Distrib Parallel Databases
DOI 10.1007/s10619-015-7173-2

Performance analysis of data intensive cloud systems based on data management and replication: a survey

Saif Ur Rehman Malik · Samee U. Khan · Sam J. Ewen · Nikos Tziritas · Joanna Kolodziej · Albert Y. Zomaya · Sajjad A. Madani · Nasro Min-Allah · Lizhe Wang · Cheng-Zhong Xu · Qutaibah Marwan Malluhi · Johnatan E. Pecero · Pavan Balaji · Abhinav Vishnu · Rajiv Ranjan · Sherali Zeadally · Hongxiang Li

Abstract As we delve deeper into the 'Digital Age', we witness an explosive growth in the volume, velocity, and variety of the data available on the Internet. For example, in 2012 about 2.5 quintillion bytes of data was created on a daily basis that originated from myriad of sources and applications including mobiledevices, sensors, individual

S.U.R. Malik (✉) · S. A. Madani
COMSATS Institute of Information Technology, Islamabad, Pakistan
e-mail: saif_ur_rehman@comsats.edu.pk

S. A. Madani
e-mail: madani@ciit.net.pk

S. U. Khan · S. J. Ewen
North dakota State University, Fargo, USA
e-mail: samee.khan@ndsu.edu

N. Tziritas
Shenzhen Institute of Advanced Technology, Shenzhen, Guangdong, China
e-mail: ntziri@gmail.com

J. Kolodziej
University of Bielsko-Biala, Bielsko-Biala, Poland
e-mail: jkolodziej@ath.bielsko.pl

A. Y. Zomaya
University of Sydney, Sydney, Australia
e-mail: albert.zomaya@sydney.edu.au

N. Min-Allah
University of Dammam, Dammam, Saudi Arabia

L. Wang
Chinese Academy of Sciences, Beijing, China
e-mail: lizhe.wang@gmail.com

Published online: 14 March 2015

archives, social networks, Internet of Things, enterprises, cameras, software logs, etc. Such 'Data Explosions' has led to one of the most challenging research issues of the current Information and Communication Technology era: how to optimally manage (e.g., store, replicated, filter, and the like) such large amount of data and identify new ways to analyze large amounts of data for unlocking information. It is clear that such large data streams cannot be managed by setting up on-premises enterprise database systems as it leads to a large up-front cost in buying and administering the hardware and software systems. Therefore, next generation data management systems must be deployed on cloud. The cloud computing paradigm provides scalable and elastic resources, such as data and services accessible over the Internet Every Cloud Service Provider must assure that data is efficiently processed and distributed in a way that does not compromise end-users' Quality of Service (QoS) in terms of data availability, data search delay, data analysis delay, and the like. In the aforementioned perspective, data replication is used in the cloud for improving the performance (e.g., read and write delay) of applications that access data. Through replication a data intensive application or system can achieve high availability, better fault tolerance, and data recovery. In this paper, we survey data management and replication approaches (from 2007 to 2011) that are developed by both industrial and research communities. The focus of the survey is to discuss and characterize the existing approaches of data replication and management that tackle the resource usage and QoS provisioning with different levels of efficiencies. Moreover, the breakdown of both influential expressions

C.-Z. Xu
Wayne State University, Detroit, MI, USA
email: czxu@wayne.edu

Q. M. Malluhi
Qatar University, Doha, Qatar
email: qmalluhi@qu.edu.qa

J. E. Pecero
University of Luxembourg, Walferdange, Luxembourg
email: johnatan.pecero@uni.lu

P. Balaji
Argonne National Laboratory, Lemont, IL, USA
email: balaji@mcs.anl.gov

A. Vishnu
Pacific Northwest National Laboratory,
Richland, USA
email: abhinav.vishnu@pnl.gov

R. Ranjan
CSIRO ICT Center, Marsfield, NSW, Australia
e-mail: rajiv.ranjan@csiro.au

S. Zeadally
University of the District of Columbia, Washington, DC 20008, USA
email: szeadlly@udc.edu

H. Li
University of Louisville, Louisville, Kentucky
email: h.li@louisville.edu

(data replication and management) to provide different QoS attributes is deliberated. Furthermore, the performance advantages and disadvantages of data replication and management approaches in the cloud computing environments are analyzed. Open issues and future challenges related to data consistency, scalability, load balancing, processing and placement are also reported.

Keywords Replication · Data management · Cloud computing systems · Performance gradation · Data intensive computing

1 Introduction and motivation

An overwhelming flow of data caused by the continuous increase of computational power has called for a paradigm shift in the computing architecture and large-scale data processing mechanisms [1]. Data intensive computations are difficult to achieve because of: (a) the increasing latency gap between multi-core Central Processing Units (CPUs) and Hard Disk Drives, and (b) the imbalance between data and computing capabilities of the current computing architecture [2]. Moreover, recent applications that manipulate huge bytes of distributed data must provide guaranteed Quality of Service (QoS) during network accessibility [3]. Providing the best effort services by ignoring the network mechanism is not enough to satisfy the customer requirements [4].

Recently, "Cloud Computing" has gained great hype, and according to Gartner cloud computing is a top ten technology of the year 2012 [5]. Cloud computing is a new paradigm related to the provision of computing infrastructure and is usually associated with large-scale data processing mechanisms. The cloud reduced the cost of managing software and hardware resources by pushing the infrastructure to the network, which allowed users to access services anywhere around the world [1]. Amazon, Google, IBM, Facebook, and Microsoft have started to establish data centers that host cloud computing applications in geographically distributed locations [6]. To deliver a desirable level of performance to the end users, the cloud must operate in a smooth and efficient way. Moreover, to ensure efficient data management (reading, writing, and storage), security, and availability, a data management system is required that efficiently manages the cloud computing-based software (load-balancer, SQL appliance, NoSQL appliance, monitoring appliance) and hardware services [virtual machines (VMs), storage, and network] [7]. Furthermore, on-premises enterprise data management systems have a large up-front cost of hardware and software that makes data management systems an ideal choice to deployed in the cloud [8].

Conventionally, relational databases and file systems were used to store data. However, with the growing demands of user, size of data, and need to get more information from the data, the utility of simpler storage systems that are easy to manage at a large scale increases. Data is an important entity in the cloud environment, and a proper, well maintained, and efficient system must be deployed to ensure reliability and availability of data. Some examples of cloud management platforms are Cassandra [9] and Hive [10] in Facebook, and HBase [11] in Streamy. A good data management system must have the ability to provide data security, availability, accessibility, and fault tolerance at all levels. Moreover, a system where data replication is used can provide better aforementioned abilities and response time, which are crucial from the perspective of the end users [12].

Data replication has been widely studied in the domain of distributed and cloud computing, such as in [13–27]. The data replication is a mechanism in which the data or some fragments of data are stored at multiple nodes or servers [28]. If one of the nodes is not accessible then the data can be accessed from a different node [29]. Data replication also involves staging, placement, and movement of data across a cloud. *Data staging* is a term used for storing data temporarily for processing at later stages of execution. In cloud computing systems, if a shared resource is not available then the data is "staged-in" at the site of execution. After the execution of "staged-in" data, the data is "staged-out" of the storage. *Data placement* is the process of placing data at different locations, and *data movement* is concerned with how: (a) data must be moved to preserve replication levels and (b) data will be accessed from different locations. Different techniques that divide the file into multiple blocks and distribute the blocks to data nodes for parallel data transfer were used previously for performance enhancement in the cloud, such as in [30,31]. However, in the abovementioned mechanism a node may be unreachable due to a failure, which is a norm rather than an exception. If one block is not available, then the whole file is not accessible. Efficient data sharing in systems is complex because of node failure, unreliable network connectivity, and limited bandwidth. The abovementioned issues must be given due consideration while implementing the cloud-based data management system architecture. To avoid performance degradation, data management and replication techniques must be designed with the goal of achieving QoS.

1.1 Performance measurement and analysis of the cloud services for data management applications

Considerable facts are available that suggest that revenue is directly influenced by the performance of services [32]. In an experiment by Google, a 20 % revenue loss was reported due to a delay of 500 ms in response time. Moreover, Amazon reported a sales decrease of 1 % due to an additional response time of 100 ms. The aforementioned examples indicate the importance and impact of the performance of the cloud services. Furthermore, the due consideration that needs to be given to, and the benefit of, performance to the cloud services are also obvious from the examples. The examples create a strong motivation to study and analyze the performance metrics of the cloud services (such as availability, throughput, and response time) for the data management and replication techniques. Therefore, we have analyzed data management and replication mechanisms based on the aforementioned performance metrics.

In this section we will discuss in detail the QoS metrics that are used to quantify the performance of the cloud services. Notably, the performance prediction in the case of shared resources, such as I/O, has been found to be complicated [33,34]. The situation becomes much complicated in context of cloud services due heterogeneity, workload interference, and variable Internet latencies. Moreover, the abovementioned

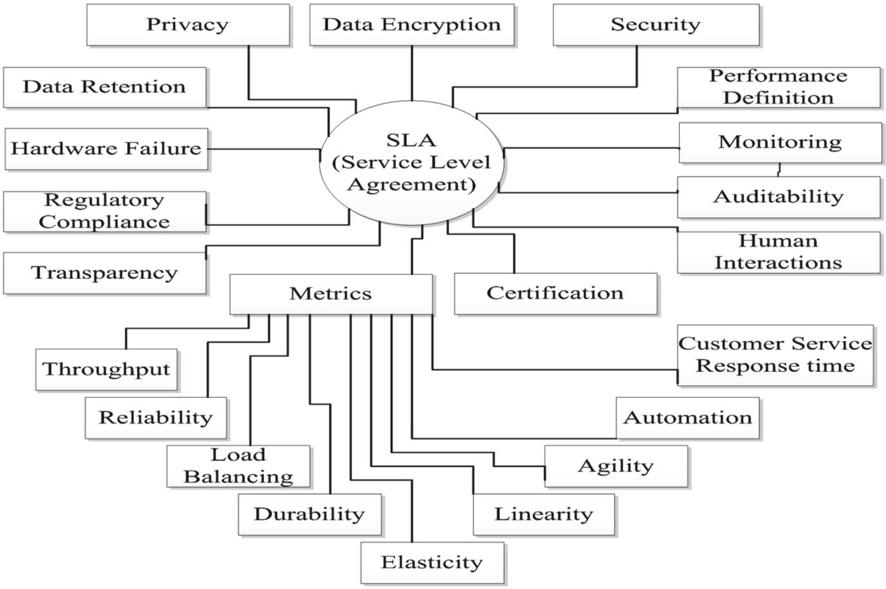

Fig. 1 Classification of SLA factors and performance metric distribution

phenomena can cause a certain amount of variation in the performance of clouds and are an active area of research [35].

The performance of a cloud services is negotiated between the CSP and the application provider/end-user via Service Level Agreement (SLA) based contract agreement that outlines the scale and scope of the QoS parameters [36]. A SLA can help improve the performance of clouds as it bounds a CSP to meet the agreed QoS in terms of availability, response time, and other parameters. Other QoS parameters that are considered in a SLA, includes privacy, data encryption, security, monitoring, auditability, human interaction, certification, metrics, transparency, regulatory compliance, hardware failure, and data retention/deletion, as depicted in Fig. 1 and elaborated below [37]:

- *Privacy* is the isolation of private data and applications in an environment that has multiple concurrent users of the system. A SLA should clearly address how privacy will be achieved.
- *Security* refers to the set of policies, technologies, and controls that must be implemented to protect data, the application, and associated infrastructure. Security is another factor of high concern for a cloud service. The cloud provider must understand the end user requirement and must enable appropriate control and federated patterns accordingly.
- *Data encryption* refers to the techniques that transform simple text into cypher text or any other form that is unreadable to unauthorized users. Encryption algorithms and relevant details, along with the policies to control the accesses to the data, must be pointed out clearly in the SLA (document).
- *Performance definition* is a term used to elaborate QoS parameters such as response time, throughput, and server availability in certain conditions. The aforementioned

measurements are usually not known to the service users, so clear description of these terms must be included in the SLA.

- *Monitoring* is a process of continuous observation of the performance of cloud services that notices and reports any potential breach from the CSP or service end users. To avoid conflicts between users and service providers, a third party that is usually neutral is inducted for monitoring. The aforementioned inducted party is responsible for reporting violations of the agreement, monitoring, and measuring the critical service parameters from both sides.
- *Auditability* is the ability to assess the systems and procedures of service providers. Users are liable for any breaches that occur with data loss or accessibility. The SLA must clearly state when and how an audit can occur.
- *Human interaction* refers to the responsiveness of the service provider whenever a request is dispatched. On-demand is a basic feature of the cloud, but in certain rare cases human interaction is required. Therefore, an SLA should adhere to this factor for some exceptional cases.
- *Transparency* refers to the extent to which nothing is hidden from the service users. If any critical data or application in the SLA is breached then service providers must be proactive in notifying the users
- *Metrics* refers to aspects that can be measured and monitored, such as throughput, reliability, and durability, as listed in Fig. 1. The metrics must be objectively and unambiguously defined in the SLA.

Performance of the cloud can be measured, monitored, and compared through the metrics shown in Fig. 1. The metrics are also known as the QoS attributes or parameters. Some other common metrics used for measuring the QoS of cloud services are: (a) *throughput*, (b) *reliability*, (c) *load balancing*, (d) *durability*, (e) *elasticity*, (f) *agility*, (g) *automation*, and (h) *customer service response time* [37].

The performance measurement criteria of cloud computing systems are huge, and to cover every aspect in one paper is infeasible. Therefore, in this survey the scope is narrowed to factors related to data-intensive cloud computing systems. The list of metrics used for the comparison and analysis of existing literature in this survey are: (a) *Availability* (the probability that a system is alive and functioning well in a stated environment, or the degree of liveliness of the system), (b) *Reliability* (the probability that a system will operate predictably under the stated environmental conditions over a specific period of time), (c) *Scalability* (the property of the system to grow, usually achieved by adding or upgrading hardware), (d) *Fault-Tolerance* (the ability of the system to respond gracefully to any unexpected failure of software or hardware), (e) *Load balancing* (the property of or methodology used by the system to divide the workload across servers, network links, CPUs, disks or other resources to achieve optimal utilization, throughput maximization, reduced response time and avoid overloading), (f) *Throughput* (amount of data or size of the message a system can process or transact per unit of time), and (g) *Consistency* (the probability that a system will not derive contradictory statements).

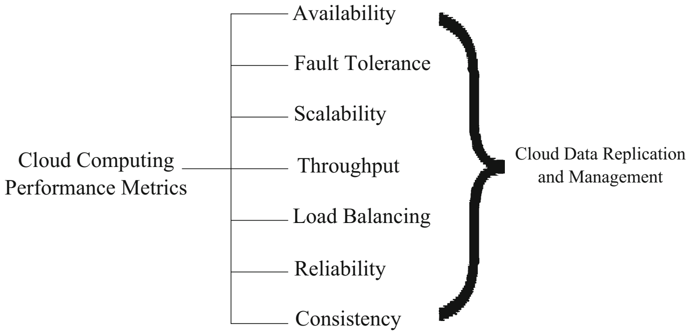

Fig. 2 A taxonomy of performance metrics used to study data replication and management strategies

1.2 Role of data management and replication in attaining high performance

Data management and data replication are the key elements instrumental for the success of the data-intensive applications in cloud computing environments [6,38]. Data management strategies provide scalability, adoptability, load and user balancing, multi-tenancy, and flexibility to the cloud services. Furthermore, availability, fault-tolerance, and failure recovery can be achieved through proper implementation of data-replication mechanisms over the cloud services [6]. These aspects require an adequate consideration towards the successful implementation of data intensive cloud computing systems. In the aforementioned respect, a fundamental understanding of the terminologies “Data Management” and “Data Replication” is compulsory to conceptualize the landscape of the novel field of cloud computing.

To comprehend the impact of data management and replication mechanisms on the cloud computing environment, the most well-known and widely used approaches are studied and compared at a single platform using some of the performance metrics discussed in Fig. 1, as shown in Fig. 2. Moreover, we have analyzed and compared the performance of data management and replication mechanisms based on the QoS metrics discussed in Sect. 1.1. Furthermore, the performance advantages and disadvantages of each approach along with the assumptions made for each mechanism are discussed. The study will help to define the strengths and imperfections of features with respect to the adaptation to the cloud computing services. Several studies related to data management and replications are available in the literature, such as [8,39–41]. However, to the best of our knowledge no survey is written that discusses and compares data management and replication approaches on the basis of performance metrics. Knowing the characteristics of the applications and how data is managed allows for optimistic utilization of network mechanisms in the cloud. Therefore, this survey can be useful for the networking community.

The highlighted contributions of the survey are:

(a) analyzing the performance advantages and disadvantages of the current state-of-the-art data management systems and data replication mechanisms in the cloud environments,
(b) comparing data management systems and replication mechanisms based on the SLA performance metrics,

(c) demonstrating the impact of different technical issues on the performance of systems and mechanisms, and
(d) identifying the open challenges and issues in the field of data replication and management in a cloud.

The remainder of the survey is organized as follows. Section 2 presents an overview of the service models, goals, and challenges involved in the cloud computing environment. Data replication, including an overview, goals, techniques, and the comparison of the techniques, is discussed in Sect. 3. Section 4 covers data management, including an overview, goals, platforms, and the comparison of data management techniques. The survey concludes with Sect. 5, which includes a discussion on the open research issues of data replication and management.

2 Cloud computing

The vision of cloud computing is to provide computing services in a manner similar to daily utilities, such as power and water. There has been a lot of discussion in industry and academia about the scope, definition, and future of cloud computing [42–44]. According to the National Institute of Standards and Technology (NIST) cloud computing is a model with which a shared pool of resources (networks, servers, storage, applications, and services) can be accessed conveniently and on-demand and that can be released with service provider interaction [1]. There are several characteristics of cloud computing, such as: (a) On-demand Self-service (users can independently and automatically utilize computing powers, such as server time and storage), (b) Broad-Network Access (user can access capabilities through heterogeneous platforms such as laptops and PDAs), and (c) Resource Pooling (resources, such as memory bandwidth and processing are available to any consumer).

Cloud computing is further classified into the categories of: (a) Software as a Service, in which applications or software are offered as services in real time to multiple organizations and end-users, (b) Platform as a Service, which provides a higher level environment where developers can build customized applications, and (c) Infrastructure as a Service (IaaS), which provides virtual resources, such as virtual machine servers (e.g. Amazon EC2), storage systems (e.g., Amazon S3), routers, bandwidth, network, switches, and other related tools that are necessary to build an application environment [8]. Every category has a different purpose that offers different facilities to businesses and individuals. These categories are also known as the models of cloud services [45]. Figure 3 depicts the categories along with examples of each service provider, and Fig. 4 shows the possible variations of cloud computing models.

The success of the cloud computing paradigm is a direct derivative of the advantages it provides to industries and organizations [46,47]. There are several issues and challenges that may hinder the successful implementation of data replication and management systems in cloud computing environments, as depicted in Table 1 [7,48–55].

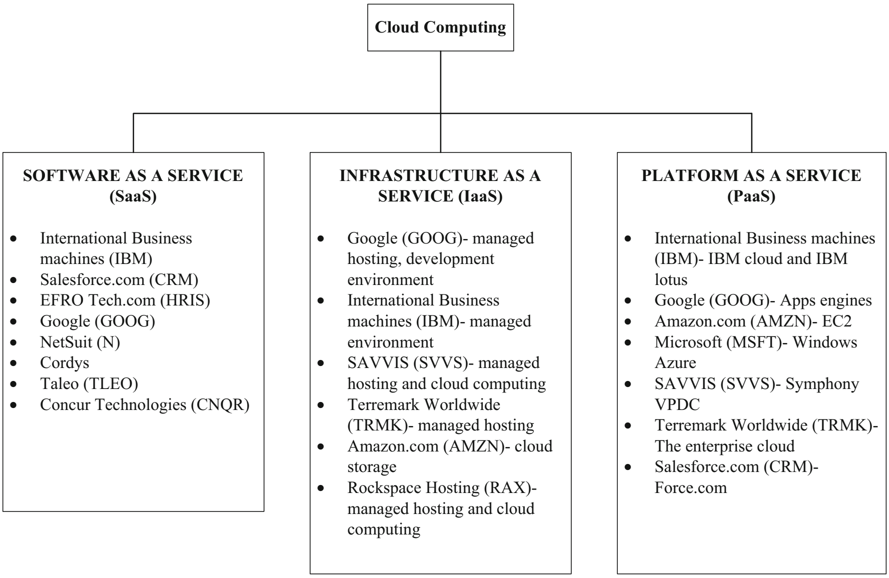
Cloud Computing
SOFTWARE AS A SERVICE (SaaS)
International Business machines (IBM)
Salesforce.com (CRM)
EFRO Tech.com (HRIS)
Google (GOOG)
NetSuit (N)
Cordys
Taleo (TLEO)
Concur Technologies (CNQR)
INFRASTRUCTURE AS A SERVICE (IaaS)
Google (GOOG)- managed hosting, development environment
International Business machines (IBM)- managed environment
SAVVIS (SVVS)- managed hosting and cloud computing
Terremark Worldwide (TRMK)- managed hosting
Amazon.com (AMZN)- cloud storage
Rockspace Hosting (RAX)- managed hosting and cloud computing
PLATFORM AS A SERVICE (PaaS)
International Business machines (IBM)- IBM cloud and IBM lotus
Google (GOOG)- Apps engines
Amazon.com (AMZN)- EC2
Microsoft (MSFT)- Windows Azure
SAVVIS (SVVS)- Symphony VPDC
Terremark Worldwide (TRMK)- The enterprise cloud
Salesforce.com (CRM)- Force.com

Fig. 3 Cloud computing service models and examples

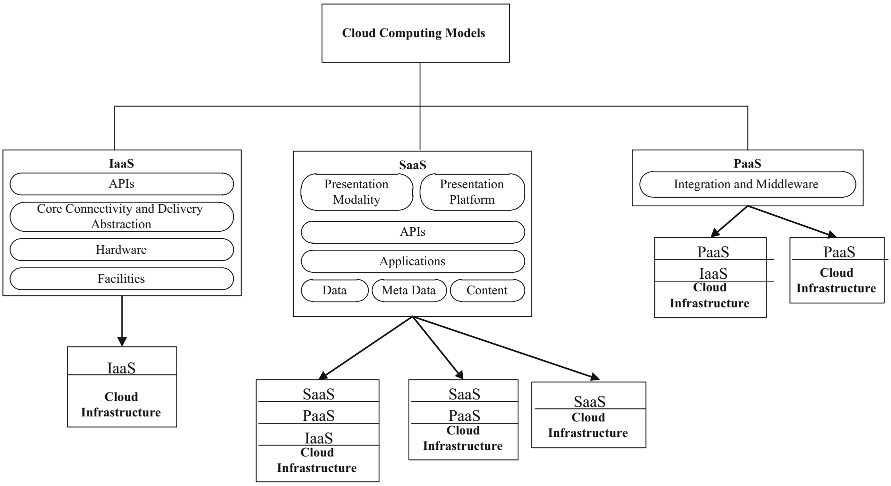

Fig. 4 Cloud computing service models variations

3 Data replication in cloud computing

3.1 Overview and previous research

Replication in cloud computing can be defined as placing multiple instances or copies of data across multiple data center locations (servers, nodes) [56]. A simple example of data replication is the practice of keeping multiple copies of an important document to recover in case of any failure or damage to the primary copy. Similarly, business organizations, governments, and other companies replicate the important financial,

Table 1 Cloud computing goals and challenges

Cloud computing	
Goals	Challenges
Reduced operational costs	Security and privacy issues
Adhering to service level agreement (SLA)	Lack of trust
	Heterogeneous cost model
Failure recovery or backup	Bandwidth and latency uncertainty
Increased processing power	Maintaining the SLA under
Collaborative tools utilization	Performance uncertainties
Achieving energy efficiency	Lack of interoperability
	Lack of quality assurance body

personal, and legal data to guarantee security, availability, and durability [29]. Examples of data storage services that have implemented replication techniques are Google file services (GFS) [57], and Amazon Simple Storage Service (Amazon S3) [58].

Data replication is used to increase the data availability of large-scale cloud storage systems. Data Objects and files are divided into several blocks and spread across the servers to enable parallel access that achieves high aggregate bandwidth. Replicating objects across the nodes has potential benefits [59]. However, maintaining replicas in excess can incur high overhead [60]. The number and placement of replicas is a key issue in data-replication management [61]. To understand how the number of replicas influences the availability and performance, an experiment is performed in [62]. The experiment is performed with the node failure ratios (failed nodes divided by total number of nodes) of 0.2 and 0.1. Experimental results illustrate that as the number of replicas increases, the availability increases until the maximum availability is reached, and the addition of replicas after the maximum point has no bearing on the system performance. Another observation from the experiment is that a lower node failure ratio reduces the number of replicas needed to reach maximum availability. Therefore, the goal must be to place the minimum number of replicas that satisfy the availability requirements. The purpose of almost all replication approaches or techniques is to improve the performance of the system (availability, reliability, fault tolerance) by keeping the cost (time) as low as possible. Replication is mandatory for systems where requirements are fault-tolerance and reliability, as established in the real world example of OceanStore [63].

The number of replicas needed and the placement of replicas is termed Replica Placement Problem (RPP), sometimes known as file allocation problem [64]. The number and location of replicas is an important aspect in replication management and performance enhancement. Studies [60,65–67] have proved that access latencies were reduced when the replica was placed in close proximity to the original on the servers. However, the trends change in RPP with advancements in the distributed systems [68]. In this survey we are concentrating only on techniques that perform or use file (unstructured data) replication.

Chu was one of the first researchers who worked on the file allocation problem (also termed as RPP) in a year 1969. A simple model that allows only nonredundant file allocation was proposed by Chu in [69,70]. The goal was to reduce the read and write cost with limited storage constraints at every site. In 1972, Casey [71] proposed a method of allocating multiple copies of files in an information network by introducing a distinction between queries and updates. The distinction was introduced because the update has to be performed in all copies and the query needs to access only one of the copies, which keeps the consistency in all replicas. In 1974, Eswaran performed a research and find out the file allocation problem is NP-complete problem [72]. In 1976, another approach was proposed by Mahmoud that suggested a change in the hardware as well as in the allocation of the files [73]. Moreover, Ref. [73] uses heuristic approach to develop a solution. In 1979, the file allocation problem was discussed in a distributed environment by Ramamoorthy [74]. Later in 1985, the file allocation problem was discussed under local area network environment with broadcasting abilities by Wah in [75]. This is the time when the development in file allocation problem starts progressing.

3.1.1 Distributed parallel file system (DPFS)

A DPFS stripes the data over different servers to achieve high performance in High Performance Computing (HPC) systems. Usually, DPFS uses Object Storage Device (OSD) for small chunks of data along with the centralized servers of metadata. Some of the well-known DPFS are briefly discussed below.

Lustre is a parallel-distributed file system that is used at many HPC centers around the world. Lustre was developed in 1999 at CMU and is now owned by Oracle. Lustre is hosted on a Linux operating system environment. Different kinds of nodes and roles are defined in Lustre by the separation of file data from the metadata using an object storage environment. Lustre usually consists of thousands of clients, Object Storage Servers, and a pair of failover metadata servers. More information about lustre can be found in [76].

Cloudstore (previously known as kosmosfs), is a distributed file system that is used to provide high performance with availability and reliability [77] and is built on the idea of GFS. The intention of cloudstore was to provide the backend storage infrastructure for data intensive applications such as data mining, cloud computing, grid computing, and search engines. The architecture of cloudstore consists of the: (a) metadata server, (b) Block server, and (c) Client library.

Parallel virtual file system (PVFS) is an open source file system that is widely used in real production clusters and is widely supported by an active developer base [78]. PVFS is a user level parallel file system that was designed for high-performance access to large data sets. Applications that use PVFS are Argonne National Lab, Sandia National Labs and Ohio Supercomputer Centers.

The following section briefly discusses various data replication and placement techniques that are implemented (between the periods of 2009–2011) for cloud computing environments in order to improve QoS parameters or metrics. Figure 5 provides taxonomy of strategies that are discussed in the paper.

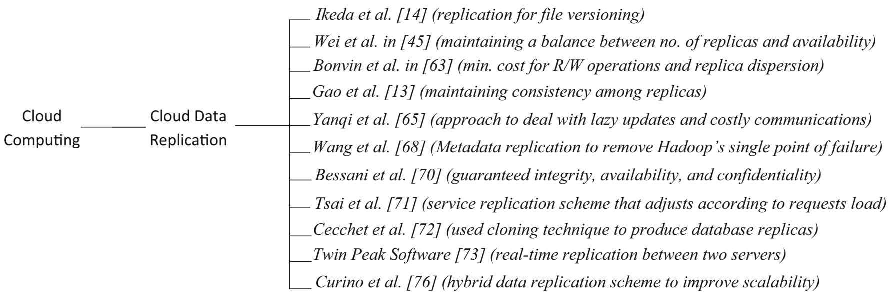
Cloud Computing
Cloud Data Replication
Ikeda et al. [14] (replication for file versioning)
Wei et al. in [45] (maintaining a balance between no. of replicas and availability)
Bonvin et al. in [63] (min. cost for R/W operations and replica dispersion)
Gao et al. [13] (maintaining consistency among replicas)
Yanqi et al. [65] (approach to deal with lazy updates and costly communications)
Wang et al. [68] (Metadata replication to remove Hadoop's single point of failure)
Bessani et al. [70] (guaranteed integrity, availability, and confidentiality)
Tsai et al. [71] (service replication scheme that adjusts according to requests load)
Cecchet et al. [72] (used cloning technique to produce database replicas)
Twin Peak Software [73] (real-time replication between two servers)
Curino et al. [76] (hybrid data replication scheme to improve scalability)

Fig. 5 A taxonomy and description of data replication strategies

3.2 Data replication approaches

Ikeda et al. [29] proposed a distributed data replication protocol for file versioning. File versioning is termed as the process of maintaining several versions of files. Reference [29] focused on the applications where the frequency of requests for newer version files is higher than for older version files. A distributed data replication protocol that supports file versioning is also proposed. Moreover, an analytical model is constructed for the optimal allocation of resources when the number of replicas and the read request frequencies of each version are provided. Three versions of the files are supported by the proposed protocol. In the protocol, only the client-sides are allowed to perform read and write operations, which improve availability by eliminating single point of failure. Replica nodes do not have any information about versions and only the clients are aware of the versions and cyclic alignment. The read and write requests are blocked during the write operation, causing large overhead in the cost (time) of the approach. Ikeda et al. [29] have proved the scalability and tolerance of the approach through simulations and experimentations. The strength of the proposed approach is the ability to hold several versions of the file to facilitate varied needs of numerous users. Moreover, as stated in Table 2, data recovery, scalability, and fault tolerance are some of the positive aspects of the approach. However, to manage and maintain multiple versions of the files require more resources and as the system expands the maintenance becomes complex and resource requirements increases exponentially. This may also lead to increase in cloud service leasing costs.

Wei et al. in [62] focused on the problems of: (a) placing replicas to effectively distribute the workload amongst data nodes and (b) computing the number of replicas that are needed for satisfying the availability requirement. In the abovementioned perspective, Ref. [62] proposed a Cost-Effective Dynamic Replication Management (CDRM) scheme that aims to provide cost-effective availability, improved performance, and load balancing of cloud storage. The strength of this approach is to build up a cost model that captures relationship between availability and replication that is used to determine if the availability requirements can be satisfied or not. Wei et al. in [62] considered the relationship between availability and the number of replicas for capturing the relationship. CDRM calculates and maintains a minimal number of replicas based on the availability requirements and can dynamically redistribute the workload to the data centers in a cloud computing environments. The aforementioned is done by

Table 2 Comparison of data replication techniques

Techniques	Authors	Advantage(s)	Disadvantage(s)	Assumptions	SLA Metrics						
					AT	FT	ST	TP	LB	RT	CT
Distributed data replication protocol for file versioning	Ikeda et al. [29]	Data recovery, no single point of failure, load sharing	Consumes more storage resources, no reconstruction of failed nodes	More access to newer versions, no dynamic addition of nodes	✓	✓	✓	☒	✓	✓	×
Cost-effective dynamic replication management scheme	Wei et al. [62]	Improved access latency, system stability	Failed nodes containing stripped pieces of files are not handled	All data nodes in the system are equally secure	✓	✓	✓	✓	✓	✓	☒
A self-organized, tault-tolerant and scalable replication Scheme for cloud storage	Bonvin et al. [80]	Cost efficient	Potentially large routing tables, elected server may become a bottleneck	Trustworthiness of the elected server	✓	✓	✓	×	✓	✓	×
Lazy update propagation for data replication	Gao et al. [28]	Improves throughput and reduces response time	Exhausted database connections due to a large number of threads	Communication network is FIFO	✓	×	✓	✓	☒	☒	✓
Cloud storage design based on hybrid of replication and data partitioning	Yanqi et al. [83]	Better user perceived update response and read access latency	Calculation of geographical distances and maintaining routing table entries are costly	Reliable channel, super node can never fail	✓	✓	✓	☒	☒	☒	✓
Hadoop high availability through metadata replication	Wang et al. [86]	Improved failover time	Services cannot be replaced	NA	✓	✓	✓	☒	×	✓	✓
DEPSKY: Dependable and secure storage in a cloud-of-clouds	Bessani et al. [89]	Improved access latency	No provision of IaaS model, Lack of data protection, cost of implementation is higher than other cloud implementations	Existence of a collision-resistant cryptographic hash function	✓	✓	☒	×	☒	☒	✓

Table 2 continued

Techniques	Authors	Advantage(s)	Disadvantage(s)	Assumptions	SLA Metrics						
					AT	FT	ST	TP	LB	RT	CT
Service replication with MapReduce in clouds	Tsai et al. [91]	Flexible, services can be replaced	Complicated to finish the MapReduce process, If CMS overloads then PSRS will degrade the performance of the cloud	NA	✓	✓	☒	✓	✓	☒	×
Dolly: Virtualization-driven database provisioning for the cloud	Cecchet et al. [92]	Reduced latency of starting new database replicas in virtualized public and private clouds	High storage requirements due to large number of replicas, Over time some paused VMs and VM snapshots become obsolete and are not cost effective to be resumed	Database server disk state (configuration file and data within the database) are stored on the elastic block storage volume. All components of the cloning operation has constant time	✓	✓	✓	☒	✓	☒	✓
Mirror file system for cloud computing	Twin peak software [94]	Better security, improved network speed and bandwidth, better control of hardware and software resources	NA	NA	✓	✓	✓	☒	✓	✓	☒
Schism: a workload driven approach to database replication and partitioning	Curino et al. [96]	Minimize number of distributed transactions, balanced partitions, high performance	Production of false positives (a partition will be involved in executing a statement, even though it does not need to be.)	Graph partitioning is more widely used and studied, and thus the tools are more mature	✓	☒	✓	✓	✓	☒	☒

adjusting the number and location of replicas according to the changing workload. The placement of replicas is based on node capacity (how many replicas can be stored) and blocking probability (number of replicas required to reduce access skews). Keeping replicas active improves the availability, load balancing, and the overall performance, provided that the replicas and requests are rationally distributed [12,79]. Reference [62] implemented CDRM in Hadoop Distributed File System (HDFS) and uses B+ tree to manage and update the system. The high blocking probability of a node causes the same node to be selected over and over again that causes imbalance in the system. Although, CDRM deal with the said imbalance by distributing the load, the imbalance still occurs, which the system has to deal repeatedly. This in turn also leads to the performance degradation. Moreover, Wei et al. in [62] demonstrate that CDRM satisfies availability requirements along with improved access latencies, load balancing, and system stability.

Bonvin et al. in [80] proposed a self-organized, fault-tolerant, and scalable scheme, termed Skute, for cloud data storage that dynamically allocates resources to several applications in a cost-efficient and fair way. Moreover, Skute dynamically adapts to load and disaster by determining the cost-efficient locations based on popularity. The rent (cost of using cloud services) and query response time are made efficient by using a self-managed key-value store. The scheme also offers and maintains multiple differentiated guarantees to all of the applications, regardless of the failures. Skute has a concept of Virtual Economy, in which every data partition (key range in a consistent hashing space) has the choice to make that depends on the benefit maximization. The utility offered, storage, and maintenance costs are usually considered for benefit maximization. To achieve the aforementioned maximization, the data partition can remove, replicate, or migrate itself. Furthermore, for several query rates and storage requirements an optimal resource allocation mechanism is developed that balances the query processing overhead and the availability objectives. Skute is designed for the purpose of: (a) minimum response time for read and write operations, (b) cost efficient replica dispersion, (c) distinguished availability guarantees for every data item, and (d) minimization of storage and bandwidth consumption [81]. Through experiments and simulations, the authors Bonvin et al. in [80] proved that the approach provides benefit maximization, fast convergence, and low communication overheads. The strong point of Skute is the provision of replication management scheme to evaluate replica prices and revenue across different geographic locations that relies on equilibrium analysis of data placement. However, the approach is not appropriate for high-quality content delivery, because of inaccurate transfer rate allocation [82]. Moreover, the entries of the routing tables are potentially very large that cause lookup time to increase.

Gao et al. [28] discussed the consistency between multiple data replicas. Reference [28] proposed the use of lazy update propagation to maintain consistency amongst the replicas in a cloud. In the lazy approach, the update is propagated to all other replicas after the update is committed to the original site. The lazy update operation reduces the transactional size, and the overall performance is improved due to fewer deadlocks. The authors in [28] have focused on a specific lazy replication scheme, termed "Lazy Master Replication" [59]. The approach can help increase the throughput of the information and data services, while decreasing the response time. Moreover, one of the strength of the approach is to allow consistent database maintenance, no transactions, and consistent snapshot of local data. However, the concurrent behavior of the requesting programs may cause exhausted database connections due to a large number of threads running at the same time. Furthermore, the primary copy may become a bottleneck and it must be placed online.

Yanqi et al. [83] proposed a Two-level Distributed Hash Table (TDHT) approach in distributed clouds when data partitioning is used to address the problems of: (a) applying lazy updates and (b) costly communications that occur during consistency verification. The proposed TDHT approach is the hybrid of data replication and partitioning. DHT approach is adopted by Yanqi et al. to eliminate the need of directory management, which is one of the major issues in data-partitioning-based systems [83]. The conventional centralized partitioning systems have the problem of scaling up as the system size grows. However, DHT can scale up easily because of the properties it holds, such as bounded hops and consistent hashing, as it was designed for peer-to-peer file sharing systems. A Two Level Access protocol is also designed by the authors to compare the performance of TDHT with the Distributed Version Server (DVS) [84]. The storage servers are divided into groups based on the locality. In every group, Short Secret Sharing, which is a well-known partitioning approach, is applied to each data object and the shares are distributed to servers in the group. The shares are then replicated to the other groups. Lazy approach is used to perform the updates that significantly reduce the user-perceived access latencies. The forte of the approach is to provide better user perceived update response latency and better read access latency then DVS. Moreover, consistent share verification is independently performed within a group, which reduces server-to-server communication cost. Furthermore, to achieve better performance, one-hop-routing DHT [85] scheme is used in both server and group levels. However, the approach has two important limitations. Firstly, it does not support range query because of the inherent design issue of DHT and secondly, lack of support for data migration leads to load-imbalance across the storage services. The management of shares and replicas becomes complex as the system is evolved.

According to Wang et al. [86], Hadoop provides high availability within a limited scope at a high cost of enhancement. Many mission critical applications that require high availability use Hadoop, which is designed to support data intensive distributed applications [11]. Hadoop uses the HDFS for storing data. By default, HDFS replicates the data. According to Wang et al. [86], metadata is stored on one node, allowing a system to be affected by a single point of failure. Reference [86] removed Hadoop's single point of failure to attain high availability through metadata replication approach. The solution is based on the phases of: (a) initialization, (b) replication, and (c) failover. In the initialization phase, the standby nodes are registered to a primary node and the initial metadata (version file and system image file) of the standby node is placed at the primary node. In the replication phase, the metadata, such as outstanding operations and lease states, is replicated. In failover phase, all communications are taken over by the newly elected primary node. The solution proposed by Wang et al. [86] identifies several critical nodes to provide high availability. Two nodes, (a) Namenode and (b) Jobtracker, are declared critical and are used by Wang et al. [86] to remove the single point of failure in Hadoop. The topological architectures of nodes are supported in the proposed execution environment are: (a) Active-Standby Topology, which consists of one active critical node and one standby node, and (b) Primary-Slaves Topology, which consists of one primary critical node and several slave nodes [86]. To reduce the overhead of replication, only metadata is replicated instead of a complete data copy. Moreover, experiments performed by authors illustrate the feasibility of, and effectiveness in, attaining high availability by measuring the failover time and runtime replication overheads. Furthermore, to maintain consistency among nodes, the authors use a three-phase commit protocol with non-blocking capability [87]. The highlighted aspect of the approach is to present an adaptive method that uses metadata replication of the NameNode for failover recovery to reduce the failover duration. However, the issue of single point of failure with Hadoop is still not solved by the approach [88]. Moreover, the approach is suitable for medium and not for higher number of I/O requests.

Bessani et al. [89] presented a system, DEPSKY, whose strength is to guarantee integrity, availability, and confidentiality of data stored on the cloud storage, by implementing encryption and replication on diverse commercial clouds. Byzantine Quorum System Protocols, Cryptographic Secret Sharing, and Erasure codes are combined to achieve the aforementioned objectives even in the presence of failures, data losses, and corruptions in some of the clouds. Reference [89] addresses the limitations of a cloud, such as loss and corruption of data, loss of privacy, loss of availability, and vendor lock-in. Four commercial clouds are used for deployment, and Planet-Lab (an open platform) is used to run clients and to access the service from heterogeneous geographical locations. DEPSKY needs a library to manage: (a) the heterogeneity of the interfaces and (b) the format of data accepted by each cloud. Two protocols are also proposed. The first DEPSKY protocol, which improves availability and integrity through replication using quorum technique, is termed Available DEPSKY (DEPSKY-A). To overcome the limitation in security of the DEPSKY-A, second protocol Confidentiality and Available DEPSKY (DEPSKY-CA) was proposed. DEPSKY supports multiple writer-locks and has read optimization protocols. Other protocols are provided by DEPSKY, such as garbage collection and cloud reconfiguration. Through extensive simulation and experiments, Bessani et al. [89], demonstrate that the proposed system can provide confidentiality and high availability. Since many distributed cloud services are combined together to achieve performance, it results in aggravated network upload and download cost. Moreover, DEPSKY does not provide IaaS model and does not protect data in IaaS clouds that provide VMs to the user [90].

Tsai et al. [91] proposed Service-Level MapReduce (SLMR), a new service replication scheme that allows a cloud to adjust its service instance deployments in response to existing and projected service request loads. In a cloud environment, services wait to serve the requests of users. If a service receives more requests than it can handle, additional resources need to be acquired. The SLMR is based on MapReduce, which is a parallel processing mechanism commonly used in cloud environments, such as Google App Engine (GAE). The SLMR includes dynamic service replication and pre-deployed service replication. Moreover, (a) a passive SLMR approach, which depends on the Cloud Management Service (CMS), and (b) an active SLMR approach, which does not need support from the CMS, are introduced in [91]. Tsai et al. [91] use MapReduce to split a task into smaller tasks, distribute the tasks to all nodes, and execute the tasks in parallel. The phases involved in MapReduce are: (a) map and (b) reduce. Reference [91] implemented map and reduce as library functions instead of services. Tsai et al. [91] developed two strategies for composing service-oriented MapReduce applications: (a) Passive Service Replication Strategy (PSRS) and (b) Active Service Replication Strategy. CMS manages the SLMR processes. Input data is split into partitions according to user requirement service capabilities and processor capacities. Map services are selected based on the request to make the service different from other map services. The CMS dynamically creates an appropriate number of map services, split services, shuffle services, cache services, reduce services, and output services, based on processing capacity and the timing constraints. The strength of the proposed technique is the ability to adjust its service instance deployment according to the load of running and anticipated service requests. However, the approach does not present any specific strategies for provisioning resources for computational tasks. Moreover, if the CMS overloads, then the PSRS will affect the performance of the cloud.

Cecchet et al. [92] address the challenges of providing shared-nothing replicated database in the cloud environment by using cloning techniques to produce database replicas. Reference [92] proposed Dolly, which is a system that dynamically provides database replicas in the cloud by efficiently using VM snapshots and cloning techniques. The VM of an existing replica is cloned, including the operational environment, database engine with all configurations, settings, and the database. In Dolly, the cloned VM is initiated on a new physical machine that results in a new replica that needs to be synchronized with other replicas before processing user requests. Dolly incorporates a network performance model for the estimation of latency based on the: (a) snapshot size and (b) database resynchronization latency that incurred for producing a replica (copying data across servers). The aforementioned network performance helps in guiding the replica scheduling process before the projected network traffic (workload) could increase. Moreover, an intelligent scheduling technique is also implemented in Dolly to determine whether to use a new VM snapshot or to use an older snapshot for scheduling replica creation activity. Furthermore, to optimize the resource usage administrators can tune the scheduling decisions, which are achieved by implementing a user-defined cost function to characterize database-provisioning policies in a cloud platform. The strength of Dolly is to create a new database instance in advance from a disk image snapshot and replays the transaction log to bring the new instance to the latest state. However, the instance created may take some warm-up time that can further degrade the performance. The authors Cecchet et al. [92] implemented the proposed system on commercial-grade open source databases and used public (Amazon EC2) and private clouds for demonstrating the optimized resource usage through the proposed cost estimation function.

Twin Peak Software [93] proposed a patented file system that is developed and implemented on Solaris and Linux, termed Mirror File System (MFS) [94]. MFS allows real-time replication between two servers. Moreover, the update can be replicated to an additional server or servers. When a MFS system receives an update from the application, all of the files linked by the MFS are updated in real-time [95]. All access to files and directories in both file systems are controlled by MFS. The replication process is transparent to the user. Moreover, the user application performs normal file and directory operations, such as read and write. Read operations only need to go to one file system for accessing the data. However, MFS also makes sure that updates are propagated to all of the file systems to make data consistent across all mirrored systems. Experimental evaluation has shown that MFS achieves better performance, security, and availability in the cloud [93]. The strength of the MFS is to replicate live files between geographically distant systems in real time. However, if one replica is updated, then all of the replicas have to be physically updated. Therefore, to reduce the cost of writing replica, it is imminent to develop an optimized and efficient replica placement technique.

Curino et al. [96] proposed Schism, a hybrid data replication and partitioning approach, to improve the scalability of distributed databases. The Schism models the system as a directed graph, where nodes and edges represent tuples and transactions, respectively. The inputs to the proposed system are a database, the representative workload, and the number of desired partitions. The basic steps involved in Schism are: (a) data pre-processing, which computes the read and write sets for each input transaction in the workload, (b) graph creation, representing database and workload in terms of a graph, (c) graph partitioning, which produces a balanced minimum-cut partitioning where each tuple is assigned to one partition and each partition is assigned to one physical node, (d) partition explanation, which analyzes the statements from the input workloads to extract the list of attributes and rules, and (e) final validation, which is the cost of per-tuple partitioning and range partitioning compared with hash and full table partitioning. The output of the Schism is a partitioning and a replication strategy that minimizes the overall cost of running the query against the stored database tuple. Distributed transactions are expensive in Online Transaction Protocol (OLTP), which is why graph partitioning algorithms are used in Schism to find the balanced partitions that can approximately minimize the number of multi-site transactions [96]. Moreover, a number of heuristics containing sampling and grouping of records are proposed to reduce the complexity of the graphs and to optimize the performance. Furthermore, a decision tree based approach to explain the partitioning in terms of a compact set of predicates is proposed. The authors Curino et al. [96] demonstrate through experiments that Schism is highly practical and provides better partitioning performance and ease of integration to existing databases. The strength of the approach is to find out the balanced partitions that can minimize the number of multi-sited transaction by using the graph partitioning algorithms. The focus is on fine grained partitioning that is beneficial for OLTP. However, the proposed approach is not appropriate for the complex and long queries that span around large data and tuples, such as in case of data intensive applications.

3.2.1 Discussion

The breadth of the research challenges as regards to data replication in cloud computing systems is very large, such as transactional replication, state machine replication, database replication, disk storage replication, and file-based replication. The approaches discussed in Sect. 3.2 are designed for a specific data management use-case or application. For example, the approach discussed in [29] is suitable for applications where user access patterns are imbalanced, such as most users access latest news, while others access older versions that may have been published few days ago. However, the cost of managing replicas in [29] is much higher as compared to other approaches, such as

[62] and [80]. Maintaining consistent replicas is one of the major issues in replication. The approach in [28] discusses the consistency of multiple replicas using lazy update that improves the throughput and response time. Reference [83] targeted the anomalies of using lazy updates and proposed a hybrid approach that provides better update response and read access latency. A method for failover recovery to reduce the failover time is proposed in [86]. To overcome the challenges in cloud services, such as loss and corruption of data, privacy, loss of availability, an approach is proposed in [89], which provide guaranteed data integrity, availability, and confidentiality. In general, most of the replication approaches are designed to provide availability of the data (as shown in Table 2). However, the application in which they can be implemented may vary. Accordingly, the performance may vary based on the application type.

The comparison of data replication techniques discussed in Sect. 3 is given in Table 2. A thorough analysis of each technique is presented with the summary on: (a) advantages, (b) disadvantages, (c) assumptions, and (d) the SLA metrics addressed in a particular technique. The legends for Table 2 are: (a) availability (AT), (b) fault tolerance (FT), (c) scalability (ST), (d) throughput (TP), (e) load balancing (LB), (f) reliability (RT), and (g) consistency (CT). NA denotes "Not Available". The symbol "☒" is used to represent that the particular metric is not discussed in the paper, and the"✓" and "×" symbols are used for "Provided" and "Not provided", respectively.

4 Data management in cloud

4.1 Overview

The goals of cloud computing based data management systems and approaches include availability, scalability, elasticity, performance, multi-tenancy, load and tenant balancing, fault tolerance, ability to run in a heterogeneous environment, and flexible query interface [46,47,97,98]. There are several challenges that hinder the successful deployment of data intensive applications on the cloud, including availability of a cloud service, data confidentiality, data lock-in, data transfer bottlenecks, application parallelization, shared-nothing architecture, performance unpredictability, and application debugging in large-scale distributed systems [6,97]. Researchers have proposed different strategies for enabling efficient data management systems using the cloud services. In this section we discuss different data management techniques proposed by researchers to achieve different performance goals and to deliver the required QoS to the end users.

The following section explains some of the data management approaches and system that are recently (2007–2011) been deployed in the cloud environment. Figure 6 shows taxonomy of the data management strategies studied in the paper.

4.2 Data management approaches and systems

Zhang et al. [99] proposed a multi-dimensional index approach that supports point and range queries for managing data stored over cloud services. Lack of efficient indexing techniques limits cloud based storage services, such as Amazon S3 and Azure Blob to

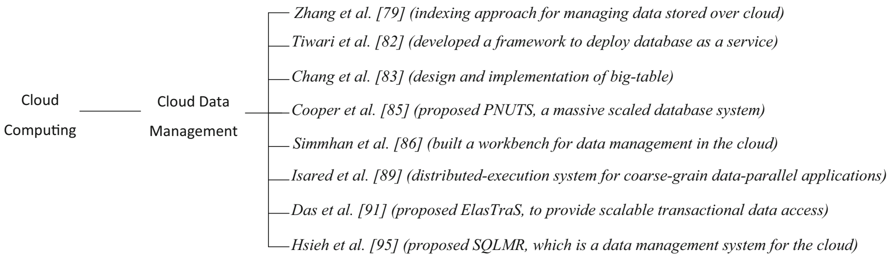
Cloud Computing
Cloud Data Management
Zhang et al. [79] (indexing approach for managing data stored over cloud)
Tiwari et al. [82] (developed a framework to deploy database as a service)
Chang et al. [83] (design and implementation of big-table)
Cooper et al. [85] (proposed PNUTS, a massive scaled database system)
Simmhan et al. [86] (built a workbench for data management in the cloud)
Isared et al. [89] (distributed-execution system for coarse-grain data-parallel applications)
Das et al. [91] (proposed ElasTraS, to provide scalable transactional data access)
Hsieh et al. [95] (proposed SQLMR, which is a data management system for the cloud)

Fig. 6 A taxonomy and description of data management strategies

support only simple keyword-based queries rather than efficiently resolving complex queries [11,57]. Data stored over the cloud storage service changes constantly and increases at an exponential rate, which makes indexing a complicated task. Reference [99] proposed a possible solution for supporting range queries by combining the R-Tree [100] and KD-Tree [101] to organize data records and perform fast query processing and efficient multi-dimensional indexing. Moreover, to accommodate the challenge of maintaining a consistent indexing structure on a large, transient data volume, Zhang et al. [99] proposed a cost-estimation-based strategy that can update the index structure effectively. A series of experiments were performed to illustrate that the scalability of the proposed indexing technique with the increase in the data volume and velocity. Moreover, the approach is generic and is not dependent on the underlying virtualization architecture, which makes the proposed approach suitable to implement on any cloud computing platform. The strength of the approach in [99] is the provision of fast query processing and efficient index maintenance by combining R-Tree and KD-Tree. Moreover, the aforementioned also supports: (a) multi-dimensional queries including range and point queries, and (b) data mining algorithms, such as K-Means and DBScan. However, because of the distinct master and slave nodes structures, the application of the approach is not possible to a typical DFS, such as GFS and HDFS [102].

Tiwari et al. [103] proposed a consistency model for a data store in the cloud and developed a framework towards the goal of deploying Database as a Service (DaaS). The DaaS over the cloud requires consistency across the data partitions and replicas [103]. The desirable aspect is the provision of cost-effective algorithm that ensures distributed consistency of data without compromising availability for fully replicated data. Moreover, the proposed approach deals with the elastic and scalable transaction management without any loss of functionality. The paper [103] targeted the data intensive applications that needed stronger consistency guarantees. The general architecture of the system consists of a node, which is a virtual-machine instance. Each virtualized node consists of data objects, an update queue, a queue manager, Interest Groups (IG), and Transaction Managers (TM). The IG is used for servicing client requests. The algorithm presented by Tiwari et al. is dynamic and takes into account the current performance of the network, while undertaking data replication or transfer operations [103]. As long as the IG is up to date and running, the copies across the nodes are consistent and synchronized. The latency of the access for the application depends on two factors: (a) the current size of the IG and (b) the number of working replicas (the size of the IG is dynamically varied based on the number of working replicas).

When a replica in the IG fails, the IG is automatically re-instantiated. Therefore, the performance of the approach does not suffer because of the IG replica failure. A large number of IG instances can lead to higher write latency for the application, as the TMs have to wait for large number of acknowledgements. On the other hand, a small number of IG would lead to larger update queues at each replica and will ultimately have more queuing delay. The approach may have low performance due to the fact that only a small number of nodes from the whole system are used for the replication. In the said perspective, latency may increase for some other nodes.

Chang et al. [104] discuss the design and implementation of big-table that are used by Google and other application-engine data stores. Structured data designed to scale to a huge set of commodity servers can be stored by the big-table. The aforementioned is a popular storage system used by most of the products and projects designed by Google, including Orkut, Google Docs, and Google Earth. A variety of demanding workloads that require throughput and data that is sensitive to latency, such as batch-processing jobs, can be achieved through the use of big-table. Fully relational data models are not supported by the big-table. Big-table is a distributed, sparse, multidimensionally-sorted map, which is indexed by a row key, column key, and a timestamp, where every value is an uninterrupted array of bytes [104]. The row keys are arbitrary strings of up to 64KB in size and data is maintained in lexicographic order by row key. Column keys are grouped into a set of column families, which are used to form the basic unit of access control in the big-table. The timestamp, which is a 64-bit integer, is used to index the versions of data stored in the big-table. Big-table uses Application Programing Interface (API) that provides the function of creating and deleting tables and column families. The API also performs the function of changing the cluster, table, and column family metadata. Some of the building blocks of big-table, such as Google File System, are rented from Google [57]. Moreover, cluster management system is used to schedule jobs, manage resources on shared machines, and machine failures. Chubby is a distributed locking service that forms the basis for the big-table [105]. Chubby performs several tasks in big-table, such as (a) ensuring that at least one master node is active all the time, (b) big-table data bootstrap location storage, (c) storage of schema information, and (d) listing access controls. The tasks that Chubby performs are critical to the functionality of big-table. If Chubby becomes unreachable for a certain period of time, then the whole big-table operation stops. The performance analysis provided by Chang et al. [104] demonstrated that big-table can help data intensive applications in attaining high performance and availability. Moreover, the size of the clusters can be easily scaled up by adding more machines as the resource demands change. The nucleus of the approach is the ability to scale to a very large size and to support varying demands of applications in terms of size of data and latency requirements. Big-table provides a simple data model that supports dynamic control over data layouts and formats. However, it weakens the 'A' (atomicity) guarantee from Atomicity, Consistency, Isolation, and Durability (ACID) by not offering a complete relational API. The only atomic sequence offered is read-modify-write sequence on data stored under a single row key [6]. Moreover, support for secondary indexes, multiple table creation, materialized views, and hash arranged tables, are not evidently discussed [106].

Cooper et al. [106] proposed Platform for Nimble Universal table Storage (PNUTS) to serve the data for the web applications rather than supporting complex queries. PNUTS is a massive scale hosted database system that provides (a) data storage organized as hashed or ordered tables, (b) low latency for large numbers of concurrent requests such as updates and queries, and (c) consistency guarantees. The PNUTS serves online workload, which contains queries that read or write single records. However, multiple records can be retrieved in parallel by using a "multiget" operation provided in PNUTS and specifying primary keys and predicate. The router component is responsible for routing a query (the storage unit required to be accessed for the read or write purpose). The primary key spaces of the tables are divided into intervals, where each interval leads to one tablet. The boundaries of each tablet map storage units to the tablet. Join operations are not supported by the PNUTS query model, as joins are considered too expensive in massive-scale systems [106]. The concept of log or archive is not available in PNUTS. However, if the update is not completed or lost before being applied to the disk, then PNUTS uses a redo log for replaying the update. APIs supported by the PNUTS are: (a) Read-Any (returns any possible version available from the record), (b) Read-Critical (returns versions that are nearly as new or as new as the required version), and (c) Read-Latest (returns the latest version of the record). Read-Critical and Read-Latest have higher latency and accuracy than Read-Any. To manage the operational load, Ref. [106] designed the system to cover and scale: (a) multiple replicas worldwide, (b) failover automation, and (c) load balancing. Moreover, the experimental results demonstrate that an efficient level of performance can be achieved under a variety of load conditions for individual record lookups and range scans. The strong point of PNUTS is the provision of efficient read access to geographically distributed clients, while providing serial writes. Moreover, application level guarantees are provided by serializing all requests to the same key. However, no consistency amongst the keys is provided in the PNUTS. Furthermore, the said approach weakens the consistency model by adopting a form of timeline consistency so that all replicas do not have to agree on the current value of the stored data.

Simmhan et al. [107] built a Trident scientific workflow workbench for data management in the cloud. The extensiveness of instruments and sensors that observe the physical and biological world brings huge quantities and a diverse range of rich data [108]. Therefore, data management of workflow in the cloud needs consideration. In this aspect, Ref. [107] investigated two classes of workflows: (a) Science Application workflows and (b) Data Preparation workflows. The workflows were used to drive common and discrete requirements from workflow systems in the cloud. Simmhan et al. [107] further use workflow examples from two collaborations, the North-East Pacific Undersea Networked Experiments (NEPTUNE) oceanography project and the Pan-STARRS astronomy project [109], to draw the comparisons. Furthermore, a number of collaborative roles for large scale data intensive computing, such as Pan-STARRS and NEPTUNE, are also defined by Simmhan et al. [107]. The roles are: (a) Producer (generates Level-0 data), (b) Curator (maintains the integrity of the shared data contributed by the producers), (c) Data Valet (produces the shared data products), (d) Publisher (builds and operates the entry point to search, curate, and use the data products), and (e) Consumer (the scientist or researcher performing an investigation).

The data valets and curators bear a greater responsibility, as the actions can potentially impact all consumers in ways that are often unknown to the consumer. Analysis of the workflow classes can guide the evolution of managing data flow systems in a cloud environment. Simmhan et al. [107] identified two important classes of workflow users, (a) Data Valets and (b) Domain Researchers. Trident focused on extending the functionality of the commercial workflow management system, which resulted in a smaller code to maintain, improved performance, and a complete requirements understanding unique to a scientific workflow running on the cloud. The strength of Trident is to build workflows with control and data flows from built-in and user-defined activities. The activities include control flow operations, invocation of web services, and SQL querying. However, open provenance model is not supported by the native Trident model and integrating both of the models is a complex task.

Isared et al. [110] discuss Dryad, a general-purpose distributed-execution system introduced by Microsoft for coarse-grain data-parallel applications. Dryad was designed to achieve scalability from small clusters of computers to data centers with thousands of computers. The strength of the Dryad execution engine is to handle several issues and complexities, such as the creation of large (concurrent and distributed) applications, scheduling, recovery, and data transportation. Moreover, Dryad provides flexibility by providing fine control to the developers on the communication graph and routines running inside the Dryad. Communication flow determines the overall structure of the job in Dryad. Communication flow consists of vertices and edges, where vertices represent the program and edges represent the data channels. During runtime the computational graph is automatically mapped to the physical resource. Structured items are transported on each channel in a finite sequence. A job manager in Dryad, responsible for coordination of the job, can be run at the user workstation or within the cluster. Features of the job manager include: (a) construction of the job communication graph through the application specific code and (b) the work scheduling across the resources that are available. The data can be sent directly in the graph between the vertices. Dryad uses its own high-level language that is the generalization of two other execution environments such as SQL and MapReduce. The aforementioned language is termed DryadLINQ [111]. The generalization between the environments is achieved: (a) through an expressive data model adopted by strongly typed .NET objects and (b) by providing support for imperative and declarative operations within the high level programming language. DryadLINQ provides a powerful hybrid of declarative and imperative programming. The aforementioned is achieved by exploiting the LINQ (Language Integrated Query). The purpose of the system design is to provide a flexible and efficient computation in any LINQ enabled languages such as C# and Visual Basic. Data parallel portions of the program are automatically translated in a distributed plan, which is then sent to the Dryad execution platform for execution. Reference [110] demonstrated an excellent behavior in scaling and performance of Dryad on small-scale and large-scale clusters. Dryad puts disk materialization steps between data flow stages that causes the Dryad iterative jobs to reload the data from disks, incurring a significant performance penalty. Moreover, a complex programming interface is required to support encapsulating multiple asynchronous stages into a single process. Furthermore, the aforementioned is only targeted to batch processing and not for continuous queries.

Das et al. [112] proposed Elastic Transaction data Store (ElasTraS), which addresses the issue of scalability and elasticity to provide scalable transactional data access. ElasTraS provides elasticity similar to that of elastic cloud, with the additional provision of transactional guarantees. Reference [112] combined previously proved elastic databases and scalable systems, such as [104,113,114] to deal with the concurrency control, isolation, recovery, and the limitations of distributed database systems. ElasTraS consists of: (a) transactional managers, responsible for transactional guarantees and providing elastic scalability with increase in demands, (b) application and web servers, which interact with the database, (c) load balancer to handle requests and implement the load balancing policy, (d) a higher level transactional manager, which receives requests from the load balancer and determines whether to execute the request locally or to forward it to the owning transaction manager, (e) an owning transaction manager, which has the rights to access the data by the transaction, (f) the distributed storage layer, which stores actual data for data store, and (g) a metadata manager, which manages information, such as system state and metadata, for the tables. The strength of the proposed approach is the provision of transactional guarantees along with the concurrency control, recovery, and isolation. ElasTraS supports static and dynamic partitioning. However, in dynamic partitioning transactional guarantees are limited to single partition because the applications are not aware of the other partitions. Moreover, ElasTras offers limited transactional semantics (mini-transactions) when dynamic partitioning is performed on the dataset and has no support for structuring computations [115].

Hsieh et al. [116] proposed SQLMR, which is a data management system for the cloud. SQLMR combines SQL and MapReduce. The desirable aspect of the approach is the combination of the aforementioned technologies that inherits the programming advantages of SQL along with the fault tolerant, heterogeneous, and scalable functionalities of MapReduce. Moreover, Ref. [116] provides a compiler to translate SQL programs to MapReduce. Furthermore, a technique to dynamically convert SQL files to HDFS that is to be accepted as an input to MapReduce runtime engine is proposed. The input to SQLMR is SQL queries that are translated into the MapReduce jobs. The architecture of SLMR consists of: (a) a SQL-MapReduce compiler, which converts SQL statements to sequential MapReduce jobs, (b) a query result manager, which searches the log to find if similar query results are available in the cache, (c) a database partitioning and indexing manager, which is responsible for managing data files, partitioning the new data, and creating indexes, and (d) an optimized Hadoop, which is responsible for the generation of optimized MapReduce jobs. Hsieh et al. [116] conducted several experiments to illustrate the scalability of data and system with respect to increasing data sizes. The approach suffers from the network load unbalancing because of the random placement of reducers, causing the reducers to become stragglers on a busy rack.

4.3 Discussion

It is noteworthy to consider that every data management system has been implemented to achieve different degree of performance metrics, such as consistency, scalability, and fault tolerance. Therefore, it is not straight forward to compare the aforementioned systems. However, we attempt to compare the systems under some common features. The big-table is built on column families having a low level query interface. Moreover, eventual consistency model is used to ensure consistency and performance requirements. Compared to big-table, PNUTS uses key-value store data model with timeline consistency model that weakens the consistency of the system, while focusing on the performance and availability of the data. The Dryad, on the other hand, uses relational store data model and follow strict consistent consistency model to ensure the consistency and availability requirements. Similarly, the approach presented in [103], ensures distributed data consistency without compromising availability but is not as scalable as big-table and PNUTS. The query processing mechanism and index maintenance of the approach proposed in [99] is fast and efficient as compared to the other approaches discussed in Sect. 4.2. One system cannot be best for all the workloads and different systems make different tradeoffs to provide optimized results. In general, maintaining ACID guarantees in the presence of replication is very hard. Moreover, according to CAP theorem [117], two out of three (consistency, availability, and partitions) properties can only be provided by any system.

The comparison of data management platforms that are discussed in Sect. 4 is given in Table 3. Analysis of each platform is presented with the: (a) advantages, (b) disadvantages, (c) assumptions, and (d) SLA metrics addressed in a particular technique. The legend for Table 3 is the same as that of Table 2.

5 Conclusions, open research issues, and future directions

In the last few decades, data has been produced in massive quantities by the Internet connected devices and applications. Continuous increases in the computational power and the advancements in recent technologies are the main reasons behind the data growth [1]. In this paper we have studied Data Replication and Management, two instrumental technologies that are widely used to manage massive quantities of data on cloud services. These techniques are required to assure strict QoS on data operations (search, upload, download, replicate, and the like). A comprehensive survey of techniques along with the (a) advantages, (b) disadvantages, (c) assumptions, and (d) SLA-based performance metric topographies are explored in this paper. The techniques are compared and analyzed based upon the abovementioned features. We also analyze the working of numerous data replication techniques and how data-intensive applications are deployed in the cloud. The knowledge provided in the paper can be further exploited to design and model new mechanisms or approaches in the cloud. Furthermore, the analysis of each approach, issue, and suitability to support and operate in certain environments led us to the identification of the following open research issues.

- *Managing data consistency in data replication:* Maintaining the consistency of the data across a number of mirrored nodes is a very complicated task in distributed data replication systems as compared to centralized systems. Moreover, maintaining consistency while balancing the cost of read and write operations is another major concern. Several protocols have been proposed in [29,118–122] to address this issue. There are different techniques of read and write operations that are used

Table 3 Comparison of cloud data management platforms

Technique	Authors	Advantage(s)	Disadvantage(s)	Assumptions	SLA Metrics						
					AT	FT	ST	TP	LB	RT	CT
Trident scientific workflow workbench for data management in the cloud	Simmhan et al. [107]	Smaller code base, Unique to scientific workflow	Workflow timer cannot resume. Custom activity within a workflow cannot be added twice	NA	✓	✓	✓	×	×	☒	✓
Efficient multi-dimensional index for cloud data management	Zhang et al. [99]	Efficient index maintenance, Platform independent improved query efficiency	Update schedules are unknown	Number of queries forwarded to slave node is proportional to the volume of all the node cubes of the slave node	✓	☒	✓	✓	✓	☒	☒
Transactional data management over the Cloud	Tiwari et al. [103]	Cost-effective	Possible inconsistencies due to data replication occurring only on the subset system of a small number of nodes	IG must have the minimum number of nodes	✓	✓	✓	☒	✓	☒	✓
Big-table: A distributed storage system for structured data	Chang et al. [104]	Dynamic control over data layout and format	Does not support a full relational data model, If Chubby is unavailable then Big-Table is unavailable	NA	✓	×	✓	✓	✓	✓	✓
PNUTS: Yahoo!'s hosted data serving platform	Cooper et al. [106]	Multiple records retrievals in parallel	No support for complex queries and join operations	NA	✓	✓	✓	☒	✓	✓	✓
Dryad: distributed data-parallel programs from sequential building blocks	Isard et al. [110]	Fine control over the communication graph and subroutines, Greater flexibility to easily compose basic common operations	Support sequential programs only	Vertices are deterministic	☒	✓	✓	✓	☒	✓	☒

Table 3 continued

Technique	Authors	Advantage(s)	Disadvantage(s)	Assumptions	SLA Metrics						
					AT	FT	ST	TP	LB	RT	CT
ElasTraS: an elastic transactional data store in the cloud	Das et al. [112]	Transactional guarantees in a scalable manner, Light-weight	ElasTraS provides Atomicity, Consistency, Isolation, and Durability (ACID) guarantees for transactions limited to a single partition	Reads and writes to the storage layer are atomic	✓	✓	✓	☒	✓	☒	✓
SQLMR: a scalable database management system for cloud computing	Hsieh et al. [116]	SQL-like queries can be used instead of writing MapReduce code	Placement of reducers is random, which may result in network load unbalancing and make the reducers on a busy rack become stragglers	Size of intermediate data is constant	✓	✓	✓	✓	×	☒	✓

by different protocols to maintain the consistency between the data. For example in [29], write operations are assumed to occur very rarely in the applications. When any write operation is to be performed, all other read and write requests are blocked, which causes a huge overhead, narrows the applicability, and limits the system dependability. Therefore, there is a need for an efficient technique that can effectively overcome the aforementioned limitations while balancing the cost of read and write operations. Conflict Resolution and data consistency is also main concerns in data replication. Several tools and techniques have been proposed to maintain consistency by using conflict resolution strategies [123]. However, the time between the failure and triggering the conflict resolution strategy is an issue.

- *Scalability problem:* The databases scales-up in many dimensions, such as data structures, no. of users, network complexity, size, and distance. Therefore, all of the aforementioned aspects must be addressed as a whole, so that the resources can be allocated in a way that can meet the requirements of a data intensive application [124]. The design alternatives for the development of distributed data management system are usually accompanied by some performance implications. The suitability of distributed transaction mechanisms, such as two-phase locking and two-phase commit protocol, in a WAN-based distributed environment is a big question. Moreover, the feasibility and scalability of protocols and algorithms as the system grows geographically, is another concern [125,126]. Furthermore, the overheads of replica control protocols, such as ROWA, increases as the number of replicas increases. For example, in ROWA, read operation is translated to one physical read but a write operation is translated to physical writes on all of the copies.
- *Requirements mismatch between DBMS and distributed operating system:* There is a requirement mismatch between the DBMS and the underlying operating systems. Functions, such as support for distributed transaction, concurrency control and recovery, management of distributed persistent data, and access methods, are either not provided or not fully supported by the distributed operating systems. Moreover, the support for the aforementioned functions can affect the performance of the systems [127]. Furthermore, the conventional functions, such as task and buffer management, perform by the distributed operating systems needs to be modified to support distributed DBMS [128]. The coupling of DBMS and distributed operating system is a complex task and the architectural framework of commercial operating systems does not allow a trivial incorporation of the considerations.
- *Communication channels:* Network communication and other related problems are always related to distributed environment. Similarly, data replication and management systems in cloud also have to deal with the said problems. Reliable communication channels have to be implemented to ensure data availability and to avoid data loss during normal operations. A lot of research has been done in developing group communication protocol, as in [129]. To ensure in order replica updates, a reliable multicast with total order is required in data replication and the presence of group communication limits the scalability for the aforementioned [129]. Moreover, the group communication layer makes the configuration and tuning of the network difficult. Furthermore, there are several tradeoffs between ease of configuration, flow control, and performance, such as TCP performance in network switches vs.

UDP multicast. Network latency and unreliability of long distance links are big hurdles in the extension of multi-master replication to WAN [130]. The bandwidth availability is increasing, but the physical constraints on latency over worldwide links limits the evolution of latency. DBMS communicate with database drivers using TCP connections, which offers reliable communication and in order packet delivery. However, to detect connections failure it relies on timeouts. In case of a network failure, the communication is blocked till the keep-alive time is expired. The aforementioned, results in an unacceptable long time of blockage (from 2 s to 2 h) that affects the performance and availability of data.
- *Energy efficient data management:* Efficient utilization of energy can radically reduce the cost of a cloud application. According to a certain estimate, the total operational expenditure of cooling and powering a data center is 53 % of the overall cost. In the U.S. the total energy consumed by the data centers is more than 1.5 % of the total energy produced by U.S., and the percentage is projected to grow 1.8 % annually [131]. Therefore, the CSPs have to develop a management platform that not only provides flexibility, scalability, multi-tenancy, and other services, but also reduces the energy consumption while meeting the standards and regulations implemented by the relevant governments. To overcome the challenge of implementing the aforementioned platform, a number of approaches from several authors have been proposed, such as discussed in [132] and [133]. Design of efficient hardware that can minimize energy use by powering off certain idle components is now trivial [134]. Moreover, there are software-oriented techniques to improve energy efficiency like Energy-Aware Job Scheduling [135,136], and Server Consolidation [137]. Some network protocols and infrastructures are also designed for the same purpose. Efficient routing tables can also be designed which can switch off the idle routes and save the energy.
- *Backups:* A fundamental part of replicated and data management systems is the backup. For large and problematic backups, the databases are taken offline. The performance of the systems is usually degraded during the backup operations. The backup time is not only the time to dump the data, it also involves the time to resynchronize the replicas by reapplying all the updates missed during the backup. Therefore, the backup time can take several hours depending on the size of database that results in a performance gradation.

Other issues related to the use of data replication and database management systems to support data management platforms over high performance computing systems, such as clouds, are listed below.

(a) There exists a lack of powerful I/O optimizations that can harness parallel I/O capabilities of current multiprocessor architectures [138].
(b) Data consistency and integrity semantics provided by almost all DBMS are an added obstacle in achieving high-performance.
(c) There exists a lack of application-specific access pattern information.
(d) Current I/O access strategies and optimizations are targeted at only a few well-defined access and storage patterns [138].
(e) Distributed query and transaction processing.

We have highlighted some of the aforementioned research issues involved in data replication and management in cloud. Future directions may involve striving for the solutions of the above issues. For example, one way to deal with the scalability issue is to develop measurement tools and performance models. However, performance models for distributed DBMS have not been extensively studied. A proper discussion for the solutions of the issues listed above can easily be the topic of a paper about the size of this paper. Therefore, we have only highlighted some of the open research issues, there are many other important technical problems that await solution and new ones may also arise as a result of the technological changes.

Acknowledgments The authors are thankful to Kashif Bilal and Osman Khalid for the valuable reviews, suggestions, and comments.

References

1. Mell, P., Grance, T.: Definition of cloud computing. Technical report, National Institute of Standard and Technology (NIST) (2009)
2. Bell, G., Gray, J., Szalay, A.: Petascale computational systems. IEEE Comp. **39**(1), 110–112 (2006)
3. Lamanna, M.: High-energy physics applications on the grid. In: Wang, Lizhe, Jie, Wei, Chen, Jinjun (eds.) Grid Computing: Infrastructure, Service, and Applications, pp. 433–458. CRC Press, Boca Raton (2009)
4. Khatib, Y., Edwards, C.: A Survey-Based Study of Grid Traffic. In: Proceedings of GridNets, pp. 41–48 (2007)
5. Gartner: Gartner top ten disruptive technologies for 2008 to 2012. Emerging trends and technologies roadshow http://www.gartner.com/it/page.jsp?id=681107, Accessed (2011)
6. Abadi, D.: Data management in the cloud: limitations and opportunities. IEEE Data Eng. Bull. **32**(1), 3–12 (2009)
7. Leinwand, A.: The Hidden Cost of the cloud: Bandwidth Charges, GIGAom, Jul. 17 2009, http://gigaom.com/2009/07/17/the-hidden-cost-of-the-cloud-bandwidth-charges/, Accessed May 12 (2011)
8. Sakr, S., Liu, A., Batista, D., Alomari, M.: A survey of large scale data management approaches in cloud environments. IEEE Commun. Survey Tutor. **09**, 1–26 (2011)
9. Cassandra: Available at http://incubator.apache.org/cassandra/, Accessed (2011)
10. Thusoo, A., Sarma, J., Jain, N., Shao, Z., Chakka, P., Anthony, S., Liu, H., Wyckoff, P., Murthy, R.: Hive-A warehousing solution over a MapReduce framework. In VLDB, pp. 1626–1629 (2009)
11. HBase: Available at http://hadoop.apache.org/hbase/, Accessed (2011)
12. Loukopoulos, Thanasis, Ahmad, Ishfaq, Papadias, Dimitris: An overview of data replication on the internet. In: Proceedings of the International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN.02), pp. 27–32 (2002)
13. Kia, H.S., Khan, S.U.: Server replication in multicast networks. In: 10th IEEE International Conference on Frontiers of Information Technology (FIT), Islamabad, Pakistan, pp. 337–341 (2012)
14. Khan, S.U., Ahmad, I.: A pure Nash equilibrium based game theoretical method for data replication across multiple servers. IEEE Trans. Knowl. Data Eng. **21**(4), 537–553 (2009)
15. Khan, S.U.: A frugal auction technique for data replication in large distributed computing systems. In: International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA), Las Vegas, NV, USA, pp. 17–23 (2009)
16. Khan, S.U., Ardil, C.: A competitive replica placement methodology for Ad Hoc networks. In: International Conference on Parallel and Distributed Computing Systems (ICPDCS), Oslo, Norway, pp. 128–133 (2009)
17. Khan, S.U., Ahmad, I.: Comparison and analysis of ten static heuristics-based internet data replication techniques. J. Parallel Distrib. Comput. **68**(2), 113–136 (2008)
18. Khan, S.U., Maciejewski, A.A., Siegel, H.J., Ahmad, I.: A game theoretical data replication technique for mobile Ad Hoc networks. In: 22nd IEEE International Parallel and Distributed Processing Symposium (IPDPS). Miami (2008)

19. Khan, S.U., Ahmad, I.: A pure Nash equilibrium guaranteeing game theoretical replica allocation method for reducing web access time. In: 12th International Conference on Parallel and Distributed Systems (ICPADS), Minneapolis pp. 169–176 (2006)
20. Khan, S.U., Ahmad, I.: Game theoretical solutions for data replication in distributed computing systems. In: Rajasekaran, S., Reif, J. (eds.), Handbook of Parallel Computing: Models, Algorithms, and Applications. Chapman & Hall/CRC Press, Boca Raton (2007). ISBN 1-584-88623-4, Chapter 45
21. Khan, S.U., Ahmad, I.: Data replication in large distributed computing systems using supergames. In: International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA), Las Vegas, pp. 38–44 (2006)
22. Khan, S.U., Ardil, C.: A frugal bidding procedure for replicating WWW content. Int. J. Inform. Technol. **5**(1), 67–80 (2009)
23. Khan, S.U., Maciejewski, A.A., Siegel, H.J.: Robust CDN replica placement techniques. In: 23rd IEEE International Parallel and Distributed Processing Symposium (IPDPS). Italy, Rome (2009)
24. Khan, S.U., Ardil, C.: A fast replica placement methodology for large-scale distributed computing systems. In: International Conference on Parallel and Distributed Computing Systems (ICPDCS), Oslo, pp. 121–127 (2009)
25. Wu, Y., Li, G., Wang, L., Ma, Y., Kolodziej, J., Khan, S.U.: A review of data intensive computing. In: 12th International Conference on Scalable Computing and Communications (ScalCom), Changzhou, (2012)
26. Khan, S.U., Ahmad, I.: A cooperative game theoretical replica placement technique. In: 13th International Conference on Parallel and Distributed Systems (ICPADS), Hsinchu, (2007)
27. Khan, S.U., Ahmad, I.: Replicating data objects in large-scale distributed computing systems using extended Vickery auction. Int. J. Comput. Intell. **3**(1), 14–22 (2006)
28. Gao, Aiqiang, Diao, Luhong: Lazy update propagation for data replication in cloud computing. In: 5th International Conference on Pervasive Computing and Applications (ICPCA), pp. 250–254 (2010)
29. Ikeda, Takahiko, Ohara, Mamoru, Fukumoto, Satoshi, Arai, Masayuki, Iwasaki, Kazuhiko: A distributed data replication protocol for file versioning with optimal node assignments. In: Proceedings of IEEE International Pacific Rim International Symposium on Dependable Computing 2010, pp. 117–125 (2011)
30. Khan, S.U., Ahmad, I.: Discriminatory algorithmic mechanism design based WWW content replication. Informatica **31**(1), 105–119 (2007)
31. Khan, S.U., Ahmad, I.: A semi-distributed axiomatic game theoretical mechanism for replicating data objects in large distributed computing systems. In: 21st IEEE International Parallel and Distributed Processing Symposium (IPDPS). Long Beach (2007)
32. Kohavi, R., Henne, R.M., Sommerfield, D.: Practical guide to controlled experiments on the Web: Listen to your customers not to the HiPPO. In: Proceedings of ACM International Conference on Knowledge Discovery and Data Mining (KDD 2007), pp. 959–967
33. Gulati, A., Merchant, A., Varman, P.: pClock: An arrival curve based approach for QoS in shared storage systems. In: Proceedings ACM International Conference on Measurement and Modeling of Computer System (SIGMETRICS), (2007)
34. Gulati, A., Merchant, A., Varman, P.: mClock: Handling throughput variability for hypervisor IO scheduling. In: Proceedings of the 9th OSDI, (2010)
35. Wang, J., Varmany, P., Xie, C.: Avoiding performance fluctuation in cloud storage. In: Proceeding of High Performance Computing (HiPC), pp. 1–9 (2010)
36. Goiri, I., Julia, F., Fito, J., Macias, M., Guitart, J.: Resource-level QoS metric for CPU-based guarantees in Cloud providers. In: 7th international workshop on economics of grids, Clouds, systems, and services, pp. 34–47 (2010)
37. Amrhein, D., Anderson, P., de Andrade, A., Armstrong, J., Arasan, E., Bartlett, J., Bruklis, R., Cameron, K., Cohen, R., Crawford, T. M., Deolaliker, V., Easton, A., Flores, R., Fourcade, G.: Review and summary of cloud service level agreements. http://public.dhe.ibm.com/software/dw/cloud/library/cl-rev2sla-pdf.pdf
38. Kliazovich, D., Bouvry, P., Khan, S.U.: Simulation and Performance Analysis of Data Intensive and Workload Intensive Cloud Computing Data Centers. In: Kachris, C., Bergman, K., Tomkos, I. (eds.) Optical Interconnects for Future Data Center Networks.Springer, New York, USA, ISBN: 978-1-4614-4629-3, Chapter 4

39. Goel, S., Buyya, R.: Data Replication Strategies in Wide Area Distributed Systems. Enterprise Service Computing: From Concept to Deployment, Robin G. Qiu (ed), pp. 211–241, ISBN 1-599044181-2, Idea Group Inc., Hershey (2006)
40. Pallickara, S.L., Pallickara, S., Pierce, M.: Scientific Data Management in the Cloud: A Survey of Technologies, Approaches and Challenges. Chapter 22: pp. 517–534, Handbook of Cloud Computing. Springer. ISBN: 978-1-4419-6523-3 (2010)
41. Ramakrishnan, R.: Data Management in the Cloud. In: Proceedings of IEEE 25th International Conference on Data Engineering(ICDE '09), pp. 5–5 (2009)
42. Gonzalez, L., Merino, L., Caceres, J., Lindner, M.: A break in the clouds: towards a cloud definition. Comp. Commun. Rev. **39**(1), 50–55 (2009)
43. Plummer, D., Bittman, T., Austin, T., Cearley, D., Smith, D.: Cloud Computing: Defining and Describing an Emerging Phenomenon. Technical report, Gartner (2008)
44. Staten, J., Yates, S., Gillett, F., Saleh, W., Dines, R.: Is cloud computing ready for the enterprise?. Technical Report, Forrester Research (2008)
45. Bojanova, I., Samba, A.: Analysis of cloud computing delivery architecture models. In: IEEE Workshops of International Conference on Advanced Information Networking and Applications (WAINA), Biopolis, pp. 453–458 (2011)
46. Abouzeid, A., Bajda-Pawlikowski, K., Abadi, D., Rasin, A., Silberschatz, A.: Hadoopdb: an architectural hybrid of MapReduce and DBMS technologies for analytical workloads. Publ. Very Large Database (PVLDB) **2**(1), 922–933 (2009)
47. Cooper, B., Baldeschwieler, E., Fonseca, R., Kistler, J., Narayan, P., Neerdaels, C., Negrin, T., Ramakrishnan, R., Silberstein, A., Srivastava, U., Stata, R.: Building a cloud for Yahoo!. IEEE Data Eng. Bull. **32**(1), 36–43 (2009)
48. Pfleeger, C.P., Pfleeger, S.L.: Security in Computing, 4th edn. Prentice Hall PTR, Upper Saddle River (2006)
49. Chen, Y., Paxson, V., Katz, R.H.: What's New about cloud Computing Security?, Technical Report UCB/EECS-2010-5, EECS Department, University of California, Berkeley (2010)
50. Ristenpart et al.: Hey, you, get off of my cloud! Exploring information leakage in third- party compute clouds. In: Proceedings of the 16th ACM Conference on Computer and Communication Security (CCS-09), pp. 199–212. ACM Press (2009)
51. Habib, S.M., Ries, S., Muhlhauser, M.: Cloud Computing Landscape and Research Challenges regarding Trust and Reputation. In: 7th International Conference on Ubiquitous Intelligence & Computing and 7th International Conference on Autonomic & Trusted Computing (UIC/ATC), 2010, pp. 410–415 (2010)
52. Person, S.: Taking account of privacy when designing cloud computing services, Technical Report, HPL-2009-54, HP Laboratories (2009)
53. Everett, C.: Cloud computing: a question of trust. Comput. Fraud Security **2009**(6), 5–7 (2009)
54. Dillon, T.S., Wu, C., Chang, E.: Cloud computing: issues and challenges, In: Proceedings of 24th IEEE International Conference on Advanced Information Networking and Applications (AINA-2010), pp. 27–33 (2010)
55. Mouline, I.: Why assumptions about cloud performance can be dangerous to your business. J. Cloud Comput. **2**(3), 24–28 (2009)
56. Goel, S., Buyya, R.: Data replication strategies in wide area distributed systems. In: Qiu, Robin G. (ed.) Enterprise Service Computing: From Concept to Deployment, pp. 211–241, ISBN 1-599044181-2, Idea Group Inc., Hershey, PA, USA (2006)
57. Ghemawat, S., Gobioff, H., Leung, S.-T.: The Google file system. In: Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles (Bolton Landing, NY, USA, 2003). SOSP '03. pp. 29–43 (2003)
58. Amazon.com: Amazon simple storage service (Amazon S3), http://aws.amazon.com/s3, Accessed on 2011
59. Gray, J., Helland, P., O'Neil, P., Shasha, D.: The danger of replication and a solution. In: Proceedings of International Conference on Management of Data ACM SIGMOD, Montreal, pp. 173–182 (1996)
60. Loukopoulos, T., Ahmad, I.: Static and adaptive distributed data replication using genetic algorithms. J. Parallel Distrib. Comput. **64**(11), 1270–1285 (2004)
61. Ullah Khan, Samee, Ahmad, Ishfaq: A pure Nash equilibrium-based game theoretical method for data replication across multiple servers. IEEE Trans. Knowl. Data Eng. **21**(4), 537-553 (2009)

62. Wei, Q., Veeravalli, B., Gong, B., Zeng, L., Feng, D.: CDRM: A cost-effective dynamic replication management scheme for cloud storage cluster. In: IEEE International Conference on Cluster Computing 2010, pp. 188–197 (2010)
63. Kangasharju, J., Roberts, J., Ross, K.: Object replication strategies in content distribution networks. In: Proceedings of Sixth International Workshop on Web Caching and Content Distribution (WCW '01), pp. 455–456 (2001)
64. Dowdy, L., Foster, D.: Comparative models of the file assignment problem. ACM Comput. Surveys **14**(2), 287–313 (1982)
65. Khan, S., Ahmad, I.: Heuristic-based replication schemas for fast information retrieval over the internet. In: Proceedings of 17th International Conference on Parallel and Distributed Computing Systems (PDCS '04), pp. 278–283 (2004)
66. Li, B., Golin, M., Italiano, G., Deng, X.: On the optimal placement of Web Proxies in the internet. Proc. IEEE INFOCOM '00 **1**(1), 1282–1290 (2000)
67. Qiu, L., Padmanabhan, V., Voelker, G.: On the placement of web server replicas. Proc. IEEE INFOCOM '01 **1**(2), 1587–1596 (2000)
68. Loukopoulos, T., Lampsas, P., Ahmad, I.: Continuous replica placement schemes in distributed systems. In: International Conference on Supercomputing (ICS'05) Boston, June 20–22
69. Chu, W.W.: Optimal file allocation in a multiple-computer information system. IEEE Trans. Comput. **C–18**, 885–889 (1969)
70. Chu, W.W.: Optimal file allocation in a computer network. In: Abramson, N., Kuo, F.F. (eds.) Computer-Communication Networks, pp. 83–94. Prentice-Hall, Englewood Cliffs (1973)
71. Casey, R.G.: Allocation of copies of files in an information network. In: Proceedings of AFZPS 1972 SJCC, vol. 40, pp. 617–625. AFIPS Press (1972)
72. Eswaran, K.P.: Placement of records in a file and file allocation in a computer network. In: Proceedings of the ZFZP Congress on Information Processing 1974, pp. 304–307. North-Holland, Amsterdam (1974)
73. Mahmoud, S., Riordon, J.S.: Optimal allocation of resources in distributed information networks. ACM Trans. Database Syst. **1**(1), 66–78 (1976)
74. Ramamoorthy, C.V., Wah, B.W.: The placement of relations on a distributed relational database. In: Proceedings of the 1st International Conference on Distributed Computing Systems (Huntsville, Ala., Oct. 1979). IEEE, New York, pp. 642–650 (1979)
75. Wah, B.W., Lien, Y.-N.: Design of distributed databases on local computer systems with a multiaccess network. IEEE Trans. Softw. Eng. **SE–11**(7), 606–619 (1985)
76. Wang, F., Oral, S., Shipman, G., Drokin, O., Wang, T., Huang, I.: Understanding lustre filesystem internals. Technical Report ORNL/TM-2009/117, Oak Ridge National Lab., National Center for Computational Sciences (2009)
77. Cloudstore (kosmosfs), http://code.google.com/p/kosmosfs/. Accessed 12 June 2012
78. Haddad, I.F.: PVFS: A parallel virtual file system for linux clusters. In: 4th Annual Linux Showcase and Conference, pp. 317–328. Atlanta (2000)
79. Huang, H., Hung, W., Shin, K.G.: FS2: dynamic data replication in free disk space for improving disk performance and energy consumption. In: Proceedings of the 20th ACM Symposium on Operating Systems Principles (SOSP 2005), (2005)
80. Bonvin, N., Papaioannou, T.G., Aberer, K.: A self-organized, fault tolerant and scalable replication scheme for cloud storage. In: Proceedings of the Symposium on Cloud Computing, pp. 205–216. Indianapolis, USA (2010)
81. Decandia, G., Hastorun, D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., Sivasubramanian, S., Vosshall, P., Vogels, W.: Dynamo: amazon's highly available key-value store. In: Proceedings of ACM Symposium on Operating Systems Principles, pp. 205–220. New York (2007)
82. Silvestre, G., Monnet, S., Krishnaswamy, R., Sens, P.: AREN: A Popularity aware replication scheme for cloud storage. In: IEEE International Conference on Parallel and Distributed Systems (ICPADS), pp. 189–196 (2012)
83. Ye, Y., Xiao, L., Yen, I., Bastani, F.B.: Cloud storage design based on hybrid of replication and data partitioning. In: Proceedings of IEEE Sixteenth International Conference on Parallel and Distributed Systems (ICPADS), pp. 415∼422. (2010)
84. Ye, Y., Yen, I., Xiao, L., Bastani, F.: Secure. Dependable and high performance cloud storage. Technical Report: UTDCS-10-10

85. Gupta, A., Liskov, B., Rodrigues, R.: One Hop lookups for peer-to-peer overlays. In: Proceedings of the Hot Topics in Operating Systems, Hawaii (2003)
86. Wang, F., Qiu, J., Yang, J., Dong, B., Li, X., Li, Ying: Hadoop high availability through metadata replication. In: Proceeding of the first international workshop on cloud data management, pp. 37–44 (2009)
87. Skeen, D., Stonebraker, M.: A formal model of crash recovery in a distributed system. IEEE Trans. Softw. Eng. **9**(3), 219–228 (1983)
88. Suresh, A.: HadoopT: Breaking the Scalability Limits of Hadoop. Diss, Rochester Institute of Technology, Rochester (2011)
89. Bessani, A., Correia, M., Quaresma, B., Andr'e, F., Sousa, P.: DepSky: Dependable and secure storage in a cloud-of-clouds. In: Proceedings of the European Conference on Computer Systems (EuroSys), pp. 31–46 (2011)
90. Francisco, R., Correia, M.: Lucy in the sky without diamonds: Stealing confidential data in the cloud. In: IEEE/IFIP 41st International Conference on Dependable Systems and Networks Workshops (DSN-W) (2011)
91. Tsai, W., Zhong, P., Elston, J., Bai, X., Chen, Y.: Service replication with MapReduce in clouds. In: Tenth International Symposium on Autonomous Decentralized Systems, pp. 381–388 (2011)
92. Cecchet, E., Singh, R., Sharma, U., Shenoy, P.: Dolly: virtualization-driven database provisioning for the cloud. In: Proceedings of the 7th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments, pp. 51–62 (2011)
93. Twin Peaks Software Inc. http://www.TwinPeakSoft.com. Accessed 04 May 2012
94. Twin Peaks Software Inc., Mirror File System for Cloud Computing. U.S Patent number: 7418439
95. MFS presentation at usenix.org fast 08, http://www.usenix.org/events/fast08/wips_posters/slides/wong.pdf
96. Curino, C., Jones, E., Zhang, Y., Madden, S.: Schism: a workload-driven approach to database replication and partitioning. In: VLDB, pp. 48–57 (2010)
97. Armbrust, M., Fox, A., Rean, G., Joseph, A., Katz, R., Konwinski, A., Gunho, L., David, P., Rabkin, A., Stoica, I., Zaharia, M.: Above the clouds: A Berkeley View of cloud Computing. Tech. Rep. UCB/EECS-2009-28, EECS Department, U.C. Berkeley (2009)
98. Khan, S.U., Min-Allah, N.: A goal programming based energy efficient resource allocation in data centers. J. Supercomput. **61**(3), 502–519 (2012)
99. Zhang, X., Ai, J., Wang, Z., Lu, J., Meng, X.: An efficient multi-dimensional index for cloud data management. In: Proceedings of cloudDB'2009, pp. 17–24
100. Sellis, T.K., Roussopoulos, N., Faloutsos, C.: The R -tree: a dynamic index for multi-dimensional objects. VLDB J., pp. 507–518 (1987)
101. Garcia-Molina, H., Ullman, J.D., Widom, J.: Database System Implementation. Prentice Hall Inc, Upper Saddle River (1999)
102. Haojun, L., Han, J., Fang, J.: Multi-Dimensional index on Hadoop Distributed File System. In: IEEE 5th International Conference on Networking, Architecture and Storage (NAS) (2010)
103. Tiwari, R.G., Navathe, S.B., Kulkarni, G. J.: Towards transactional data management over the cloud. In: proceedings of Second International Symposium on Data, Privacy, and E-Commerce, pp. 100–107 (2010)
104. Chang, F., Dean, J., Ghemawat, S., Hsieh, W., Wallach, D., Burrows, M., Chandra, T., Fikes, A., Gruber, R.: Bigtable: A distributed storage system for structured data. ACM Trans. Comput. Syst. **26**(2), 4 (2008)
105. Burrows, M.: The chubby lock service for loosely-coupled distributed systems. In: Operating systems design and implementation, pp. 335–350 (2006)
106. Cooper, B., Ramakrishnan, R., Srivastava, U., Silberstein, A., Bohannon, P., Jacobsen, H., Puz, N., Weaver, D., Yerneni, R.: Pnuts: Yahoo!'s hosted data serving platform. Publ. Very Large Database (PVLDB) **1**(2), 1277–1288 (2008)
107. Simmhan, Y., Barga, R., van Ingen, C., Lazowska, E., Szalay, A.: Building the Trident Scientific Workflow Workbench for Data Management in the cloud. In: Third International Conference on Advanced Engineering Computing and Applications in Sciences, 2009. ADVCOMP '09, pp. 41–50 (2009)
108. Hey, T., Trefethen, A.: The Data Deluge: An e-Science Perspective, in Grid Computing: Making the Global Infrastructure a Reality. Wiley, Chichester (2003)

109. Barnes, C.R., Bornhold, B.D., Juniper, S.K., Pirenne, B., Phibbs, P.: The NEPTUNE Project–a cabled ocean observatory in the NE Pacific: Overview, challenges and scientific objectives for the installation and operation of Stage I in Canadian waters. In: Symposium on Underwater Technology and Workshop on Scientific Use of Submarine Cables and Related Technologies, pp. 308–313 (2007)
110. Isard, M., Budiu, M., Yu, Y., Birrell, A., Fetterly, D.: Dryad: distributed data-parallel programs from sequential building blocks. In: European Professional Society for Systems (EuroSys), pp. 59–72 (2007)
111. Yu, Y., Isard, M., Fetterly, D., Budiu, M., Erlingsson, U., Gunda, P., Currey, J.: Dryad linq: A system for general-purpose distributed dataparallel computing using a high-level language. In: OSDI, pp. 1–14 (2008)
112. Das, S., Agrawal, D., Abbadi, A.E.: Elastras: An elastic transactional data store in the cloud. In: Workshop on Hot Topics in Cloud Computing (2009)
113. Gray, J., Reuter, A.: Transaction Processing: Concepts and Techniques. Morgan Kaufmann Publishers Inc., San Francisco (1992)
114. Weikum, G., Vossen, G.: Transactional Information Systems: Theory, Algorithms, and the Practice of Concurrency Control and Recovery. Morgan Kaufmann Publishers Inc., San Francisco (2001)
115. Aguilera, M.K., Merchant, A., Shah, M., Veitch, A., Karamanolis, C.: Sinfonia, A new paradigm for building scalable distributed systems. In: SOSP, pp. 159–174 (2007)
116. Hsieh, M., Chang, C., Ho, L.Y., Wu, J., Liu, P.: SQLMR : A scalable database management system for cloud computing. In: Proceedings of International Conference on Parallel Processing, pp. 315–324 (2011)
117. Gilbert, S., Lynch, N.: Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. SIGACT News **33**(2), 51–59 (2002)
118. Youn, H., Lee, D., Lee, B., Choi, J., Kim, H., Park, C., Su, L.: An efficient hybrid replication protocol for highly available distributed system. In: Proceedings of IASTED International Conference on Communications and Computer Networks, pp. 508–513 (2002)
119. Gifford, D.K.: Weighted voting for replicated data. In: Proceedings of 7th ACM Symposium on Operating Systems Principles, pp. 150–162 (1979)
120. Agrawal, D., Abbadi, A.: The tree Quorum protocol: an efficient approach for managing replicated data. In: Proceedings of 16th Very Large Database Conference, pp. 243–254 (1990)
121. Taheri, J., Zomaya, A.Y., Bouvry, P., Khan, S.U.: Hopfield neural network for simultaneous job scheduling and data replication in grids. Future Gener. Comput. Syst. **29**(8), 1885–1900 (2013)
122. Khan, S.U., Ahmad, I.: Replicating data objects in large distributed database systems: an axiomatic game theoretical mechanism design approach. Distrib. Parallel Databases **28**(2–3), 187–218 (2010)
123. Moiz, S.A., Sailaja, P., Venkataswamy, G., Supriya, N.: Database replication: a survey of open source and commercial tools. Int. J. Comput. Appl. **13**(6), 1–8 (2011)
124. Khan, S.U., Ahmad, I.: Non-cooperative, semi-cooperative, and cooperative games-based grid resource allocation. In: 20th IEEE International Parallel and Distributed Processing Symposium (IPDPS). Rhodes Island, (2006)
125. Garcia-Molina, H., Lindsay, B.: Research directions for distributed databases. IEEE Q. Bull. Database Eng. **13**(4), 12–17 (1990)
126. Stonebraker, M.: Future trends in database systems. IEEE Trans. Knowl. Data Eng. **1**(1), 33–44 (1989)
127. Razavi, A., Moschoyiannis, S., Krause, P.: Concurrency control and recovery management in open e-Business transactions. In: WoTUG Communicating Process Architectures, pp. 267–285 (2007)
128. Christmann, P., Härder, T.H., meyer-wegener, K., Sikeler, A.: Which kinds of OS mechanisms should be provided for database management. In: Nehmer, J. (ed.), Experiences with Distributed Systems, pp. 213–251. Springer, New York
129. GORDA Project: State of the Art in Database Replication Deliverable D1.1, http://gorda.di.uminho.pt/deliverables, Accessed on 08 June 2013 (2006)
130. Abdellatif, T., Cecchet, E., Lachaize, R.: Evaluation of a Group Communication Middleware for Clustered J2EE Application Servers. ODBASE, Cyprus (2004)
131. Energy, STAR Data Center Energy Efficiency Initiatives, http://www.energystar.gov/ia/partners/prod_development/downloads/EPA_Datacenter_Report_Congress_Final1.pdf?d7a4-0cec. Accessed 16 Aug 2012
132. Andersen, D.G., Franklin, J., Kaminsky, M., Phanishayee, A., Tan, L., Vasudevan, V.: FAWN: A fast array of wimpy nodes. In: Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles, pp. 1–14 (2009)

133. Szalay, A.S., Bell, G.C., Huang, H.H., Terzis, A., White, A.: Low-power amdahl-balanced blades for data intensive computing. ACM SIGOPS Oper. Syst. Rev. **44**(1), 71–75 (2010)

134. Nedevschi, S., Popa, L., Iannaccone, G., Ratnasamy, S., Wetherall, D.: Reducing network energy consumption via sleeping and rate-adaptation. In: NSDI'08: Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation, pp. 323–336, Berkeley (2008). USENIX Association

135. Goiri, I., Le, K., Haque, M.E., Beauchea, R., Nguyen, T.D., Guitart, J., Torres, J., Bianchini, R.: GreenSlot: Scheduling Energy Consumption in Green Datacenters. In: Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis, ACM, p. 20 (2011)

136. Khan, S.U., Bouvry, P., Engel, T.: Energy-efficient high-performance parallel and distributed computing. J. Supercomput. **60**(2), 163–164 (2012)

137. Marzolla, M., Babaoglu, O., Panzieri, F.: Server Consolidation in Clouds through Gossiping, TR UBLCS-2011-01. Department of Computer Science, University of Bologna, Italy (2011)

138. Shen, X., Liao, W., Choudhary, A., Memik, G., Kandemir, M.: A high-performance application data environment for large-scale scientific computations. IEEE Trans. Parallel Distrib. Syst. **14**(12), 1262–1274 (2003)