University of Sharjah
**89** PUBLICATIONS **10,982** CITATIONS
SEE PROFILE

Nor Badrul Anuar
University of Malaya
**206** PUBLICATIONS **12,461** CITATIONS
SEE PROFILE

Abdullah Gani
University of Malaya
**291** PUBLICATIONS **19,228** CITATIONS
SEE PROFILE

Ibrar Yaqoob
Charles Sturt University
**139** PUBLICATIONS **17,453** CITATIONS
SEE PROFILE

# MapReduce: A bibliometric, review and open challenges

Ibrahim Abaker Targio Hashem [a], Nor Badrul Anuar [a], Abdullah Gani [a], Ibrar Yaqoob [a], Feng Xia [b], Samee Ullah Khan [c]

targio@siswa.um.edu.my, badrul@um.edu.my, abdullah@um.edu.my, ibraryaqoob@siswa.um.edu.my, f.xia@ieee.org, samee.khan@ndsu.edu

[a] Mobile Cloud Computing Research Lab, Faculty of Computer Science and IT, University of Malaya, Kuala Lumpur, Malaysia

[b] School of Software, Dalian University of Technology, Dalian 116620, China

[c] NDSU-CIIT Green Computing and Communications, North Dakota State University, Fargo, ND 58108, USA

**Abstract**

The continuous increase in computational capacity over the past years has produced an overwhelming flow of data or big data, which exceeds the capabilities of conventional processing tools. Big data signify a new era in data exploration and utilization. The MapReduce computational paradigm is a major enabler for underlying numerous big data platforms. MapReduce is a popular tool for the distributed and scalable processing of big data. It is increasingly being used in different applications primarily because of its important features, including scalability, fault tolerance, ease of programming, and flexibility. Thus, bibliometric analysis and review was conducted to evaluate the trend of MapReduce research assessment publications indexed in Scopus from 2006 to 2015. This trend includes the use of the MapReduce framework for big data processing and its development. The study analyzed the distribution of published articles, countries, authors, keywords, and authorship pattern. For data visualization, VOSviewer program was used to produce distance- and graph-based maps. The top 10 most cited articles were also identified based on the citation count of publications. The study utilized productivity measures, domain visualization techniques and co-word to explore papers related to MapReduce in the field of big data. Moreover, the study discussed the most influential articles contributed to the improvements in MapReduce and reviewed the corresponding solutions. Finally, it presented several open challenges on big data processing with MapReduce as future research directions.

**Keywords** big data, MapReduce, Hadoop, bibliometric

## Introduction

In the digital information era, big data have emerged as a research area to precisely tackle the vast amounts of data generated. The term “big data” is chiefly used to describe the volumes, variety, and velocity of data (M. Chen et al., 2014). Such data typically include large amounts of semi-structured and unstructured data formats that are extremely difficult to store, process, and analyze with traditional database technologies. Big data are considered a promising technology (Manyika et al., 2011; McAfee et al., 2012; Rothstein, 2015) and are beneficial for numerous organizations. In particular, telecommunication companies employ big data for monetizing communication traffic data (Yazti & Krishnaswamy, 2014). Manufacturing companies adopt big data to determine an optimal maintenance cycle for replacing component parts before they fail, thereby increasing uptime and customer satisfaction (Zhu et al., 2014). Pharmaceutical firms use big data to accelerate drug discovery and offer highly personalized

medicine (Greenspan & Valkova, 2014). Government agencies apply big data to protect confidential information against cyber-attacks (Kim et al., 2014; Lyon, 2014).

Extracting meaningful and valuable information from huge datasets is important for providing attractive new services and improving the quality of the existing ones (Kambatla et al., 2014; Talia, 2013). The big data stored in a distributed fashion require processing in parallel (Philip Chen & Zhang, 2014), such that new knowledge and innovation can be mined within a reasonable amount of time. Data processing has been successfully adopted in a number of applications (Hsu, 2014), such as data mining, data analytics, scientific computation, and search engine. However, processing big data has been challenged by these applications because of the complexity of the data that should be processed and the scalability of the underlying algorithms that support such processes (Labrinidis & Jagadish, 2012). MapReduce is possibly the most popular framework for processing the existing large-scale data (Dean & Ghemawat, 2008) (Dean & Ghemawat, 2010) primarily because of its important features that include scalability, fault tolerance, ease of programming, and flexibility. Nowadays, MapReduce is primarily used for expressing distributed computations on massive amounts of data and an execution framework for large-scale data processing on clusters of commodity servers. Cluster computing provides high performance on distributed system environments, including computer power, storage, and network communications (Bollier & Firestone, 2010) with emphasis on how cluster computing can provide a hospitable context for data growth.

In this paper, we provide an overview of MapReduce to understand the state-of-the-art research achievements. In view of these achievements, the literature related to big data has grown substantially. An investigation on the process of implementing bibliometric analytics techniques for evaluating the growing body of knowledge on MapReduce research is therefore urgently required. The major advantage of bibliometric analysis is its ability to put a large quantity of studies into perspective. This study aims to quantitatively evaluate the trends of MapReduce-related research literature from 2006 to 2015. Bibliometric methods are employed to examine various publication characteristics, such as countries, authors, publications, research fields, and citations. The academic field has been aggressively researching the themes and different aspects of MapReduce from various perspectives. We discuss some of the articles contributed to the improvements in the MapReduce framework for large data sets with parallel distributed algorithms and describe adoption and solutions that aim to alleviate the current problems. We also describe several research challenges that grant future research directions, with emphasis on presenting insights into the future of big data processing.

The remainder of this paper is organized in seven sections. Following the Introduction, Section 2 provides the background information on MapReduce programing model and Hadoop. Section 3 presents a research method used to develop this biometrics analysis. Section 4 provides the findings of our analysis. Section 5 discusses the most influential articles contributed to the improvements in the MapReduce framework. Section 6 highlights the open challenges encountered in applying this programming framework. Finally, Section 7 concludes the study.

## Background

### MapReduce programming model

MapReduce (Dean & Ghemawat, 2008) is a simplified programming model for processing large amounts of datasets pioneered by Google for data-intensive applications. The model is stunningly simple, and it effectively supports parallelism (Lämmel, 2008).

Such a model is adopted through Hadoop implementation, quickly spreading and transforming into a dominant force in the field of big data (Polato et al., 2014). MapReduce enables an unexperienced programmer to develop parallel programs and create a program that can use computers in a cloud. In most cases, programmers are required to execute only two functions, namely, the map (mapper) and reduce functions (reducer), which are commonly utilized in functional programming. The mapper regards the key/value pair as input and generates intermediate key/value pairs, and the reducer merges all pairs associated with the same (intermediate) key and then generates an output. The map function is applied to each input (key1, value1), in which the input domain is different from the generated output pairs list (key2, value2). The elements of the list (key2, value2) are then grouped by a key. After grouping, the list (key2, value2) is divided into several lists [key2, list (value2)], and the reduce function is applied to each list [key2, list (value2)] for generating a final result list (key3, value3).

## Hadoop

Hadoop (Hadoop, 2011) (Shvachko et al., 2010) is an open-source Apache Software Foundation (ASF) project written in Java that provides cost-effective, scalable infrastructure for the distributed processing of large datasets across clusters of commodity. Hadoop was "inspired" by Google File System GFS (Ghemawat et al., 2003) and Google's MapReduce distributed computing environment. Initially, started as a distributed Nutch search engine project and named by developer Doug Cutting after his son's toy elephant (White, 2009). It has been successfully used for processing highly distributable problems across a large amount datasets using commonly available servers in a very large cluster, where each server has a set of inexpensive internal disk drives. Current Hadoop project consists of three main modules, that is, the distributed file system called Hadoop Distributed File System (HDFS), MapReduce engine, and Yet Another Resource Negotiator (YARN).

### HDFS

HDFS is an open source version of the Google's GFS designed to run on commodity hardware. Like other Hadoop-related technologies, HDFS has become a key tool for managing pools of big data and supporting big data analytics applications and also is a highly fault-tolerant and is designed to be deployed on low-cost hardware. HDFS is a Master/Slaver architecture illustrated in Fig. 1, consisting of NameNode called master, Secondary NameNode called checkpoint, and several DataNode called slaves. NameNode is the controller that handles all file system operations; hence, any request that comes to the file system such as create, delete, and read a file goes through it. The meta-data are present in the NameNode, which registers attributes such as access times, modification, permission, and disk space quotas. NameNode also handles block mappings. In particular, the file is divided into blocks (Default 64MB), in which each block is independently replicated across DataNode for redundancy and periodically sends a report of all existing blocks to the NameNode. DataNode manages the block creation, deletion, and replication upon the instruction from the NameNode. The cluster may incorporate thousands of DataNode and tens of thousands of HDFS clients per cluster given that each DataNode may concurrently execute multiple application tasks.

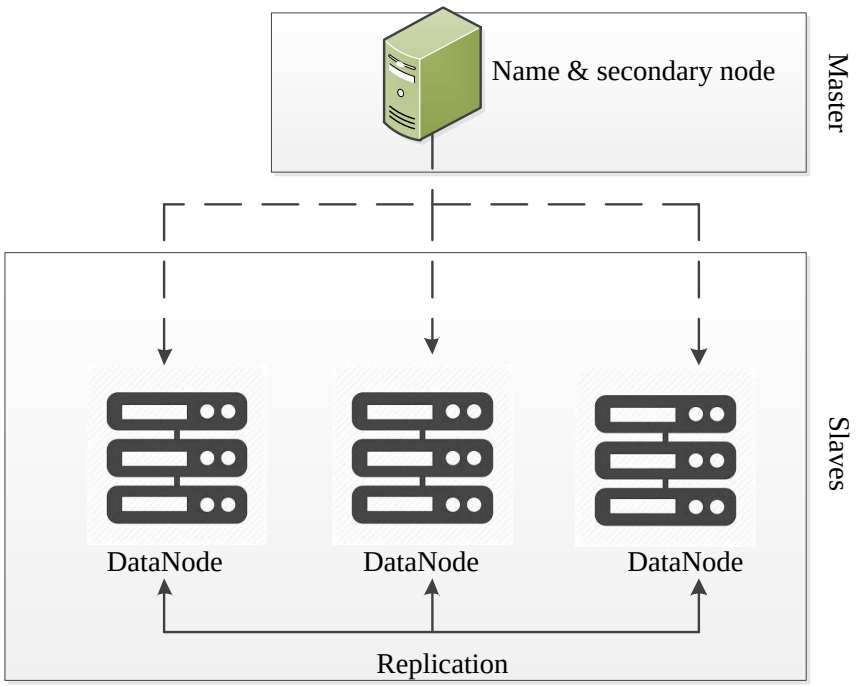


**Fig. 1** HDFS Architecture

Moreover, DataNode sends a Heartbeat message to the NameNode periodically, the default heartbeats interval is three seconds. If the NameNode is unable to detect heartbeat message due to loss of connectivity between the NameNode and DataNode. The NameNode consider to be out of service and the block replicas hosted by that DataNode to be unavailable or dead and does not forward any new requests to that particular DataNode. The NameNode then schedules creation of new replicas of those blocks on other DataNode. Meanwhile, the job of the Secondary NameNode is not to be secondary to the NameNode, but only to periodically read the file system changes log and provides backup for the former, thereby updating it. In larger clusters environment, Secondary NameNode usually run on a different machine than the primary NameNode since its memory requirements are on the same order as the primary NameNode. This task enables NameNode to start up faster next time.

**Hadoop MapReduce**

MapReduce engine comprises several components as shown in Fig. 2. In particular, the main component is job client, which submits the job to the clusters. Job tracker oversees the task tracker and provides execution plans, coordinates the jobs, and schedules them across the task tracker. Meanwhile, task tracker breaks down the jobs into Map and Reduce tasks. Each task record has slots for execution map, gradually reduces, and reports the progress.

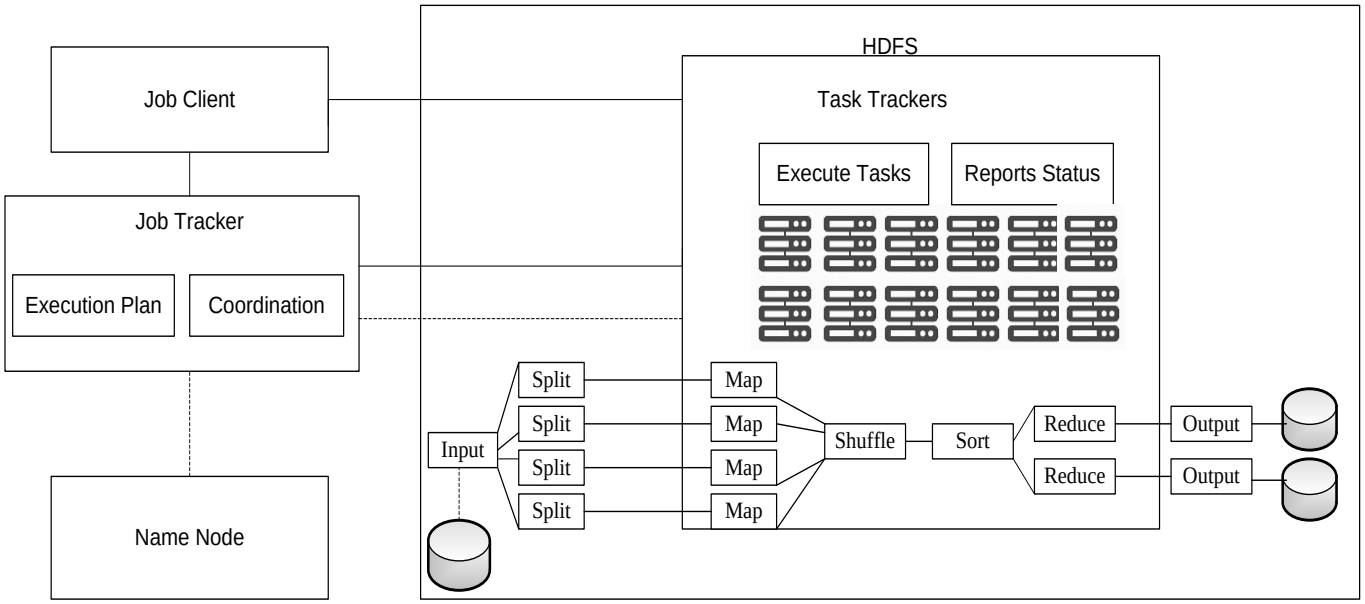


**Fig. 2** MapReduce architecture

Then the input data are divided into input splits based on the input format. Input splits equate to a map task, which runs in parallel. Input format determines how the files are parsed into the MapReduce pipeline. Map transforms the input splits into intermediate key/value pairs based on user-defined code. Shuffle and sort: moves intermediate key/value pairs outputs to the reducers and sorts them by the key. The reducer merges all pairs associated with the same (intermediate) key and then generates an output based on the user-defined code.

**YARN**

YARN (Vavilapalli et al., 2013) is a resource manager that represents a generational shift in the architecture of Apache Hadoop. It utilizes the MapReduce programming framework by default to perform efficient data processing by separating the processing engine and resource management capabilities of MapReduce; hence, it makes the Hadoop environment highly suitable for operational applications that cannot wait for batch jobs to be completed. This feature simplifies the support of maintaining a multi-tenant environment, managing and monitoring workloads, implementing security controls, and providing high-availability features for Hadoop. Moreover, YARN enables programmers to design and implement distributed frameworks while sharing a common set of resources and data (Murthy et al., 2013). Resource manager, the central entity of YARN, employs a node manager to launch containers that could either map or reduce tasks and monitor the operations of individual cluster nodes. The resource manager consists of two interfaces, namely, clients submitting applications and application masters who dynamically negotiate with an access to resources and others toward node managers. Application masters codify their need for resources in terms of one or more resource requests, each of which tracks a number of containers (Vavilapalli et al., 2013). A period from 2006 to 2015 was chosen as the time frame for the literature search to reflect this research area.

## Research method

This study focuses on the present status and future development trends of MapReduce research; thus, content analysis covering Scopus bibliographic database encompassing abstracts, titles, keywords, affiliations, citations, countries, and authorship for academic publications is considered to provide a quantitative description (Mao et al., 2015).

### Research objectives

This paper aims to investigate MapReduce research performance for the period of 2006 to 2015. To achieve this aim, the following objectives must be accomplished:

To investigate the global trends of publications and citations in MapReduce research for large analytical processing;

To evaluate MapReduce research performance across multidisciplinary fields;

To identify authorship patterns and global collaboration networks;

To provide researchers and practitioners with a basis for enhanced understanding of the global development of research; and

To discuss some of the related MapReduce improvements based on the selected papers from the literature.

### Data collection

The data employed in this study were obtained from the databases of Scopus. Scopus is a popular academic database for literature reviews and research analyses (Falagas et al., 2008). Its databases on different scientific fields are more extensive than Web of Science; these databases are frequently used for searching the literature (Meho & Yang, 2007). We collected bibliographic records published from 2006 to 2015 that were pertinent to MapReduce research. The use of keywords, such as “Hadoop”, “MapReduce”, “Hadoop MapReduce”, “Apache MapReduce”, “Hadoop distributed file system”, and “Apache Hadoop”, generated papers that mentioned “MapReduce” , “ Hadoop”, and “big data”. We excluded the term “big data” from our search because it returned 15,207 documents; some of the documents were beyond the scope of this study. Thus, these topics were limited using Boolean operators AND, OR, and NOT. The MapReduce literature is assumed to be representative of the research documents in Scopus. In Scopus, the articles and cited references in different fields, such as science, social sciences, and art and humanities, are updated weekly (Falagas et al., 2008). The database covers journal articles, conference proceedings, books, book sections, and patents. The sample of the datasets used to demonstrate the analytical approach was selected from 6,481 bibliographic records published from 2006 to 2015. The sample comprised journals, conference articles, books, book sections, patents, and review papers. The parameters used in the analysis included author, title, document type, publication year, language, country, institution, citations, and global collaboration.

## Findings

### Bibliometric analysis

Fig. 3 illustrates the pie graph of research fields ranked by the percentage of publications. The articles contributed by computer science were the most frequently published or in-press articles, accounting for 60% of the total literature. By contrast, articles contributed by engineering comprised 16% of the total literature. The distributions of articles published in other fields are summarized in Fig. 3. A significant exponential increasing trend is evident in the number of articles published by computer science during the period of 2006 to 2015.

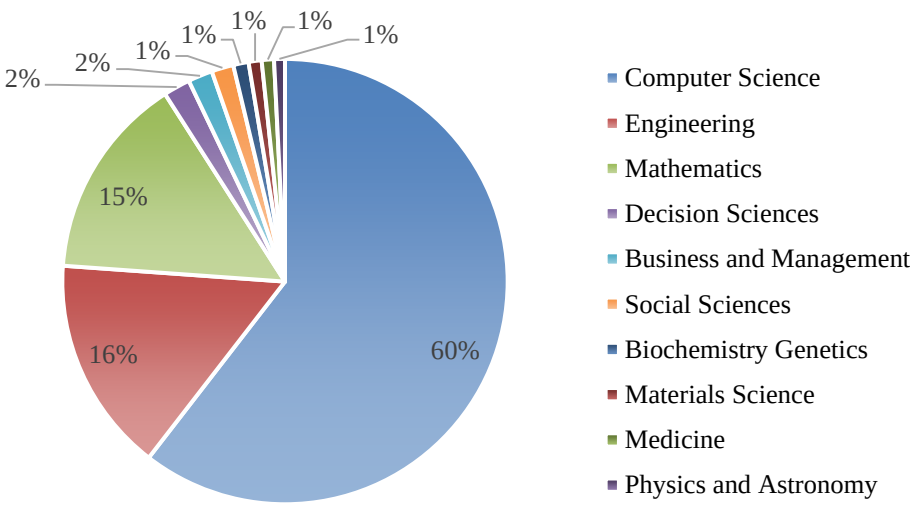


**Fig. 3** Research fields ranked by the percentage of publications

## Output trends

A total of 6,481 documents (i.e., published in various conferences, journals, books, book sections, and patent sources) was found across MapReduce research. Fig. 4 depicts the growth in the number of papers published in 2006–2015. The number of publications in 2006, 2007, and 2008 was small, but it progressively increased between 2013 and 2014. The largest number of papers (i.e., 1,649) was evident in 2014, and a subsequent significant growth was noted in 2013 and 2014. From Fig. 4 it can be found that the number started to decline in 2015 down to 1,178. However, the number of publications is expected to increase in 2016 and likely to be continuing in the future. Given the importance of MapReduce framework for big data processing, this increasing trend reflects the attention to the challenges of big data processing among the researchers. English was the predominant language of articles, except for a few articles in Chinese (162), Turkish (4), Japanese (4), Russian (2), French (1) and other languages. According to previous bibliometric result studies carried out by (Fu et al., 2013) (Sun et al., 2012) concluded that the English is the prime publications language.

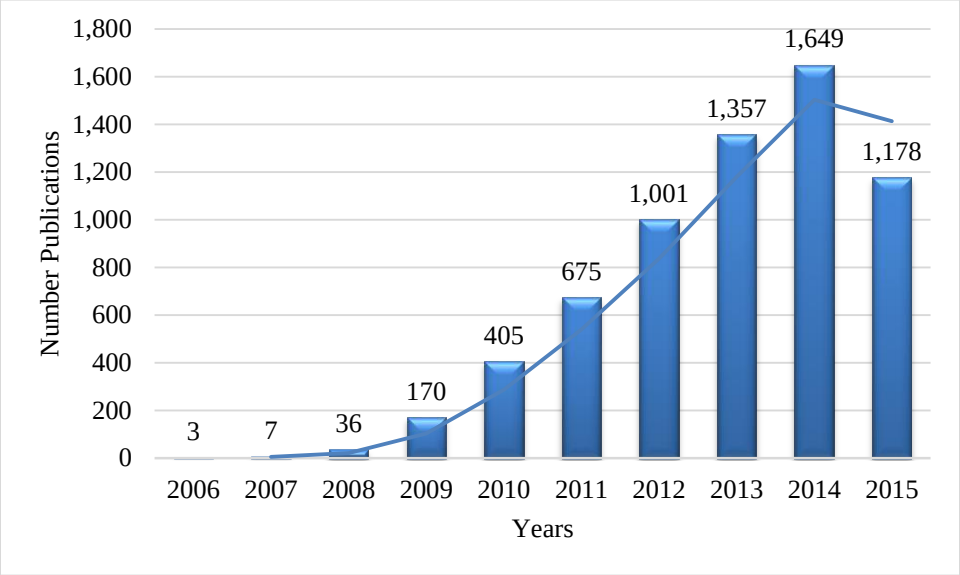


**Fig. 4** Number of publications per year

## Countries

In terms of the contributions of countries, the largest contributor was China, with 2,128 publications over nine years, followed by the United States with 1,810 publications and India with 429 publications. The total number of publications published by each country is shown in Fig. 5.

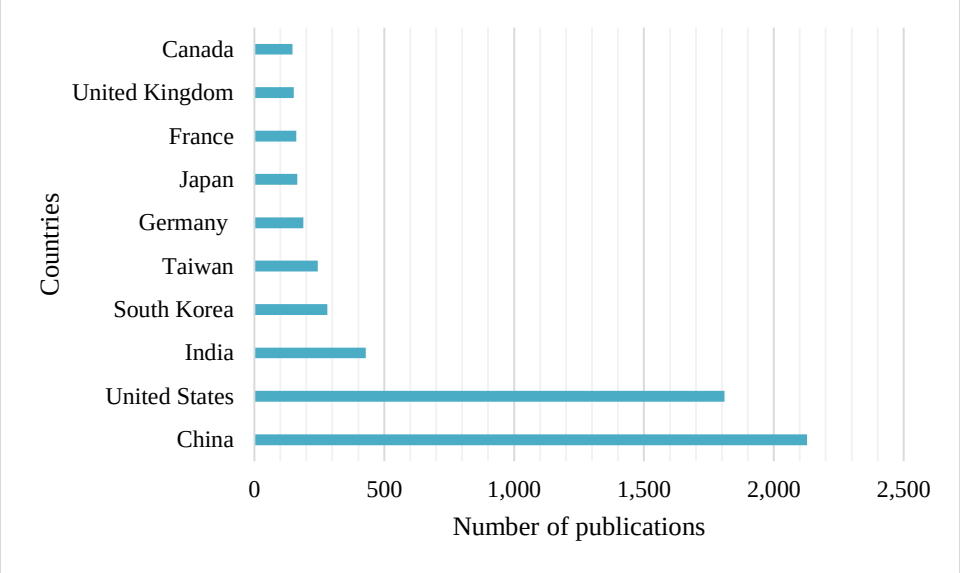


**Fig. 5** Number of publications by countries

In terms of the average number citations per publication from 2006 to 2015 of these 20 countries as shown in Table 1, the United States (12.77) ranked first, followed by Singapore (8.43). Higher h-index value was achieved by the United States (58), followed by China (29) and Germany (15).

**Table 1** Contribution and impact of the top 20 countries in MapReduce research

| Rank | Country | Total number of publications | Total number of citations | Average number of citations per publication | h-index |
|---|---|---|---|---|---|
| 1 | China | 2128 | 4546 | 2.14 | 29 |
| 2 | United States | 1,810 | 23109 | 12.77 | 58 |
| 3 | India | 429 | 340 | 0.79 | 9 |
| 4 | South Korea | 281 | 593 | 2.11 | 11 |
| 5 | Taiwan | 244 | 378 | 1.55 | 8 |
| 6 | Germany | 188 | 1238 | 6.59 | 15 |
| 7 | Japan | 165 | 237 | 1.44 | 7 |
| 8 | France | 162 | 289 | 1.78 | 8 |
| 9 | United Kingdom | 151 | 489 | 3.33 | 12 |
| 10 | Canada | 147 | 486 | 3.30 | 10 |
| 11 | Australia | 112 | 512 | 4.19 | 9 |
| 12 | Italy | 101 | 349 | 3.46 | 11 |
| 13 | Singapore | 94 | 792 | 8.43 | 14 |
| 14 | Spain | 87 | 183 | 2.10 | 7 |
| 15 | Brazil | 82 | 344 | 4.19 | 9 |
| 16 | Greece | 80 | 202 | 2.56 | 8 |
| 17 | Hong Kong | 65 | 385 | 5.92 | 8 |
| 18 | Switzerland | 53 | 154 | 2.90 | 7 |
| 19 | Poland | 46 | 128 | 2.78 | 5 |
| 20 | Netherlands | 43 | 187 | 4.35 | 5 |

Conference papers were the most frequently used document type (4,395), followed by journal articles (published or in-press; 1,529). The distributions of other types of publications are shown in Fig. 6. To provide a broader evaluation, all of the documents were selected for further analysis. A decreasing trend was evident in the number of journals compared with conference papers during the period of 2006 to 2015.

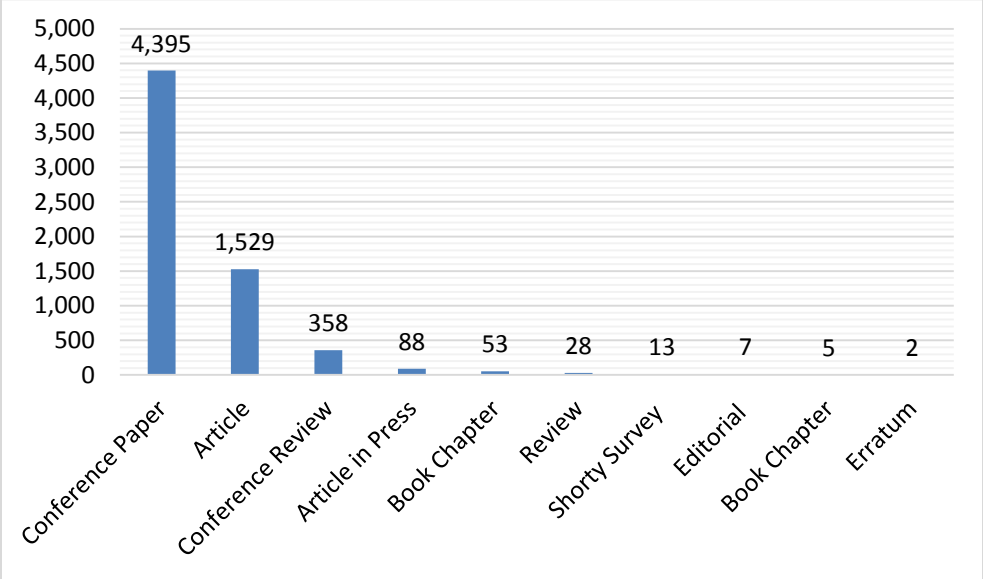


**Fig. 6** Distribution of publication types

## Top cited papers

The 20 highly cited articles are presented in Table 2. The publications were selected based on the keywords used, namely, "Hadoop", "MapReduce", "hadoop mapreduce", "apache mapreduce", and "hadoop distributed file system". These terms represented the MapReduce research as well as big data. Researchers from different fields utilized the MapReduce technologies to manage their large amounts of the data and their complexity. The first publication was cited 3,803 more often than the second highest cited publication, 2,268. (McKenna et al., 2010) provided a critical analysis genome using the MapReduce framework. The second highly cited paper focused on the development of the MapReduce framework for big data processing on large clusters.

**Table 2** Insights from the 20 most-cited articles

| Title | Journal | Year | Citations |
|---|---|---|---|
| MapReduce: Simplified data processing on large clusters | Communications Of The ACM | 2008 | 3803 |
| The Genome Analysis Toolkit: A MapReduce framework for analyzing next-generation DNA sequencing data | Genome Research | 2010 | 2,268 |
| MapReduce: A Flexible Data Processing | Communications Of The ACM | 2010 | 382 |

| | | | |
|---|---|---|---|
| Tool | | | |
| Evaluating MapReduce for multi-core and multiprocessor systems | Conference: 13th International Symposium on High-Performance Computer Architecture Location: Phoenix, AZ | 2007 | 346 |
| HadoopDB: An architectural hybrid of mapreduce and DBMS technologies for analytical workloads | Proceedings of the VLDB Endowment | 2009 | 337 |
| CloudBurst: highly sensitive read mapping with MapReduce | Bioinformatics | 2009 | 283 |
| Mars: A MapReduce Framework on Graphics Processors | Conference: 17th International Conference on Parallel Architectures and Compilation Techniques Location: Toronto | 2008 | 221 |
| Haloop: Efficient iterative data processing on large clusters | Proceedings of the VLDB Endowment | 2010 | 206 |
| Twister: A runtime for iterative MapReduce | HPDC 2010 - Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing. | 2010 | 193 |
| MapReduce and parallel DBMSs: Friends or foes? | Communications of the ACM | 2010 | 191 |
| Hive - A Petabyte Scale Data Warehouse Using Hadoop | Conference: 26th IEEE International Conference on Data Engineering (ICDE 2010) Location: Long Beach, CA | 2010 | 173 |
| Hadoop++: Making a yellow elephant run like a cheetah (without it even noticing). | Proceedings of the VLDB Endowment | 2010 | 168 |
| Document Building a high level dataflow system on top of Map-Reduce: The Pig experience | Proceedings of the VLDB Endowment | 2009 | 149 |
| The performance of mapreduce: An in depth study | Proceedings of the VLDB Endowment | 2010 | 145 |
| MapReduce for data intensive scientific analyses | Proceedings - 4th IEEE International Conference on eScience, eScience 2008 | 2008 | 134 |
| Graph Twiddling in a MapReduce World | Computing In Science & Engineering | 2009 | 132 |
| The Hadoop distributed file system | 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies, MSST2010. | 2010 | 123 |
| Efficient parallel set-similarity joins using MapReduce. | Proceedings of the ACM SIGMOD International Conference on Management of Data | 2010 | 119 |
| MAD skills: New analysis practices for big data | Proceedings of the VLDB Endowment | 2009 | 115 |

| | | | |
|---|---|---|---|
| A comparison of join algorithms for log processing in MapReduce | Proceedings of the ACM SIGMOD International Conference on Management of Data. | 2010 | 109 |

## Keyword co-occurrence analysis

### From 2006 to 2011

The initial term map covering 2006 to 2011 consists of 160 terms and contains only 1297 publications in four clusters. This initial map has shown less terms, although it contents terms used from 2006 to 2011 compare to other maps. MapReduce research is increasingly widening its focus from a previously invention of internet emphasis on data storage and processing. Thus, the result revealed various terms, such as “data intensive application”, “scheduler”, “mapreduce job”, “overhead”, and “resource allocation” (green, left). In Fig. 7 the largest cluster (red right) identifies several terms, such as “mining”, “datasets”, “case study”, “parallel algorithms”, “large dataset”, and “information”. The rest of the clusters focused on various topics, such as “services”, “network”, “storage system”, and “runtime”.

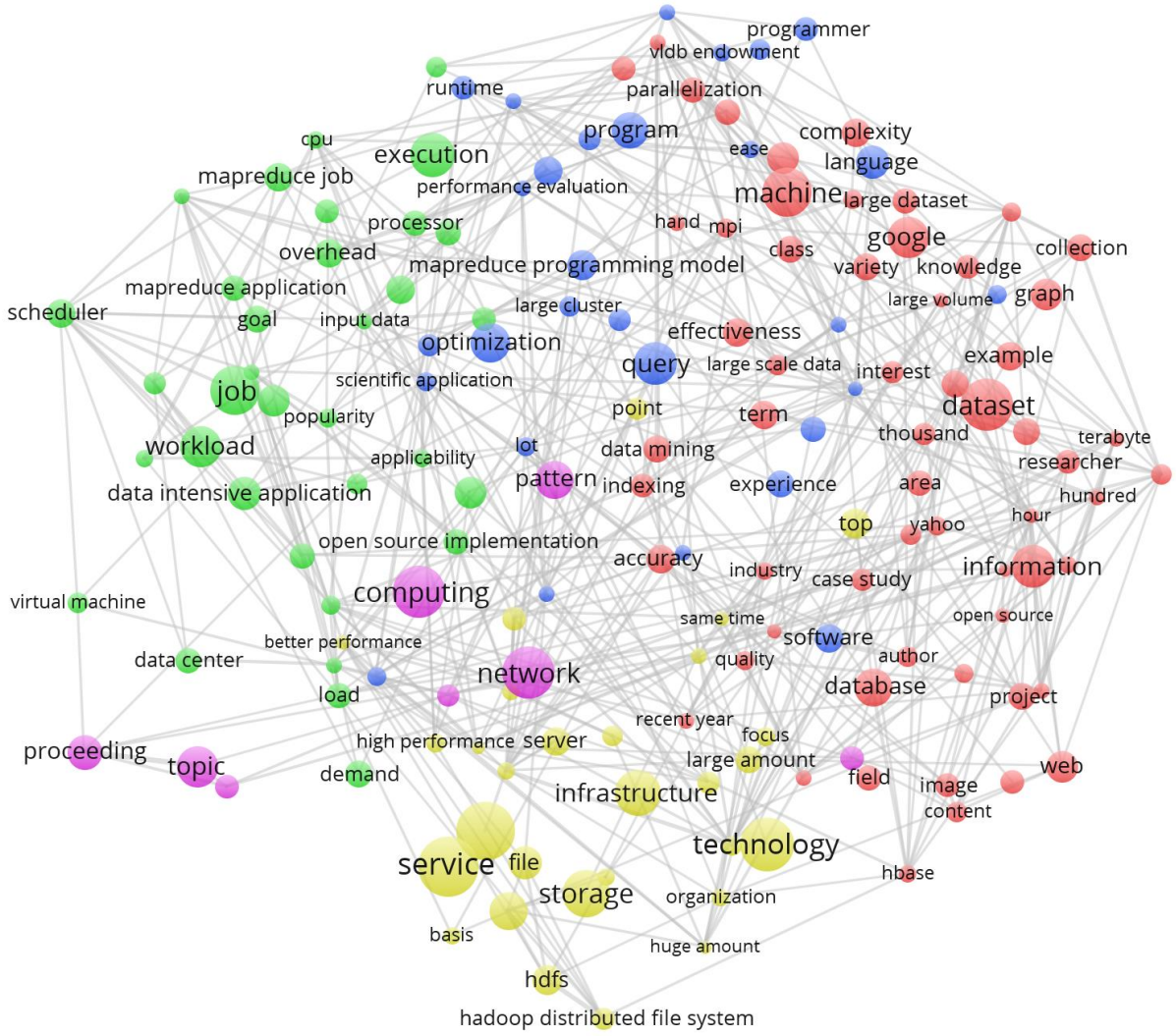

**Fig. 7** Term map of MapReduce research 2006-2010

**From 2012 to 2013**

The next term map (see Fig. 8) covering research publications from 2012 to 2013 comprises 2,358 articles, representing an increase of more than 200% from the previous period. The map presents 290 terms with a minimum number of occurrences of 20 in five loose clusters. Fig. 8 shows the maturing of big data research, particularly using Hadoop MapReduce. Several research areas are identified such as “workload”, “big data”, “programming model”, ”service”, ”information”, and ”storage”.

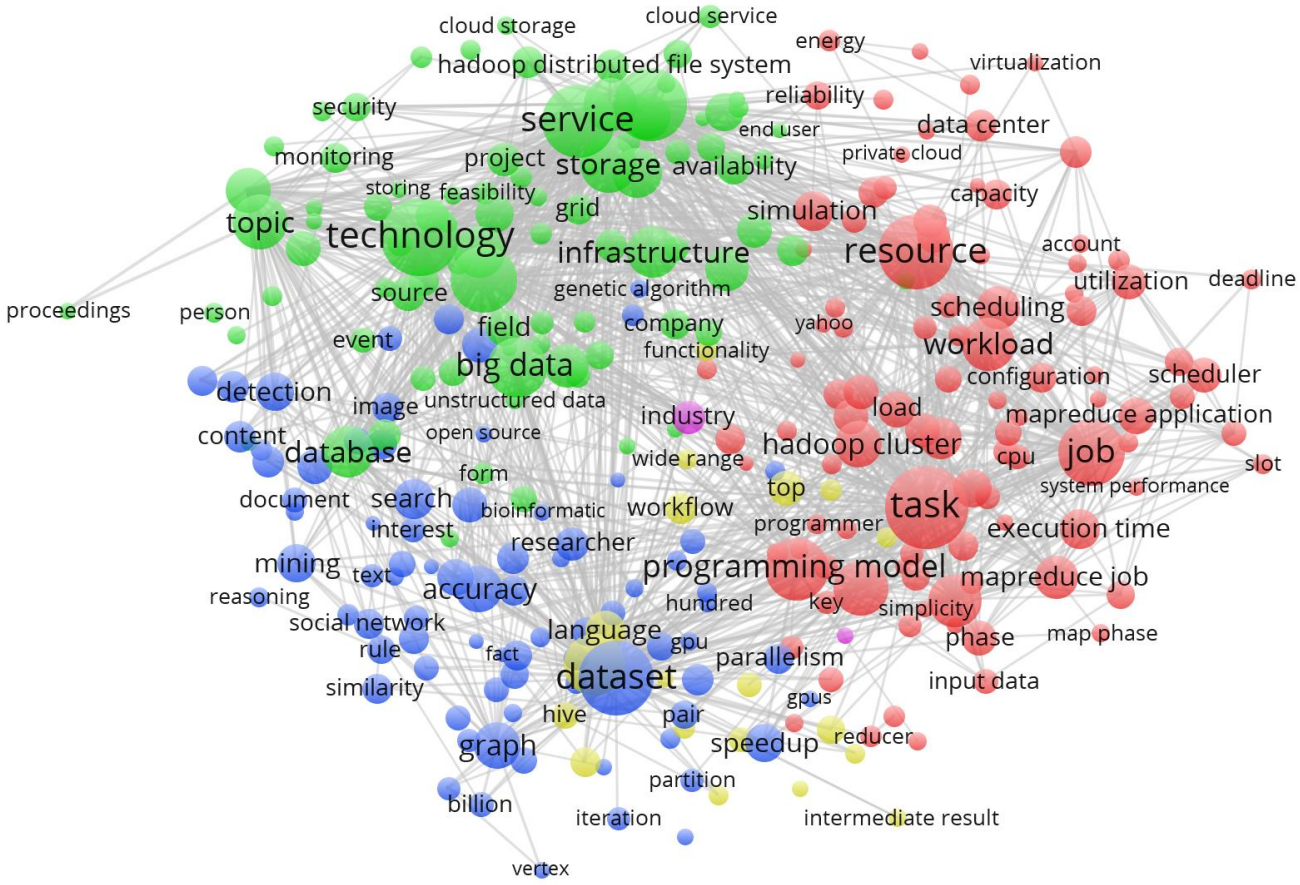


**Fig. 8** Term map of MapReduce research 2012-2013

**From 2014 to 2015**

The map covering 2013 to 2015 (see Fig. 9) contains the largest number of published articles in four clusters. The density and subject matter of this map is the most evolved compare to previous years. Overall, this view of the MapReduce framework is highly established. Other terms can be identified, such as “technology”, “management”, “storage”, “programming model”, “datasets”, “energy consumption”, “query processing”, “cloud storage”, “workload”, “data mining”, “large graph”, “benchmark”, “scheduling”, “big data analytics”, “processing”, “security”, “hdfs”, “cloud’, “data center”, “collaborative filtering”, “video”, “images”, and “server”.

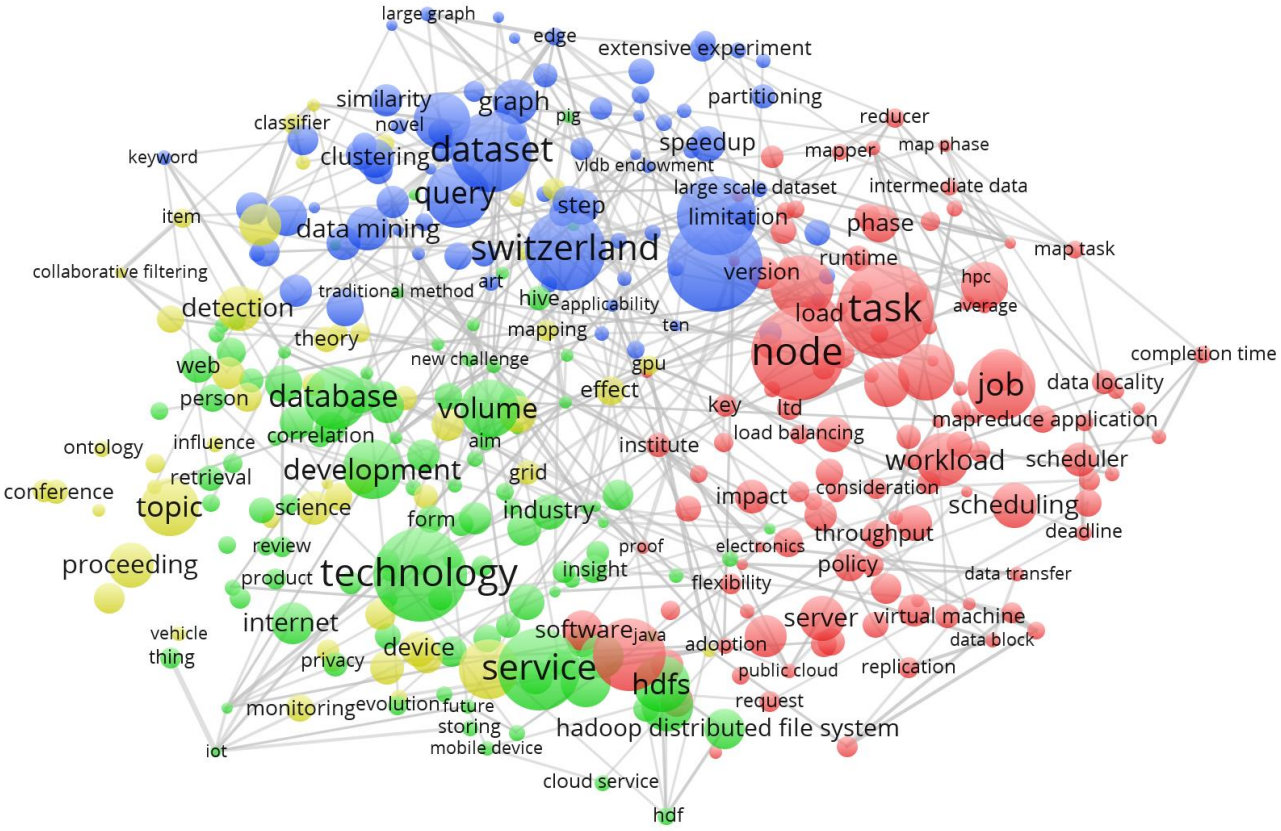


**Fig. 9** Term map of MapReduce research 2013-2015

**From 2006 to 2015**

Fig. 10 depicts the term map covering research articles from 2006 to 2015; this map compresses 6,481 publications. The map presents 734 terms in four loose clusters. It also illustrates the maturing of big data. The research from 2006 to 2015 uncovered the persistence of several research areas, including the performance of big data algorithms and development. Big data research is increasingly widening its focus from an earlier invention of the internet emphasis on architecture, algorithm, and performance of data analytics. This focus is identified with the clustering of various terms, such as “resource”, “scheduling”, “task”, “workload”, and “overhead” (green right). The red left cluster focuses on identifying the current trends of big data research area, including terms such as “mining”, “personal information”, “healthcare”, “city”, “governance”, “sharing”, and “privacy”. The rest of the clusters focused on related topics (blue, bottom right), such as “classification”, “accuracy”, “image”, “learning”, “bioinformatics”, “massive datasets”, and “parameter”.

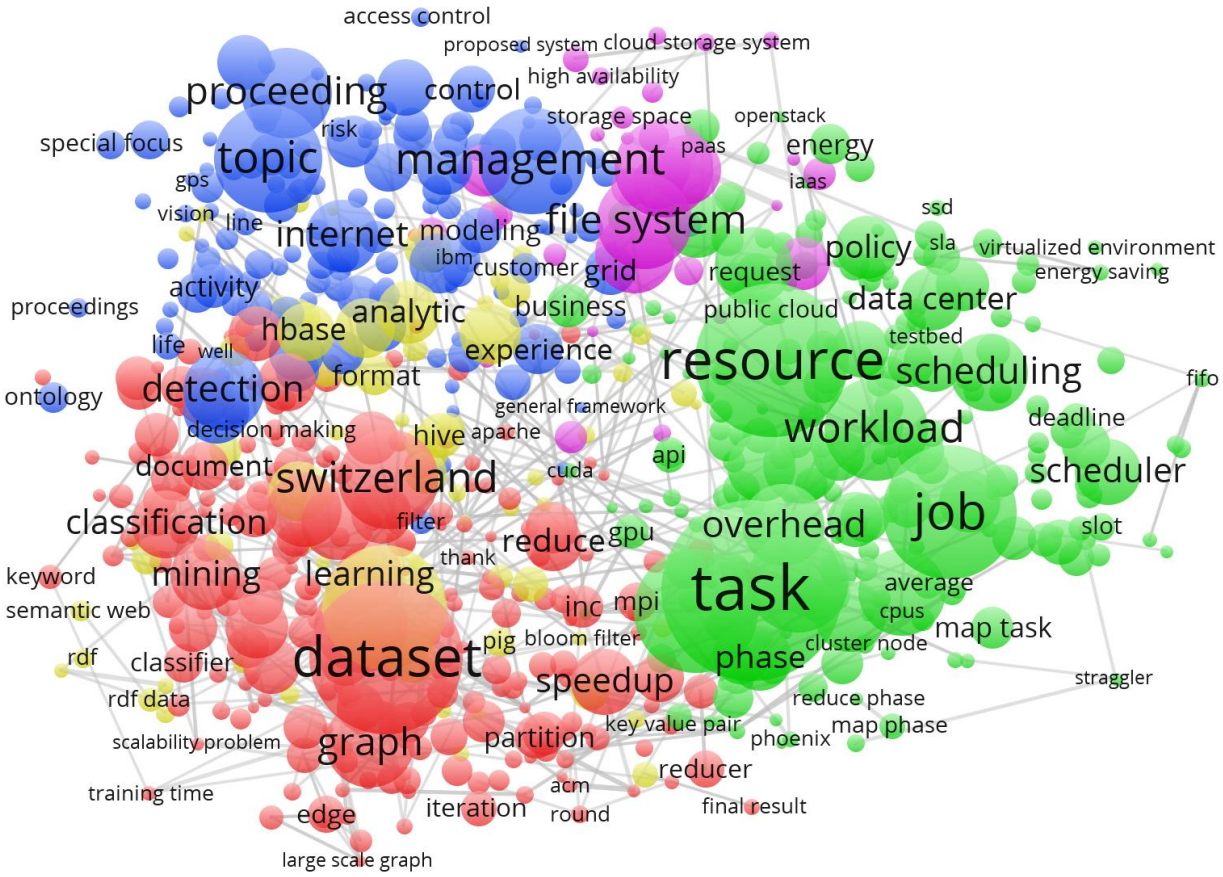


**Fig. 10** Term map of MapReduce research 2006-2015

The results of the frequency analysis of data provide the foundation for analyzing the key topics and aspects representing a respective research year. The keywords used in the paper titles and abstracts are a summarized in Table 3. The emergence of new popular topics can be quickly identified by examining the occurrence of keywords within a specified time frame. Analyzing the co-occurrences of keywords helps identify topics or aspects that are strongly related to each other. The bibliometric study extracts 5,000 unique keywords from the titles and abstract fields of the collected source articles. Note that keywords are always specified by the authors.

**Table 3** Shows most popular keywords from the list of the keywords.

| No | Top keywords | Frequency |
|---|---|---|
| 1 | Map-reduce | 2737 |
| 2 | MapReduce | 1899 |
| 3 | Hadoop | 1410 |
| 4 | Cloud computing | 1335 |
| 5 | Big data | 894 |
| 6 | Algorithms | 883 |

| | | |
|---|---|---|
| 7 | Data handling | 599 |
| 8 | Digital storage | 511 |
| 9 | Distributed computer systems | 482 |
| 10 | Data processing | 440 |
| 11 | Data mining | 426 |
| 12 | Scheduling | 362 |
| 13 | Information management | 338 |
| 14 | Parallel architecture | 333 |
| 15 | Optimization | 275 |
| 16 | Clustering algorithms | 272 |
| 17 | Social networking | 247 |
| 18 | Internet | 222 |
| 19 | Query processing | 209 |
| 20 | Artificial intelligence | 206 |

## Most productive authors

As shown in Table 4, the top 25 authors extensively contributed to big data research. Each author listed in Table 4 published at least 20 articles, and together they contributed 1,247 articles to the datasets. The author with the largest number of documents 73 is wang y., followed by li y. with 72 documents. However, the latter co-authored more articles (i.e., 123) than wang y. (i.e., 109).

**Table 4** The most productive authors

| No | Author | Documents | Co-authorships | Cluster |
|---|---|---|---|---|
| 1 | wang y. | 73 | 109 | 2 |
| 2 | li y. | 72 | 123 | 9 |
| 3 | liu y. | 70 | 98 | 9 |
| 4 | zhang x. | 59 | 94 | 5 |
| 5 | zhang j. | 58 | 88 | 4 |
| 6 | zhang y. | 56 | 84 | 1 |
| 7 | li j. | 55 | 82 | 4 |
| 8 | chen y. | 54 | 63 | 5 |
| 9 | li x. | 53 | 99 | 11 |
| 10 | wang x. | 53 | 86 | 4 |
| 11 | wang j. | 52 | 59 | 1 |
| 12 | wang h. | 50 | 95 | 5 |
| 13 | wang z. | 49 | 94 | 2 |
| 14 | wang c. | 49 | 79 | 1 |
| 15 | liu j. | 47 | 77 | 12 |
| 16 | zhang h. | 45 | 62 | 2 |
| 17 | wang l. | 43 | 67 | 1 |
| 18 | liu x. | 43 | 41 | 2 |

| | | | | |
|---|---|---|---|---|
| 19 | chen j. | 41 | 87 | 5 |
| 20 | zhang z. | 41 | 73 | 7 |
| 21 | xu y. | 40 | 85 | 9 |
| 22 | li z. | 37 | 78 | 9 |
| 23 | zhou x. | 36 | 75 | 5 |
| 24 | yang j. | 36 | 52 | 12 |
| 25 | wang w. | 35 | 74 | 10 |

Fig. 11 depicts a co-authorship network map generated from the authors' publications. The names of some authors Fig. 11 are not visible in order to avoid overlapping labels. Various components, such as author notes, author name, occurrence weight, and network relationship clustering, are included in the map. The document authorship network is a network that expresses the existence of the co-authorship relationship between the authors of scientific papers.

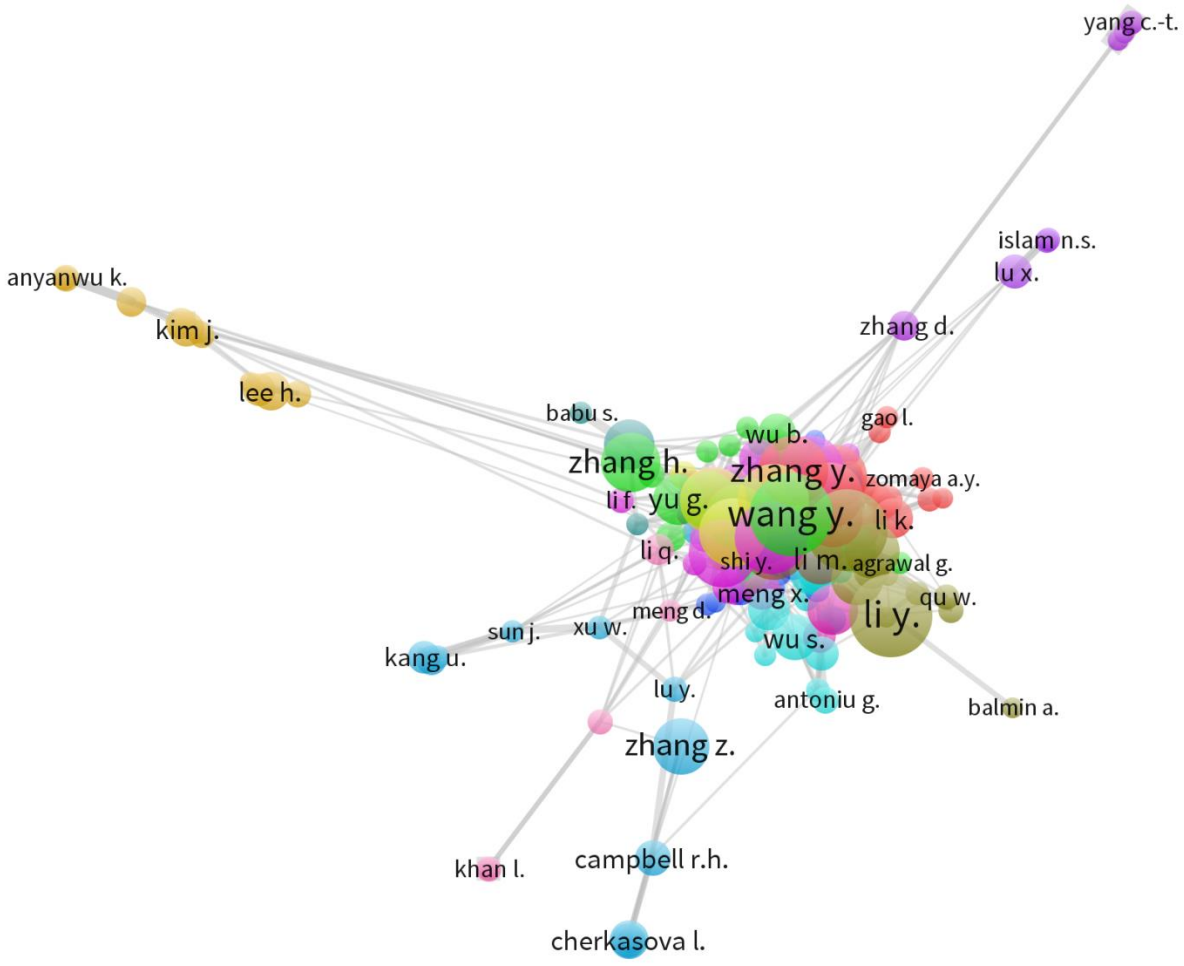


**Fig. 11** Bibliometric mapping of co-occurrence of authors

## Most influential articles contributed to the improvements in MapReduce framework

MapReduce represents the future technology of data processing that may become more dominantly popular than other parallel data processing tools such as DBMS. In this section, we discuss a review of the efforts and proposals that aim to alleviate a few MapReduce problems. We have selected papers contributed to the improvement in a MapReduce framework from Scopus published in 2006 to 2015 (see Table 5).

**Table 5** Most influential articles contributed to the improvement in MapReduce framework

| Category | System | Authors | Time Cited | Year | Publisher |
|---|---|---|---|---|---|
| **Declarative Interfaces** | Pig Latin | (Olston et al., 2008) | 472 | 2008 | Source of the Document Proceedings of the ACM SIGMOD International Conference on Management of Data |
| | Hive | (Thusoo et al., 2009) | 361 | 2009 | Proceedings of the VLDB Endowment |
| | Tenzing | (L. Lin et al., 2011) | 36 | 2011 | Proceedings of the VLDB Endowment |
| | Jaql | (Beyer et al., 2011) | 62 | 2011 | Proceedings of the VLDB Endowment |
| | SQL/MapReduce | (Friedman et al., 2009), | 30 | 2009 | Proceedings of the VLDB Endowment |
| | DUAL Table | (Hu et al., 2014) | 0 | 2015 | Proceedings - International Conference on Data Engineering |
| **Data access** | Llama | (Y. Lin et al., 2011) | 36 | 2011 | Proceedings of the ACM SIGMOD International Conference on Management of Data |
| | Cheetah | (S. Chen, 2010) | 44 | 2010 | Proceedings of the VLDB Endowment |
| | RCFile | (He et al., 2011) | 62 | 2011 | Source of the Document Proceedings - International Conference on Data Engineering |
| | CIF | (Floratou et al., 2011) | 42 | 2011 | Proceedings of the VLDB Endowment |
| | Trojan Data Layouts | (Jindal et al., 2011) | 19 | 2011 | Proceedings of the 2nd ACM Symposium on Cloud Computing, SOCC 2011 |
| **Data processing** | MRShare | (Nykiel et al., 2010) | 78 | 2010 | Proceedings of the VLDB Endowment |
| | ComMapReduce | (L. Ding et al., 2013) | 3 | 2013 | Data and Knowledge Engineering |
| | Incoop | (Bhatotia et al., 2011) | 7 | 2011 | Proceedings of the 2nd ACM Symposium on Cloud Computing, SOCC 2011 |
| | REX | (Mihaylov et al., 2012) | 15 | 2012 | Proceedings of the VLDB Endowment |
| **Iteration** | HaLoop | (Bu et al., 2010) | 194 | 2010 | Proceedings of the VLDB Endowment |
| | Twister | (Ekanayake et al., 2010) | 185 | 2010 | HPDC 2010 - Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing |

| | | | | | |
|---|---|---|---|---|---|
| | Twister4Azure iterative MapReduce | (Gunarathne et al., 2013) | 1 | 2013 | Future Generation Computer Systems |
| | iMapReduce | (Zhang et al., 2012) | 29 | 2012 | Journal of Grid Computing |
| | HadUP | (Lee et al., 2014) | 6 | 2014 | Future Generation Computer Systems |
| **Data Transfer** | SHadoop | (Gu et al., 2014) | 5 | 2014 | Parallel and Distributed Computing |
| | Tiled-MapReduce | (R. Chen & Chen, 2013) | 2 | 2013 | Transactions on Architecture and Code Optimization |
| | MapReduce-based Big Data Processing on Multi-GPU systems | (Jiang et al., 2014) | 1 | 2015 | Cluster Computing |
| | Architecture of next generation apache Hadoop Mapreduce framework | (Murthy et al., 2011) | - | 2011 | Apache Hadoop |
| **Resource allocation** | Maestro | (Ibrahim et al., 2012) | 8 | 2012 | Proceedings - 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing |
| | LATE Scheduler | (Zaharia et al., 2008) | 1180 | 2008 | Usenix |
| | COSHH | (Rasooli & Down, 2014) | 4 | 2014 | Future Generation Computer Systems |
| | MCP | (Qi et al., 2014) | 5 | 2014 | IEEE Transactions on Computers |
| | Quincy | (Isard et al., 2009) | 96 | 2009 | Proceedings of the 22nd ACM SIGOPS Symposium on Operating Systems Principles |
| | MRA++ | (Anjos et al., 2015) | 1 | 2015 | Future Generation Computer Systems |
| | ARIA | (Verma et al., 2011) | 32 | 2011 | Proceedings of the 8th ACM International Conference on Autonomic Computing |
| | Flex | (Wolf et al., 2010) | 16 | 2010 | ACM/IFIP/USENIX 11th International Middleware Conference |
| | MapReduce indexing strategies | (McCreadie et al., 2012) | 10 | 2012 | Information Processing and Management |
| **Communications** | Muppet | (Lam et al., 2012) | 14 | 2012 | Proceedings of the VLDB Endowment |

## Declarative Interfaces

The demand of complex queries has significantly increased with the growing number of companies that realizes analysis and decision systems to support their business operations. Such complex queries sometimes require extra expressions in one or more query running in batch. Moreover, minimizing the number of queries and processing time for large amount of related data encountered by the current big data processing techniques is challenging. MapReduce is a great approach of handling a query problem. However, MapReduce has its own limitations in dealing with queries. Table 6 summarizes the declarative interfaces system for MapReduce framework.

Pig Latin (Olston et al., 2008) was designed for analyzing huge amounts of data sets in a time efficient manner. Pig Latin is a language that prefers map-reduce style programming over Structured Query Language (SQL) style programming to control the execution plan. This pseudo-language offers high-level data manipulation primitives such as projection and join, but in a significantly less declarative style than SQL. In (Olston et al., 2008), Pig compiles Pig Latin into physical plans that are executed

over Hadoop platform, which is an implementation of MapReduce programming model. Consequently, the results of Pig Latin exhibited greater performance in analyzing data compared with direct Hadoop. Similarly, (Zhou et al., 2012) presented a scripting language named Scope structured computations optimized for the parallel execution of analyzing large amount of data on clusters, supporting SQL by combining benefits from both traditional parallel database and MapReduce execution engine to enable easy programmability and to deliver scalability and high performance via advanced optimization.

Hive (Thusoo et al., 2009) was developed on top of Hadoop. In particular, this platform employs MapReduce programming framework for reducing the complexity of query processing for developers, offers the structure for warehouse in HDFS and other input sources (i.e., Amazon S3), and is a sub-platform in the eco-system of Hadoop. Hive has its own query language (HiveQL), which compiles to MapReduce and enables user defined functions (UDFs). Moreover, this platform is primarily based on three related data structures, namely, Tables, Partitions, and Buckets. Tables correspond to HDFS directories, can be distributed in various Partitions, and can then be divided in Buckets.

Tenzing (L. Lin et al., 2011) is a query engine, which is built on map-reduce to analyze Google data. Several enhancements have been realized to develop such an engine in Google MapReduce, thereby enabling scalability, high throughput, low latency, high performance, reliability, and metadata awareness. These enhancements also make SQL operators more efficient. Tenzing has been adopted by Google and some other communities (e.g., finance and marketing etc) to perform ad hoc analysis. Tenzing service was run over 1.5 petabytes data, and its results were exceedingly fruitful as discussed in (L. Lin et al., 2011).

Jaql (Beyer et al., 2011) is a scripting declarative language used to analyze large-scale semi-structure data with Hadoop's MapReduce programming model in a parallel manner. The design features of this functional data processing such as flexibility, reusability, and scalability enhance the capability of Hadoop in analyzing large datasets of thousands of users and log processing (Beyer et al., 2011). Jaql script can run on unstructured schema and often works on the top level of abstraction for logical operations. Moreover, Jaql enables users to determine the evaluation plan of script to add new operators. Such a query language is adopted by IMB because of its extraordinary results on Hadoop running Jaql as discussed in (Beyer et al., 2011). Jaql infrastructure is based on two layers, namely, the I/O and storage layers. When a script is written for a specific purpose, it passes through an interpreter, is compiled with a parser, and then evaluates the data from I/O layer. The remote nodes interpretation is performed using MapReduce. The Jaql compiler can translate Jaql script into a set of MapReduce jobs. The storage layer provides an API to access the data from different local and distributed systems, such as HDFS, DB2, Netezza, HBase, and web. I/O API (derive this flexibility from Hadoop) enables parallelism during evaluation and helps read and write common file formats.

In (Friedman et al., 2009), a new approach of implementing a UDF called SQL/MapReduce, which can overcome the abovementioned limitations of UDFs (e.g., poor capability), was proposed to parallelize executions and install time input and output specifications. This objective is derived from MapReduce programming model, which enables users to implement UDF in their own choice of programming language with API. This approach is implemented within the "Aster dataset systems' nCluster databases". The results revealed that this certain technique enables high-scalable computations in databases (Friedman et al., 2009).

DUALTable (Hu et al., 2014) is a hybrid storage model for Hive, particularly developed for storing selective data in HDFS or Hbase. In this storage model, new records are executed on HDFS. Moreover, this storage enables the adoption of the advantages of both HDFS and Hbase, improving computation performance without affecting sequential read. DUAL Table also enables the

multi-version feature of Hbase to maintain the record of changes and employs a cost model for deciding whether to place the modified information in the specified table or overwrite it within the entire table. The experiments of this model are performed on real data from "china state grid," and the findings specify that DUAL Table outclasses Hive by orders of scale in genuine settings.

**Table 6** Summary of the declarative interfaces system for MapReduce framework

| Interface system | Objective | Example | Pros | Cons | Ref |
|---|---|---|---|---|---|
| Sawzall | To provide high parallelism, while dealing with large clusters to analyze large amounts data | Google | • High-parallel computation<br>• Efficient ad hoc analysis | • Logical relational model<br>• Physical realization | (Pike et al., 2005) |
| SQL/MapReduce | To enable users in implementing UDF in their own choice of programming language with API | Aster | • Parallel computation of complex operations<br>• Simplification of queries | • Non-portable functions<br>• Low-level language is not declarative | (Friedman et al., 2009) |
| Tenzing | To perform ad hoc analysis on Google data | Google | • Tighter control related to program execution<br>• Efficient data access | • Lack of control structures, user interfaces, and performance bottlenecks(operators) | (L. Lin et al., 2011) |
| Hive | To reduce the complexity of query processing for developers | Facebook | • Simplification of query processing<br>• High-performance computation | • Is not supported for realizing updates and deletion<br>• Does not implement access control | (Thusoo et al., 2009) |
| Pig Latin | Designed to perform ad hoc analysis on large data sets | Yahoo | • Directly works at the file level<br>• Efficient analysis of large datasets | • Requires high-level, general data flow language<br>• Needs to code declarative query plans | (Olston et al., 2008) |
| Jaql | To analyze large-scale semi-structure data with Hadoop's MapReduce programming model in a parallel manner | IBM | • Flexibility<br>• Reusability | • Is complex to understand | (Beyer et al., 2011) |
| DualTable | To store selective data in HDFS or Hbase (hybrid storage model) | Chine State Grid | • Query optimization<br>• Execution environment optimization<br>• I/O optimization | • Requires extensive knowledge on lower level programming | (Hu et al., 2014) |
| HAWQ | To support massively parallel processing SQL engines sitting on top of HDFS for analyzing large amounts data | Pivotal lnc | • Fast query processing<br>• Read optimized storage | • Needs high availability of nodes | (Chang et al., 2014) |

## Data access

MapReduce is designed to scan the entire input file in a sequential-oriented manner (Dittrich et al., 2010; Sakr et al., 2013). This process, however, may increase the response time expectations when big data are processed. Consequently, such an instance induces two challenges during data processing. The first challenge pertains to the amount of big data stores that should be

analyzed as efficiently as possible. Nonetheless, these data do not match those of a well-configured parallel DBMS. The second challenge is the layout of data, which is inefficient because the data MapReduce should be executed at any given time in a distributed and parallel manner and may call for multiple heterogeneous datasets. To address two challenges several indexing and layout techniques are proposed.

Dittrich et al. (2010) proposed Hadoop++ system, which offers indexing techniques called Trojan indices for data stored in HDFS over UDFs without changing the original underlying Hadoop framework. In terms of these techniques, the only changes are implemented on the internal management of the data splitting process. Trojan indices are generated during the data load time and are embedded into logical input splits via bulk loaded read-optimized indices. In this event, the schema and anticipated MapReduce job can be identified for constructing appropriate indices (Sakr et al., 2013). The Trojan indices Hadoop++ also provides join technique of DBMS-independent and non-invasive for co-partitioning the input data during load time. For example, similarly partitioning function on the join attributes can be applied into two given input data during loading time. The co-group pairs, which contain similar join key from two relations, are placed on the same node. In this manner, the joins can be processed locally within each node at a query time. Experimental results have demonstrated that Hadoop++ outperforms HadoopDB.

Dittrich et al. (2012) introduced Hadoop Aggressive Indexing Library HAIL, which is an improvement of Hadoop MapReduce. In particular, HAIL provides support for uploading the pipeline of HDFS to create different clustered indices on each replica by keeping the existing physical replicas of an HDFS block in different sort orders. Moreover, because Hadoop MapReduce framework suffers from high scheduling overhead for short running jobs, HAIL enables the creation of more than three indices at a reasonable cost and significantly reduces this overhead with novel splitting policy to partition data during query time. This technique has significantly improved the performance of MapReduce jobs by including indexing in the MapReduce framework. However, such a scheme has failed to adopt changes in the workload of user as it forms indices upfront. Richter et al. (2012) explored this challenge by proposing LIAH, which is a parallel adaptive approach for indexing at minimal costs in MapReduce systems. LIAH aims to automatically and incrementally adapt to the workload of users by forming clustered indices on HDFS data blocks for executing MapReduce jobs. The system also parallelizes indexing with both map tasks computation and disk I/O.

Llama (Y. Lin et al., 2011) is a column-wise data management system on MapReduce. This system uses a column-wise partitioning scheme to transform the imported data into CFile, a special file format designed for such a system. Llama essentially aims to partition data into vertical groups, in which every group is stored in HDFS based on the selected column. Llama has been compared against Hiv, and the results of performance evaluation confirm the robustness, efficiency, and scalability of the former (Y. Lin et al., 2011).

Cheetah (S. Chen, 2010) is a distributed warehouse system developed on top of MapReduce and uses data storage in columnar format. This warehouse system provides features such as succinct query languages, seamless integration, and high performance. Moreover, Cheetah employs many techniques ranging from data compression and access method for multi-query optimization. The choice of compression

RCFile (He et al., 2011) introduces a record columnar file and is a placement structure on top of Hadoop, examining three commonly accepted data placement structures (i.e., row, column, and hybrid stores). This system also uses column-wise compression for providing efficient storage space utilization. RCFile is adopted by Hive and Pig, two most widely developed

systems in Yahoo and Facebook. Moreover, such a system outperforms other structures in most cases (He et al., 2011) and utilizes PAX layout similar to Cheetah; yet, it does not adopt block-level compression.

CIF (Floratou et al., 2011) is a column-oriented storage technique developed on top of Hadoop and uses binary storage formats to improve system performance over native use of text files. This format is compatible with the replication and scheduling constrains of Hadoop. Unlike other techniques such as RCFile and Cheetah, CIF offers a skip list column formats and lazy records construction strategy. Evidence indicates that using column-oriented storage can improve the performance of the map phase in Hadoop.

Trojan Data Layouts (Jindal et al., 2011) improve the challenges faced by using row-wise, column-wise, and PAX layout, which all incur poor system performance. The idea underlying this system is that it can internally organize data blocks into attribute groups according to the workload for improving data access times. Moreover, HDFS distributes the replicas of each data block across a cluster of machines. A different data layout is automatically formed per replica with Trajan HDFS. Comparative results suggest that Trojan Layout surpasses Hadoop with Row and PAX layouts (Jindal et al., 2011).Table 7 shows the comparison of different input data selection approaches.

**Table 7** Comparison of different input data selection approaches

| System | Data layout | Index creation | Compression | Partitioning | Comments |
|---|---|---|---|---|---|
| Hadoop++ (Dittrich et al., 2010) | Row-wise | Block-level | No | Horizontal | Based on using UDFs |
| HAIL (Dittrich et al., 2012) | PAX | Replica-level | Yes | Vertical | Uses record reader |
| LIAH (Richter et al., 2012) | Row-wise | Replica-level | No | Vertical | Index creation on map tasks |
| Llama (Y. Lin et al., 2011) | Column-wise | Block-level | Yes | Vertical | Runs on top of Hadoop |
| Cheetah (S. Chen, 2010) | Column-wise | Block-level | Yes | No | Runs on top of Hadoop |
| RCFile (He et al., 2011) | Row and Column-wise | N/A | Yes | No | Data placement structure |
| CIF (Floratou et al., 2011) | Column-wise | N/A | Yes | Horizontal | Leverages extensibility features |
| Trojan layouts (Jindal et al., 2011) | Trojan | Replica-level | No | Vertical | Uses Trojan HDFS |

## Data processing

The use of big data processing infrastructure in a cloud computing environment, in which different MapReduce jobs can produce similar results, may induce redundancy and may waste processing resources. However, sharing such result can decrease the overall amount of work, thereby reducing the cost while utilizing the resources of the processing infrastructure (Sakr et al., 2013). MRShare system (Nykiel et al., 2010) is a popular sharing framework in cloud computing that identifies different queries (jobs)

that can be processed in a batch mode by merging jobs into groups and treating each group as a single query. The optimization problem is examined to increase the overall saving of queries for preventing redundant executions. ComMapReduce (L. Ding et al., 2013) is a framework, which is an extension of the original MapReduce. ComMapReduce can effectively acquire certain shared information with efficient lightweight communication mechanisms. The idea behind this framework is that no communication may transpire among Mappers, neither among Reduces in parallel processing, particularly when the output results are remarkably smaller than the original input data. Thus, ComMapReduce can create an overlap and may waste time in processing the unpromising intermediate data. ComMapReduce is based on three basic communication approaches, namely, Eager, Lazy, and Hybrid for shared information. Compared with the original MapReduce framework in all metric without affecting the existing characteristics of the former, ComMapReduce is deemed better. Apart from that, joining multiple large data sets with complex conditions that require repetition of full computation of the entire datasets is feasible with frameworks such as MapReduce, which is obviously inefficient and wastes resources (Lee et al., 2014).

Incoop (Bhatotia et al., 2011) is a MapReduce framework used for incremental computations. Unlike ComMapReduce, Incoop detects changes in the input and automatically updates the output by employing an efficient, fine-grained result reuse mechanism. In particular, Incoop joins content-based chunking to the file system to detect incremental changes in the input file and to partition the data to maximize the reuse mechanism. The system has confirmed that incremental runs can be efficiently improved at a modest cost during the initial and first run, in which no computations can be reused.

Mihaylov et al. (2012) presented a programming model delta that supports iterative computations. Nevertheless, the main aim of this model is to provide an incremental refinement of the results. REX is formed around the incremental updates approach called programmable deltas, which is used for the state refinement of operators, instead of accumulating the state produced by previous iterations. Therefore, deltas can describe the changes between iterations for propagation and computation to increase the efficiency of iterative processing, while maintaining a state that has not changed. REX differs from MapReduce because it is based on different architectures and uses cost-based optimization to improve system performance. REX outperforms HaLoop in terms of its several implementations.

### Iteration

HaLoop (Bu et al., 2010) is a modified version of the Hadoop MapReduce framework that is particularly designed to serve iterative application with loop control. HaLoop enables both the input and output of stages to be cached for saving more I/Os during iterations. In this event, the system can be efficiently improved by adding various caching mechanisms and can make task scheduler loop-aware other than extending MapReduce with programming support for iterative applications. Ekanayake et al. (2010), Twister is a lightweight MapReduce runtime system that efficiently supports iterative MapReduce computations by adding an extra combine stage after the Reduce stage. Twister provides the features for cacheable MapReduce tasks, enabling developers to form iterative applications without spending considerable time on reading and writing large amount of data during each iteration. Moreover, (Srirama et al., 2012) clarified that Twister can handle iterative problems more efficiently than Hadoop MapReduce.

Gunarathne et al. (2013) introduced Twister4Azure iterative MapReduce runtime on Windows Azure Cloud. This iterative runtime enables users to efficiently run iterative processes on large datasets in parallel computations by abstracting the complexity of scalability and the fault tolerance from the users. Twiser4Azure offers many new features, including multi-level

data caching mechanisms to overcome the latencies of cloud services, programming model for iterative MapReduce computations, drawn framework managed fault tolerance to ensure the eventual completion of the computations, and the use of decentralized cache aware task scheduling to avoid single point of failures. Daytona is an iterative MapReduce runtime project for windows Azure that supports wide class of data analytics and machine learning algorithms. Table 8 shows the comparison of iterative MapReduce systems.

**Table 8** Comparison of iterative MapReduce systems

| System | Iteration Mechanism | Scheduling Tasks | Fault Tolerance | Benefits |
|---|---|---|---|---|
| HaLoop | Loop control | Uses loop-aware task scheduling | Checkpointing | Can be used in iterative big data analysis applications; |
| Twister | Additional combine phase after reduce phase | Uses static scheduling for MapReduce | Checkpointing | Iterative MapReduce computations. |
| Twister4Azure | K-Means clustering | Cache-aware decentralized task scheduling | Decentralized control architecture | Data-intensive computing for scientific discovery |
| iMapReduce | Specifies the iterative computation with the separated map and reduces functions | Uses static scheduling for MapReduce | Checkpointing | Reducing the overhead, eliminating the shuffling of static and enabling asynchronous execution of map tasks |
| HadUP | Detects and updates | Uses static scheduling for MapReduce | Checkpointing | Provides high performance, particularly in an environment, in which task level memorization has no benefit |

iMapReduce (Zhang et al., 2012) is an iterative processing framework for distributed clusters of machine, enabling users to select the iteration computation with a separate map and to reduce functions. Moreover, iMapReduce can automatically process tasks with multiple iterations within a single job.

During graph data processing, several iterations exist. Lee et al. (2014) proposed a modified Hadoop architecture named HadUP, which is suitable for large-scale incremental processing with conventional MapReduce algorithms. Such an approach can detect and compute the changes of datasets at a fine-grained level with a deduplication-based snapshot different algorithm and propagation of updates. Consequently, this revised architecture provides high performance, particularly in an environment in which the datasets are repeated many times.

## Data transfer

SHadoop (Gu et al., 2014) is a method used for improving the performance of Hadoop MapReduce by optimizing the job and task execution mechanism. SHadoop incorporates the following main optimization approaches: (1) Optimizing the job initialization and termination stages, thereby shortening the startup and cleanup time of all jobs; (2) Providing an instant messaging communication mechanism for efficient critical event notification that can benefit the majority of the short jobs with large deployment or many tasks. However, such optimization may induce a little more burden to the JobTracker because it needs to create and delete an empty temporary directory for each job. Compared with the standard Hadoop, SHadoop can averagely achieve 25% of performance improvement for various tested Hadoop benchmark jobs and Hive applications.

Multicore is potentially suitable for MapReduce framework to fully utilize the power of its processing resources such as abundant CPU cores (R. Chen & Chen, 2013). In the current Hadoop MapReduce framework, the distributed cache exists to reduce the

number of times the CPU would stall waiting for a memory request to be fulfilled, and as a second effect, may reduce the overall amount of data that should be transferred. Jiang et al. (2014) designed two multi-GPU MapReduce algorithms, namely, MGMR++ and PMGMR systems, to eliminate GPU memory limitations and to handle large amount of data via both the CPU memory and hard disks. MGMR is an extenuation of the MGMR with flexible templates using C++ and CPU memory utilization. PMGRM fine-tunes system performance through the latest GPU features such as streams and hper-Q. Studies have shown that MGMR++ and PMGMR have advantages over both CPU and single-GPU MapReduce in terms of performance and scalability aspects.

### Resource allocation

Efficient large-scale data processing is one of the major aspects of MapReduce framework characterization. However, the efficiency limitations of this design have been verified. The sharing of resources among MapReduce clusters is challenged, particularly if it requires resources allocation control and if it automatically tailors to different applications for achieving its performance goals (Murthy et al., 2011), (Ghit et al., 2014). For example, in distributed environment, job tracker manages and monitors nodes resources across the clusters and imposes the execution for all the queued and running MapReduce jobs. Hadoop MapReduce comes with scheduling model, in which the task trackers can provide a heartbeat status to the job tracker for compelling the implementation of tasks. Subsequently, the heartbeat is periodic. A pre-defined delay always exists when tasks are scheduled for any job.

(Ibrahim et al., 2012) also proposes scheduling algorithm named Maestro, which is design to improve the performance of the MapReduce computation. The study reveals current Hadoop scheduler performs inefficient scheduling of map tasks by degrading the replicas distribution. Maestro has two objectives, first, each data node is equipped the empty slots based on the replication scheme for their input data and the number of hosted map tasks and on a; second it consider the runtime of each scheduling tasks and the probability of scheduling a map task on a given machine depending on the replicas of the task's input data. The results of presented algorithm are very promising compared to current Hadoop.

In LATE Scheduler (Zaharia et al., 2008) the task which has longest estimate time to finish is always speculatively executed, when the result of the slow task progress rate is lower than the threshold default (25th percentile). Progress rate is calculated by dividing the progress score, which is the fraction of input data read by the running time, which is the amount of time the task has been running. In this case, if the execution of the task is less than the speculativeCUP in the entire cluster and the speed of the nodes selected to run speculation execution is faster whose total progress score is higher than the 25th percentile. The slow node is identified by considering the number of finished tasks that taking long time, in a situation where nodes in datacenters may not start at the same time. Moreover, the authors assume that the nodes run at consistent speeds is challenging considering the speed of some phases are fast and others slow.

COSHH (Rasooli & Down, 2014) propose a job scheduler to improve the performance of Hadoop job completion time in a heterogeneous cluster environment which consider both the application and cluster level of the system. The scheduler is competitive with respect to other performance measures such as fairness, locality and minimum share satisfaction. The authors also designed a scheduling algorithm with classifies the job based on their requirements and find an appropriate matching of resources and jobs in the system. The evaluation is performed by implementing the scheduler in MRSIM.

Qi et al. (2014) present a new set of speculative execution strategies called Maximum Cost Performance (MCP). Unlike Late and Mantri the proposed solution uses both the progress rate and the progress bandwidth within a phase to select slow task and exponentially weighted moving average (EWMA) to predict process speed and calculate a tasks remaining time. It also determine which task to backup based on the load of the cluster using a cost-benefit model, distinguish slow worker nodes by the process speed of map tasks completed on them and optimizing data locality of map task backups. MCP has been implemented in 27adoop-0.21.

In Quincy (Isard et al., 2009) addresses the problem of scheduling jobs on distributed computing clusters that is close to application data stored on the computing node. Each job in the node is managed by a root task that is assigned by scheduler in the cluster. Such node is responsible for submitting a list of worker to the scheduler in which these works have no dependency relationship. For each worker the rate is calculated based on the preference list of computers and racks that have a high rate of data on the computer in the rack of computers.

MRA++ (Anjos et al., 2015) has been presented as a scheduling and data placement framework design on MapReduce that is suitable for heterogeneous environment. The idea behind this framework is to efficiently perform intensive data computation in heterogeneous environment. The system consists of two main features, including task scheduling job control and the heterogeneity of nodes during data distribution. Moreover, the implementations proposed in MRA++ are concerned about whether the workload can be adapted to the computing capability of each machine.

(Ibrahim et al., 2012) also proposes scheduling algorithm named Maestro, which is design to improve the performance of the MapReduce computation. The study reveals current Hadoop scheduler performs inefficient scheduling of map tasks by degrading the replicas distribution. Maestro has two objectives, first, each data node is equipped the empty slots based on the replication scheme for their input data and the number of hosted map tasks and on a; second it consider the runtime of each scheduling tasks and the probability of scheduling a map task on a given machine depending on the replicas of the task's input data. The results of presented algorithm are very promising compared to current Hadoop.

ARIA (Verma et al., 2011) is a framework for addressing the problem of job scheduler in MapReduce environment. This framework consists of three components, which are as follows: (1) Job profile for a production job that is routinely executed on new datasets; (2) MapReduce performance model for a given job; (3) Soft deadline that estimates the amount of resources required for job completion within the deadline. Resource allocations can be increased by expediting job completion and can be realized during the Map and Reduce stages.

Flex (Wolf et al., 2010) is a flexible scheduling allocation schema, which aims to provide optimization for different kinds of standard scheduling theory metrics, stretch, and makespan, while ensuing the same minimum job slot guarantees. FLEX allocation is considered to synergistically work with an add-on module or standalone pluge-in replacement to the FAIR allocation scheme in HDFS.

MapReduce is used to allocate intensive data operations to distribute machines. The efficient retrieval of information from terabyte scale of data remains to be an issue. (McCreadie et al., 2012) discussed the critical analysis of four MapReduce indexing strategies (i.e., Per-token, Per-term (Ivory), Per-document, and Per-posting list indexing). These strategies have been implemented in the existing information retrieval framework for evaluation purposes. Experiments were performed with Hadoop

MapReduce, and the results indicated that such indexing strategies are suitable for retrieving information from terabyte data in MapReduce framework.

### Communication

Communication problems in MapReduce can be attributed to the partitions in the Reduce phase. While the data locality is observed in Map, in the Reduce phase the data are distributed in a uniform manner. Unbalanced workloads of reducers lead to considerable runtime differences, parallelism is poorly exploited, and the overall runtime increases. Lam et al. (2012) investigated the challenge of executing large amount of data over cluster of machines and developed a Muppet framework to achieve low latency and high scalability. In each machine, the Muppet can begin a set of called workers, which can be divided into mappers and updaters for executing a MapUpdate application on a cluster of machines.

Clustering algorithms are commonly used in data processing. Researchers have applied MapReduce to realize a high performance in processing big data. However, clustering big data with MapReduce incurs numerous problems, including high I/O and network cost. To this end, an implementation of K-means optimization with MapReduce was introduced in (Cui et al., 2014). The proposed novel model eliminates the iteration dependence on MapReduce, making processing performance efficient, scalable, and robust. Experimental results on big data sets demonstrate that the proposed algorithm enables efficient processing compared with parallel K-means, Kmeans||, and stand-alone Kmeans++ algorithms. The quality of this algorithm is measured with cluster validation. Moreover, in MapReduce, map computation is not long enough to achieve full overlap with the long shuffle in shuffle-heavy MapReductions. The "MapReduce with communication overlap (MaRCO)" was therefore presented in (Ahmad et al., 2013) to solve the above problem. MaRCO uses fervent sort and is reduced to overcome the problem.

## Research challenges

This section presents numerous significant problems that have not been fully addressed in the current studies. In this event, considerable research effort is still required to improve the efficiency of MapReduce framework. The relevant problems that are yet to be solved are discussed below.

- *Energy efficiency:* The continuous growth in the size of the data-centers containing Hadoop MapReduce clusters of hundreds and thousands of machines to support many users, has let in a tremendous increase in the energy consumed to operate these large-scale data centers. Consequently energy efficiency becoming a key open issue in the development of different techniques and approaches to optimize power management in Hadoop clusters (Ibrahim et al.). Furthermore, in interactive data analysis, MapReduce workload runs in large clusters, whose size and cost make energy efficiency a critical concern on MapReduce, particularly in cloud environment in which large equipped infrastructure is involved. For example, (Maheshwari et al., 2012) and (Wirtz & Ge, 2011) addressed the problem of energy conservation for large data centers that run MapReduce jobs. Accordingly, tremendously increased amount of energy is consumed to operate these data centers (i.e., electricity used for operating and cooling them) and ends up with a high money bill in the order of millions of dollars. To this end, the system may offload some computation tasks to the sources in distributed data centric environment to avoid the expensive data movement costs (Kambatla et al., 2014) and to achieve fast response time with minimal energy consumption.
- *Dynamic Resource allocation:* In distributed programming framework, the impact of a stragglers on the speed of a parallelized processing can be quite significant. Specifically, it decreases resource utilization and increases job

completion time. Unless the data is distributed in a fair manner, the runtime of the slowest straggler will easily dominate the total runtime. More importantly, in a heterogeneous environment equal runtime may not always be guaranteed even when the data is partitioned equally among the available machines. This is due to the fact that the complexity of some query processing requires advanced data partitions while others may simply contain data that need not be processed for the query at hand. Thus, allocation of resources adequately is very important in processing large amounts of data with MapReduce, especially when heavy computations with limited memory involved (Yang & Chen, 2015). Also, it requires efficient resource scheduling algorithm that mainly focused on allocate computing resources to the computational tasks. Another issues considerations are optimization of computation performance, communication latency and utilization cost in clouds (Yan et al., 2014) for future resource scheduling solutions. Moreover, Directed Acyclic Graph (DAG) model for scheduling tasks can be significantly important to avoid the scheduling overhead imposed by the MapReduce.

- *Processing big data in cloud computing:* MapReduce accelerates the processing of large amounts of data in a cloud; thus, MapReduce, is the preferred computation model of cloud providers (Zhifeng & Yang, 2013). Currently, many alternative solutions are available to deploy MapReduce in cloud environments; these solutions include using cloud MapReduce runtimes that maximize cloud infrastructure services, using MapReduce as a service, or setting up one's own MapReduce cluster in cloud instances (Gunarathne et al., 2010). However, several threats and issues, such as privacy, confidentiality, integrity, and availability of data, exist in processing big data using cloud computing platforms.
- *Load balancing:* Map and Reduce stages are linked with partitioning stage, which considered an important factor that effects the performance of the MapReduce. By default data is fairly partitioned using partition algorithm which can produce system load imbalance when skewed data is encountered that is because in partition the only key is considered not the size of the data. Normally in partition the keys are distributed equally, thus each reducer can process the same number of keys. However, the input of each reducer is not only key, but a data pair such as (key, value) and the number of values of each key is different, in this case some reducers may process more data thought keys are equally assigned to reducer.

- *Mapping scheme:* It assigns multiple inputs to a set of reducers in such a manner that for each output, a reducer receives all inputs while performing computation. Owing to the limited capacity of reducers, only specific inputs can be assigned. However, the size of each individual input may vary, which results in a high communication cost. Consideration and restriction of input size are important in the MapReduce framework and can help optimize the communication cost between map and reduce phases (Afrati et al., 2014). Several solutions have been proposed to minimize the number of copies of inputs being sent to reducers. Research is required to establish an efficient mapping scheme that can minimize the communication cost without affecting the performance of a specified task.
- *Performance optimizations:* Hadoop/MapReduce is useful because of its multiple characteristics, such as scalability, fault tolerance, and large amounts of data processing. MapReduce comprises several factors, namely, task initialization time, scheduling, and monitoring performance degradation in different types of applications (Kalavri & Vlassov, 2013). In addition, Hadoop/MapReduce does not support data pipelining or overlapping of the map and reduce phases. To improve the MapReduce performance, further optimization is required in the different aspects of MapReduce, such as index creation, reuse of previous computed results, and fast query execution. MapReduce will achieve better performance if the said issues are solved.

- *Optimized data shuffling:* In MapReduce, intensive disk input/output during the shuffling phase increases the overall execution time, which in turn degrades the performance of overall systems (Lin et al., 2013). Reducing the execution time has become challenging. Many solutions have been proposed to address this problem, but no solution has solved this problem completely in an efficient manner. Only new optimized techniques can solve this problem completely and efficiently. Research in this area can increase the performance of MapReduce by reducing the shuffle phase time.
- *Automation and configuration:* Automatic tuning and configuration help while deploying the Hadoop/MapReduce cluster by setting several parameters. To perform proper tuning, both hardware and workload characteristics must be known. While performing configuration, a small mistake can cause inefficient execution of jobs, which leads to performance degradation (Lama & Zhou, 2012). To overcome this issue, several new techniques and algorithms are required; these techniques and algorithms perform calculation, in which basis setting can be performed in an efficient manner. Creating such algorithms that receive input from the user, understanding the characteristics of underlying hardware by using machine learning, and suggesting a proper setting for better performance are challenging.

## Conclusion

MapReduce has proven to be a useful programming model framework for large scale data processing. This is because of its remarkable flexibility, which allows automatic parallelization and execution on a large-scale cluster with more than thousands of nodes. In this paper, a dataset consisting of 6,481 publications fulfilled the selection criteria specified in data collection, including journal types, conference proceedings, books, book sections, and patents. The collected data in Excel were used to organize the output by publication year. The top most cited research papers were subsequently extracted. Paper frequency, citation count, and h-index were also calculated to specify the countries, institutions, and authors that had the most contribution. Cited publications included all of the records indexed by Scopus from 2006 to 2015. Collaboration type was determined based on the address of each author. To analyze the study on MapReduce research from a bibliometric perspective, we initially conducted a basic bibliometric study using keywords, including titles of publication and abstracts, through which to subsequently create the special publication. The analysis of keywords (Y. Ding et al., 2001) uncover trends and patterns about particular domain by measuring the association strengths of terms that are representative of relevant publications produced in this research. The major feature of keyword analysis is its visualization of the intellectual structure of a specific discipline into maps of the conceptual space of this field; a time-series of such maps produces a trace of the changes in this conceptual space. The keywords are used by numerous researchers to identify word and expression frequency that indicates the core content of the literature.
Second, the generation of maps and co-word clusters is required through the representation of information and the intellectual structure of MapReduce research. As the last link in the chain, words are used to clarify the cognitive structure of a field through semantic maps. This process is often referred to as co-word analysis. For data visualization, VOSviewer program is used to produce distance- and graph-based maps (van Eck & Waltman, 2009). The functionality of VOSviewer is particularly useful for displaying large bibliometric maps in a manner that is easy to interpret.

The most influential articles contributed to the improvements in MapReduce framework for large datasets is also reviewed based on the most influential articles selected from papers cover the period 2006-2015 and described adoption and solutions that aim to alleviate some of the problems. For each part, we introduce the general background, discuss the technical challenges, and review the latest advances. We finally reviewed several open research challenges, including Energy efficiency, Resource allocation,

processing big data in cloud computing, real time processing, load balancing, mapping scheme, performance optimizations, optimized data shuffling, and automation and configuration.

## Acknowledgment

This paper is financially supported by the Malaysian Ministry of Education under the University of Malaya High Impact Research Grant UM.C/625/1/HIR/MoE/FCSIT/03

**Reference**

Afrati, F., Dolev, S., Korach, E., Sharma, S., & Ullman, J. D. (2014). ASSIGNMENT PROBLEMS OF DIFFERENT-SIZED INPUTS IN MAPREDUCE.

Ahmad, F., Lee, S., Thottethodi, M., & Vijaykumar, T. (2013). MapReduce with communication overlap (< i> MaRCO)</i>. *Journal of Parallel and Distributed Computing, 73*(5), 608-620.

Anjos, J. C., Carrera, I., Kolberg, W., Tibola, A. L., Arantes, L. B., & Geyer, C. R. (2015). MRA++: Scheduling and data placement on MapReduce for heterogeneous environments. *Future Generation Computer Systems, 42*, 22-35.

Beyer, K. S., Ercegovac, V., Gemulla, R., Balmin, A., Eltabakh, M., Kanne, C.-C., . . . Shekita, E. J. (2011). *Jaql: A scripting language for large scale semistructured data analysis.* Paper presented at the Proceedings of VLDB Conference.

Bhatotia, P., Wieder, A., Rodrigues, R., Acar, U. A., & Pasquin, R. (2011). *Incoop: MapReduce for incremental computations.* Paper presented at the Proceedings of the 2nd ACM Symposium on Cloud Computing.

Bollier, D., & Firestone, C., M. (2010). *The promise and peril of big data*: Aspen Institute, Communications and Society Program Washington, DC, USA.

Bu, Y., Howe, B., Balazinska, M., & Ernst, M. D. (2010). HaLoop: Efficient iterative data processing on large clusters. *Proceedings of the VLDB Endowment, 3*(1-2), 285-296.

Chang, L., Wang, Z., Ma, T., Jian, L., Ma, L., Goldshuv, A., . . . Sherry, G. (2014). *Hawq: a massively parallel processing sql engine in hadoop.* Paper presented at the Proceedings of the 2014 ACM SIGMOD international conference on Management of data.

Chen, M., Mao, S., & Liu, Y. (2014). Big Data: A Survey. *Mobile Networks and Applications, 19*(2), 171-209.

Chen, R., & Chen, H. (2013). Tiled-MapReduce: Efficient and flexible MapReduce processing on multicore with tiling. *ACM Transactions on Architecture and Code Optimization (TACO), 10*(1), 3.

Chen, S. (2010). Cheetah: a high performance, custom data warehouse on top of MapReduce. *Proceedings of the VLDB Endowment, 3*(1-2), 1459-1468.

Cui, X., Zhu, P., Yang, X., Li, K., & Ji, C. (2014). Optimized big data K-means clustering using MapReduce. *The Journal of Supercomputing, 70*(3), 1249-1259.

Dean, J., & Ghemawat, S. (2008). MapReduce: simplified data processing on large clusters. *Communications of the ACM, 51*(1), 107-113.

Dean, J., & Ghemawat, S. (2010). MapReduce: a flexible data processing tool. *Communications of the ACM, 53*(1), 72-77.

Ding, L., Wang, G., Xin, J., Wang, X., Huang, S., & Zhang, R. (2013). ComMapReduce: an improvement of mapreduce with lightweight communication mechanisms. *Data & Knowledge Engineering, 88*, 224-247.

Ding, Y., Chowdhury, G. G., & Foo, S. (2001). Bibliometric cartography of information retrieval research by using co-word analysis. *Information Processing & Management, 37*(6), 817-842. doi:http://dx.doi.org/10.1016/S0306-4573(00)00051-0

Dittrich, J., Quiané-Ruiz, J.-A., Jindal, A., Kargin, Y., Setty, V., & Schad, J. (2010). Hadoop++: Making a yellow elephant run like a cheetah (without it even noticing). *Proceedings of the VLDB Endowment, 3*(1-2), 515-529.

Dittrich, J., Quiané-Ruiz, J.-A., Richter, S., Schuh, S., Jindal, A., & Schad, J. (2012). Only aggressive elephants are fast elephants. *Proceedings of the VLDB Endowment, 5*(11), 1591-1602.

Ekanayake, J., Li, H., Zhang, B., Gunarathne, T., Bae, S.-H., Qiu, J., & Fox, G. (2010). *Twister: a runtime for iterative mapreduce.* Paper presented at the Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing.

Falagas, M. E., Pitsouni, E. I., Malietzis, G. A., & Pappas, G. (2008). Comparison of PubMed, Scopus, web of science, and Google scholar: strengths and weaknesses. *The FASEB journal, 22*(2), 338-342.

Floratou, A., Patel, J. M., Shekita, E. J., & Tata, S. (2011). Column-oriented storage techniques for MapReduce. *Proceedings of the VLDB Endowment, 4*(7), 419-429.

Friedman, E., Pawlowski, P., & Cieslewicz, J. (2009). SQL/MapReduce: A practical approach to self-describing, polymorphic, and parallelizable user-defined functions. *Proceedings of the VLDB Endowment, 2*(2), 1402-1413.

Fu, H.-Z., Wang, M.-H., & Ho, Y.-S. (2013). Mapping of drinking water research: A bibliometric analysis of research output during 1992–2011. *Science of the Total Environment, 443*, 757-765.

Ghemawat, S., Gobioff, H., & Leung, S.-T. (2003). *The Google file system.* Paper presented at the ACM SIGOPS Operating Systems Review.

Ghit, B., Yigitbasi, N., Iosup, A., & Epema, D. (2014). *Balanced Resource Allocations Across Multiple Dynamic MapReduce Clusters}.* Paper presented at the ACM SIGMETRICS}.

Greenspan, J., & Valkova, S. (2014). Using Big Healthcare Data for ILI Situational Awareness in Georgia. *Online Journal of Public Health Informatics, 6*(1).

Gu, R., Yang, X., Yan, J., Sun, Y., Wang, B., Yuan, C., & Huang, Y. (2014). SHadoop: Improving MapReduce performance by optimizing job execution mechanism in Hadoop clusters. *Journal of Parallel and Distributed Computing, 74*(3), 2166-2179.

Gunarathne, T., Wu, T.-L., Qiu, J., & Fox, G. (2010). *MapReduce in the Clouds for Science.* Paper presented at the Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on.

Gunarathne, T., Zhang, B., Wu, T.-L., & Qiu, J. (2013). Scalable parallel computing on clouds using Twister4Azure iterative MapReduce. *Future Generation Computer Systems, 29*(4), 1035-1048. Retrieved from http://www.sciencedirect.com/science/article/pii/S0167739X12001379

Hadoop, A. (2011). Apache Hadoop. Retrieved from https://hadoop.apache.org/

He, Y., Lee, R., Huai, Y., Shao, Z., Jain, N., Zhang, X., & Xu, Z. (2011). *RCFile: A fast and space-efficient data placement structure in MapReduce-based warehouse systems.* Paper presented at the Data Engineering (ICDE), 2011 IEEE 27th International Conference on.

Hsu, C.-H. (2014). Intelligent big data processing. *Future Generation Computer Systems, 36*(0), 16-18. doi:http://dx.doi.org/10.1016/j.future.2014.02.003

Hu, S., Liu, W., Rabl, T., Huang, S., Liang, Y., Xiao, Z., . . . Pei, X. (2014). DualTable: A Hybrid Storage Model for Update Optimization in Hive. *arXiv preprint arXiv:1404.6878*.

Ibrahim, S., Jin, H., Lu, L., He, B., Antoniu, G., & Wu, S. (2012). *Maestro: Replica-aware map scheduling for mapreduce.* Paper presented at the Cluster, Cloud and Grid Computing (CCGrid), 2012 12th IEEE/ACM International Symposium on.

Ibrahim, S., Phan, T.-D., Carpen-Amarie, A., Chihoub, H.-E., Moise, D., & Antoniu, G. Governing energy consumption in Hadoop through CPU frequency scaling: An analysis. *Future Generation Computer Systems*(0). doi:http://dx.doi.org/10.1016/j.future.2015.01.005

Isard, M., Prabhakaran, V., Currey, J., Wieder, U., Talwar, K., & Goldberg, A. (2009). *Quincy: fair scheduling for distributed computing clusters.* Paper presented at the Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles.

Jiang, H., Chen, Y., Qiao, Z., Weng, T.-H., & Li, K.-C. (2014). Scaling up MapReduce-based Big Data Processing on Multi-GPU systems. *Cluster Computing*, 1-15.

Jindal, A., Quiané-Ruiz, J.-A., & Dittrich, J. (2011). *Trojan data layouts: right shoes for a running elephant.* Paper presented at the Proceedings of the 2nd ACM Symposium on Cloud Computing.

Kalavri, V., & Vlassov, V. (2013). *Mapreduce: Limitations, optimizations and open issues.* Paper presented at the Trust, Security and Privacy in Computing and Communications (TrustCom), 2013 12th IEEE International Conference on.

Kambatla, K., Kollias, G., Kumar, V., & Grama, A. (2014). Trends in big data analytics. *Journal of Parallel and Distributed Computing, 74*(7), 2561-2573.

Kim, G.-H., Trimi, S., & Chung, J.-H. (2014). Big-data applications in the government sector. *Communications of the ACM, 57*(3), 78-85.

Labrinidis, A., & Jagadish, H. (2012). Challenges and opportunities with big data. *Proceedings of the VLDB Endowment, 5*(12), 2032-2033.

Lam, W., Liu, L., Prasad, S., Rajaraman, A., Vacheri, Z., & Doan, A. (2012). Muppet: MapReduce-style processing of fast data. *Proc. VLDB Endow., 5*(12), 1814-1825. doi:10.14778/2367502.2367520

Lama, P., & Zhou, X. (2012). *Aroma: Automated resource allocation and configuration of mapreduce environment in the cloud.* Paper presented at the Proceedings of the 9th international conference on Autonomic computing.

Lämmel, R. (2008). Google's MapReduce programming model — Revisited. *Science of Computer Programming, 70*(1), 1-30. doi:http://dx.doi.org/10.1016/j.scico.2007.07.001

Lee, D., Kim, J.-S., & Maeng, S. (2014). Large-scale incremental processing with MapReduce. *Future Generation Computer Systems, 36*(0), 66-79. doi:http://dx.doi.org/10.1016/j.future.2013.09.010

Lin, L., Lychagina, V., Liu, W., Kwon, Y., Mittal, S., & Wong, M. (2011). Tenzing a sql implementation on the mapreduce framework.

Lin, M., Zhang, L., Wierman, A., & Tan, J. (2013). Joint optimization of overlapping phases in MapReduce. *Performance Evaluation, 70*(10), 720-735.

Lin, Y., Agrawal, D., Chen, C., Ooi, B. C., & Wu, S. (2011). *Llama: leveraging columnar storage for scalable join processing in the MapReduce framework.* Paper presented at the Proceedings of the 2011 ACM SIGMOD International Conference on Management of data.

Lyon, D. (2014). Surveillance, snowden, and big data: capacities, consequences, critique. *Big Data & Society, 1*(2), 2053951714541861.

Maheshwari, N., Nanduri, R., & Varma, V. (2012). Dynamic energy efficient data placement and cluster reconfiguration algorithm for MapReduce framework. *Future Generation Computer Systems, 28*(1), 119-127.

Manyika, J., Chui, M., Brown, B., Bughin, J., Dobbs, R., Roxburgh, C., & Byers, A. H. (2011). Big data: The next frontier for innovation, competition, and productivity.

Mao, G., Zou, H., Chen, G., Du, H., & Zuo, J. (2015). Past, current and future of biomass energy research: A bibliometric analysis. *Renewable and Sustainable Energy Reviews, 52*, 1823-1833. doi:http://dx.doi.org/10.1016/j.rser.2015.07.141

McAfee, A., Brynjolfsson, E., Davenport, T. H., Patil, D., & Barton, D. (2012). Big Data. *The management revolution. Harvard Bus Rev, 90*(10), 61-67.

McCreadie, R., Macdonald, C., & Ounis, I. (2012). MapReduce indexing strategies: Studying scalability and efficiency. *Information Processing & Management, 48*(5), 873-888.

McKenna, A., Hanna, M., Banks, E., Sivachenko, A., Cibulskis, K., Kernytsky, A., . . . Daly, M. (2010). The Genome Analysis Toolkit: a MapReduce framework for analyzing next-generation DNA sequencing data. *Genome research, 20*(9), 1297-1303.

Meho, L. I., & Yang, K. (2007). Impact of data sources on citation counts and rankings of LIS faculty: Web of Science versus Scopus and Google Scholar. *Journal of the american society for information science and technology, 58*(13), 2105-2125.

Mihaylov, S. R., Ives, Z. G., & Guha, S. (2012). REX: recursive, delta-based data-centric computation. *Proceedings of the VLDB Endowment, 5*(11), 1280-1291.

Murthy, A. C., Douglas, C., Konar, M., O'Malley, O., Radia, S., Agarwal, S., & KV, V. (2011). *Architecture of next generation Apache Hadoop MapReduce framework*. Retrieved from

Murthy, A. C., Vavilapalli, V. K., Eadline, D., Niemiec, J., & Markham, J. (2013). *Apache Hadoop YARN: Moving Beyond MapReduce and Batch Processing with Apache Hadoop 2*: Pearson Education.

Nykiel, T., Potamias, M., Mishra, C., Kollios, G., & Koudas, N. (2010). MRShare: sharing across multiple queries in MapReduce. *Proceedings of the VLDB Endowment, 3*(1-2), 494-505.

Olston, C., Reed, B., Srivastava, U., Kumar, R., & Tomkins, A. (2008). *Pig latin: a not-so-foreign language for data processing.* Paper presented at the Proceedings of the 2008 ACM SIGMOD international conference on Management of data.

Philip Chen, C. L., & Zhang, C.-Y. (2014). Data-intensive applications, challenges, techniques and technologies: A survey on Big Data. *Information Sciences, 275*(0), 314-347. doi:http://dx.doi.org/10.1016/j.ins.2014.01.015

Pike, R., Dorward, S., Griesemer, R., & Quinlan, S. (2005). Interpreting the data: Parallel analysis with Sawzall. *Scientific Programming, 13*(4), 277-298.

Polato, I., Ré, R., Goldman, A., & Kon, F. (2014). A comprehensive view of Hadoop research—A systematic literature review. *Journal of Network and Computer Applications, 46*, 1-25. doi:http://dx.doi.org/10.1016/j.jnca.2014.07.022

Qi, C., Cheng, L., & Zhen, X. (2014). Improving MapReduce Performance Using Smart Speculative Execution Strategy. *Computers, IEEE Transactions on, 63*(4), 954-967. doi:10.1109/TC.2013.15

Rasooli, A., & Down, D. G. (2014). COSHH: A classification and optimization based scheduler for heterogeneous Hadoop systems. *Future Generation Computer Systems, 36*, 1-15.

Richter, S., Quiané-Ruiz, J.-A., Schuh, S., & Dittrich, J. (2012). Towards zero-overhead adaptive indexing in Hadoop. *arXiv preprint arXiv:1212.3480*.

Rothstein, M. A. (2015). Ethical Issues in Big Data Health Research. *Journal of Law, Medicine and Ethics, 43*(2).

Sakr, S., Liu, A., & Fayoumi, A. G. (2013). The family of MapReduce and large-scale data processing systems. *ACM Computing Surveys (CSUR), 46*(1), 11.

Shvachko, K., Kuang, H., Radia, S., & Chansler, R. (2010). *The hadoop distributed file system.* Paper presented at the Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on.

Srirama, S. N., Jakovits, P., & Vainikko, E. (2012). Adapting scientific computing problems to clouds using MapReduce. *Future Generation Computer Systems, 28*(1), 184-192.

Sun, J., Wang, M.-H., & Ho, Y.-S. (2012). A historical review and bibliometric analysis of research on estuary pollution. *Marine Pollution Bulletin, 64*(1), 13-21.

Talia, D. (2013). Clouds for Scalable Big Data Analytics. *Computer, 46*(5), 98-101. doi:10.1109/MC.2013.162

Thusoo, A., Sarma, J. S., Jain, N., Shao, Z., Chakka, P., Anthony, S., . . . Murthy, R. (2009). Hive: a warehousing solution over a map-reduce framework. *Proceedings of the VLDB Endowment, 2*(2), 1626-1629.

van Eck, N., & Waltman, L. (2009). Software survey: VOSviewer, a computer program for bibliometric mapping. *Scientometrics, 84*(2), 523-538.

Vavilapalli, V. K., Murthy, A. C., Douglas, C., Agarwal, S., Konar, M., Evans, R., . . . Seth, S. (2013). *Apache hadoop yarn: Yet another resource negotiator.* Paper presented at the Proceedings of the 4th annual Symposium on Cloud Computing.

Verma, A., Cherkasova, L., & Campbell, R. H. (2011). *ARIA: automatic resource inference and allocation for mapreduce environments.* Paper presented at the Proceedings of the 8th ACM international conference on Autonomic computing.

White, T. (2009). *Hadoop: The Definitive Guide: The Definitive Guide*: O'Reilly Media.

Wirtz, T., & Ge, R. (2011). *Improving mapreduce energy efficiency for computation intensive workloads.* Paper presented at the Green Computing Conference and Workshops (IGCC), 2011 International.

Wolf, J., Rajan, D., Hildrum, K., Khandekar, R., Kumar, V., Parekh, S., . . . Balmin, A. (2010). Flex: A slot allocation scheduling optimizer for mapreduce workloads *Middleware 2010* (pp. 1-20): Springer.

Yan, F., Cherkasova, L., Zhang, Z., & Smirni, E. (2014). *Heterogeneous Cores For MapReduce Processing: Opportunity or Challenge?* Paper presented at the Proc. of IEEE/IFIP NOMS.

Yang, S.-J., & Chen, Y.-R. (2015). Design adaptive task allocation scheduler to improve MapReduce performance in heterogeneous clouds. *Journal of Network and Computer Applications, 57*, 61-70. doi:http://dx.doi.org/10.1016/j.jnca.2015.07.012

Yazti, D. Z., & Krishnaswamy, S. (2014). *Mobile Big Data Analytics: Research, Practice, and Opportunities.* Paper presented at the Mobile Data Management (MDM), 2014 IEEE 15th International Conference on.

Zaharia, M., Konwinski, A., Joseph, A. D., Katz, R. H., & Stoica, I. (2008). *Improving MapReduce Performance in Heterogeneous Environments.* Paper presented at the OSDI.

Zhang, Y., Gao, Q., Gao, L., & Wang, C. (2012). imapreduce: A distributed computing framework for iterative computation. *Journal of Grid Computing, 10*(1), 47-68.

Zhifeng, X., & Yang, X. (2013). Security and Privacy in Cloud Computing. *Communications Surveys & Tutorials, IEEE, 15*(2), 843-859.

Zhou, J., Bruno, N., Wu, M.-C., Larson, P.-A., Chaiken, R., & Shakib, D. (2012). SCOPE: parallel databases meet MapReduce. *The VLDB Journal—The International Journal on Very Large Data Bases, 21*(5), 611-636.

Zhu, H. P., Xu, Y., Liu, Q., & Rao, Y. Q. (2014). Cloud Service Platform for Big Data of Manufacturing. *Applied Mechanics and Materials, 456*, 178-183.