Fault Tolerance Study for Durable Storage on the Cloud

Xiaofei Zhang
Department of Computer Science and Engineering
HKUST
Clear Water Bay, Kowloon, Hong Kong
Email: zhangxf@cse.ust.hk

Lei Chen
Department of Computer Science and Engineering
HKUST
Clear Water Bay, Kowloon, Hong Kong
Email: leichen@cse.ust.hk

***Abstract*—Durability indicates a system's ability to preserve committed data in the long term. It is one of the key requirements for storage systems. Various fault tolerance strategies have been proposed for traditional distributed databases, peer-to-peer (P2P) systems, and grid systems. These strategies were all developed for specific platforms and application types and have been tailored to the characteristics of the underlying system architectures and application requirements. Cloud systems differ from these previous frameworks in that they are designed to support large numbers of customer-oriented applications, each with different quality of service (QoS) requirements and resource consumption characteristics. Moreover, most cloud architectures are deployed over large-scale geographically distributed infrastructures, raising challenges on data durability as well as system efficiency and scalability. In this paper, we investigated different approaches to the design of durable data cloud platforms. In particular, we consider prevailing cloud platforms and cloud-based applications and examine the impacts of different redundancy and repair strategies to enhance the durability of the data in the cloud. We verify the performance of various approaches to data durability in the Cloud through extensive simulations.**

I. Introduction

Cloud is a fast growing framework that serves for various enterprises and personal applications. On the one hand, it satisfies demands of huge computing resources from fast growing applications; on the other hand, it has high demand on durable storage. For example, data uploaded to pay-as-you-go online storage system are expected to be preserved for years even generations, and search engines will give more complete and precise results if well-defined data sets are properly maintained. Therefore, Cloud itself is required to be configured with a proper fault tolerant strategy to avoid data loss. However, component failures inside such a large scale distributed system are frequent due to the large number of commodity devices that datacenters employ, even though individual units have reasonably long mean time to failures. Report from Google[1] claims that, for a distributed system employing over 10 thousand nodes with MTBF (Mean Time Between Failure) of 30 years, 1-5% hard drives fail in the very first year. Such failures will raise the risk of losing data permanently or introducing inconsistency. Therefore, as a cornerstone of reliability, data durability must be addressed in the first place.

In fact, data durability, as a measurement of system ability to preserve committed data, has been studied along the evolution of traditional distributed database, P2P and Grid systems. Different fault tolerant strategies were proposed according to different system architectures and application characteristics. In traditional distributed databases, data are backed up with dedicated devices like RAID. Now this architecture cannot satisfy resource requirement of rapidly growing applications. Existing Grid database proposals[2][3][4] also consider durability. However, inherited from traditional databases, fault tolerant strategies of Grid database systems are implemented together with strict Atomic, Consistency and Isolation models, which leave little space for accommodating the needs of scalable applications with different QoS requirements. On the contrary, Cloud system provides relative loosely managed storage, like S3[5] from Amazon. Hence, applications of various QoS requirements and resource utilization patterns can scale up with the system size and clients' requests . Fault tolerant strategies for P2P systems focus on handling unpredictable behavior of peers[6][7][8], such as peers joining and leaving, which seldom happen in a Cloud system. Compared to P2P systems, Cloud is generally better organized, which guarantees stronger assumptions, and more aggressive fault tolerant strategies should be adopted to achieve better reliability and overall performance. Moreover, in Cloud, we should give high priority to the considerations usually missing in P2P systems, such as correlated failure and benefits from heterogeneous network model.

As shown in Table I, different from other infrastructures, Cloud demonstrates new features in both system architecture and application types. With various applications from both enterprises and individuals being deployed in Cloud, one critical new feature for Cloud is to satisfy different requirements on QoS and scalability. The data durability issue becomes more complex on Cloud. It requires flexible fault tolerant strategies to preserve committed data of various types of applications as long as possible. Therefore, it is necessary for us to re-explore the design of fault tolerant strategy for durable storage on Cloud.

For traditional distributed systems, fault tolerant strategies include repair schema and redundancy strategy, which refers to redundancy type, replica degree and replica placement. To refine the durability strategy for the Cloud, we must take the

978-1-4577-1637-9/11/$26.00 ©2011 IEEE

TABLE I
SIMPLE COMPARISON BETWEEN CLOUD AND OTHER DISTRIBUTED SYSTEMS

System	Architecture	QoS	Application Feature	Focus
Distributed Database	Cluster	High	Relational Query	Query Processing
P2P	Distributed Peers	Low	Insensitive to Fault	Peers' Unpredictability
Grid	Distributed Subdomains	High	CPU/IO Intensive	Resource Integration
Cloud	Distributed Datacenters	Medium	Flexible	Elasticity

following key issues into consideration:

1) Redundancy type must satisfy data feature and data updating schema;
2) Replica degree should maximize data durability and satisfy constraints from both the system and applications;
3) Replica placement highly relies on the network model and system failure distribution;
4) Data repair scheme should take the advantage that Cloud systems are well monitored.

To our best knowledge, so far there is no proposal designed for achieving durable storage on Cloud with the consideration of above all the issues. In this paper, we propose a complete fault tolerant strategy for general purposed durable storage on Cloud. Considering heterogeneous network models adopted by most Cloud systems, we conduct strategy design on the datacenter scale. For redundancy type, we find that erasure-coding[9] is good for applications with immutable data but may introduce unacceptable overhead for certain server side applications, like search engines. Replication is much simpler, and it fits to applications with general updating patterns. For the replica degree, we derive the upper bound of replica degree with respect to system and application constraints. With detailed analysis, we believe that replicas inside a datacenter should be placed at the nodes as less correlated as possible. Inside a datacenter, we adopt dynamic repair scheme for popular data records to improve the system's availability. We summarize our contributions in this work as follows:

1) We are the first to give elaborate explanation of data durability on Cloud, regarding the design of fault tolerant strategy for various featured applications on Cloud platform;
2) We provide a general guideline for choosing redundancy policy, including how to choose redundancy types and calculate the replica degree for maximum durability;
3) We propose a less-correlated based replica placement strategy inside a datacenter, which can effectively enable data to survive from correlated failures;

The rest of paper is organized as follows: Section 2 introduces necessary background. Section 3 describes our strategy for durable storage for a datacenter. Experiment results are reported in Section 4 and we conclude in Section 5.

II. BACKGROUND

Fault tolerance is critical to wide-area connected storage systems. It involves both redundancy strategy and data repair schemes. Redundancy strategy is actually a configuration of redundancy type, replica degree and replica placement. Data repair scheme defines when to initiate a repair and the way resources being allocated for repair. The decision of redundancy parameters and repair scheme can be inter-dependent on each other for a given system and application.

Redundancy is the duplication of data in order to reduce the risk of data loss. Two redundancy types employed in practical system designs are: simple replication and erasure-coding[9]. Erasure-coding is a data reconstruction technique, which divides a data block into m fragments and encodes it into h fragments. The original data block can be recovered as long as no more than $h - m$ fragments are lost. The factor $r_{\mathrm{e}} = h/m$ denotes extra storage cost. A comparison of two redundancy types is presented in Table II.

TABLE II
ERASURE-CODING VS. REPLICATION

Redundancy Type	Storage Efficiency	Network Control	Computing Overhead	Data Loss Immunity
Eraure Code	High	Easy	High	Strong
Replication	Low	Hard	Low	Weak

The replica degree is the number of replicas to create. Apparently a small replica degree means higher risk of losing data. However, larger numbers of replicas can only offer a bigger chance for data to survive a burst of failures, but it does not directly prolong data lifetime and definitely introduces huge maintenance cost. To determine the replica degree, we need to take multiple factors into consideration: the permanent failure distribution, the generation rate of new redundancy versus the data loss rate, the data consistency requirement, the synchronization mechanism, and the application featured I/O frequency and volume.

The replica placement strategy is also critical to fault tolerance. It configures where to store the newly created replica. Placement model is justified with the trade-off between how much overhead would be brought about for making decisions and data transfer, and how efficiently it helps prolonging data life time. Both heuristic and random selection methods are adopted in existing works to determine data placement. Random selection takes the assumption that failures are frequent and unpredictable, as well as that nodes' workload are homogeneous. This method is proved to be effective in some P2P networks[10]. Heuristics-based replica placement can be categorized into two types. One is to take the assumption that the system workload and node behavior can be well described by using stochastic models[6]. The other type is to put great effort on enhancing data availability, since availability implies durability[11][12].

The repair scheme is another critical consideration to obtain

TABLE III
A SIMPLE COMPARISON BETWEEN PREVAILING CLOUD SYSTEMS

Cloud System	System Organization	Application Feature			Handling Failure
		Object Size	Operation Type	Consistency	
GFS	Master-Slave	≥100MB	Mostly Appending	Lock-based	Master takes full responsibility
Dynamo	Decentralized Servers	1KB-1GB	All Types	Eventually	P2P's fault tolerant strategy
PNUTS	Decentralized Domains	Relatively Small	All Types	More Strict Eventually	Multi-level Redundancy

fault tolerance. It defines the control over data repair process, e.g. when to initialize a repair as well as how should the data reconstruction proceed. In some cases, aggressive schema is necessary, like proactive repair[13], which makes full use of idle bandwidth and storage to create as many replicas as possible. However, sometimes only necessary repair is performed to avoid overhead.

Fault tolerance is also a challenge for Cloud systems. Especially with more and more Cloud applications replacing the role of personal PC and workstations, there emerges the demand of making Cloud storage immune to permanent data loss. Cloud systems currently in practice all have their own strategies to handle failures. However, as these systems are designed for limited application types so far, there is no general fault tolerant strategy proposed for storage on Cloud. Table 3 gives a simple comparison of these systems and shows how they handle failures[14][15][16].

III. STRATEGY DESIGN

Different from unpredictable behavior of peers in a P2P system, Cloud is usually well managed, which allows for more aggressive fault tolerant strategies. Unlike Grid, Cloud is supposed to support multiple applications, and scales well with respect to system and application size. To address durability issue on Cloud, we must take application features into consideration. We believe that the following factors are important: System Organization(S.O.), Network Performance (N.P.), Consistency Model(C.M.), Application Features(A.F.) and Data Popularity(D.P.). Table 4 summarizes how they affect different aspects of fault tolerant strategy design.

TABLE IV
CONSIDERATIONS ON FAULT TOLERANT STRATEGY FOR CLOUD STORAGE

Strategy	S.O.	N.P.	C.M	A.F.	D.P.
Redundancy Type		√		√	
Replica Degree		√	√	√	√
Replica Placement	√	√	√		√
When to repair	√		√		√
How to repair		√	√	√	√

A. System Assumption and Data Loss Model

The physical presence of a Cloud-based storage system is a collection of geographically distributed datacenters connected with a wide-area network. A datacenter is built up on a collection of nodes, which could vary in computing and storage capacities, connected with an optimized network and maintained by a Cloud service provider. Multiple Cloud-based applications can be deployed in the same datacenter while one Cloud service can involve a group of datacenters. Cloud applications are different with respect to sizes, operations and QoS requirements, etc.

Since the underlying network model greatly affects fault tolerance strategy, a hybrid durability strategy is often favored and it should be designed to fit both the datacenter scale and global scale. We adopt the network model for datacenters given in [17], which proposes a fat tree structured network model, enabling any two nodes within a datacenter to communicate with each other with full port bandwidth. On the other hand, connections between datacenters are highly volume-sensitive and characterized with high latencies. Network connections on a global scale can be abstracted as a weighted label graph $G =< V, E, W, L >$. V represents a set of vertices, with each vertex representing a datacenter; E is the set of edges; W describes the set of weights on all the edges while L represents the set of attribute values of vertices. The weight $w_{ij}(v_i, v_j)$ is calculated through $Bandwidth(t)^{-1} + Latency(t)$ which measures the connection efficiency at certain time point. For application a_i, we define a vertex label is the Probability of Data Loss (PDL) in datacenter v, i.e. $Label(v, a_i) = PDL(v, a_i)$. We assume all the applications are independent from each other. Therefore, the labels of v for application a_i and a_j are not correlated.

Since the datacenter environment and Cloud-based applications are usually well monitored, we assume that we can tell whether a failure is permanent or temporary shortly after it happens.

Data loss is mainly caused by failures of hard drives. According to [18], the Probability of Disk Failure (PDF) of disks involved in Google's Cloud system follows Gamma distribution. Common Cause Failure (CCF) has been observed a lot in datacenters[19], which means more than one nodes can fail simultaneously due to the same cause. In our study, permanent data loss occurs only in the following cases: 1) All the hosts of a record fail simultaneously; 2) The last host of a record fails before it copies the data to another host. Considering the heterogeneous nature of Cloud, in this paper we only focus on the fault tolerant strategy in the datacenter scale.

B. Datacenter-Scale Durability

We first examine the system and application constraints on the design of redundancy strategy.

1) Redundancy Type: Erasure-coding is widely adopted by many large scale distributed systems like OceanStore[20],

PAST[21], CAN[22] etc. However, erasure-coding is a win only if the server availability is low. In other words, it fits the environment that features with unpredictable server behaviors[23]. It may make the workload of some server side applications such as key word search become heavy. According to [24],the latest encoding algorithm on powerful machines offers about 3.5GB/s encoding bandwidth, under the condition that fragment size fits system architecture features. We let N_{obj} denote the total number of records, ρ denote encoding(decoding) bandwidth, μ denote the average size of data records, N_{svr} denote the number of servers for encoding(decoding), B denote available system bandwidth, r_{e} and r denote storage overhead rate of erasure-coding and replication, respectively. We can have:

$$T_{\text{erasure-coding}} = \frac{N_{\text{obj}} \times \mu}{N_{\text{svr}} \times \rho} + \frac{N_{\text{obj}} \times \mu \times r_{\text{e}}}{B} \tag{1}$$

$$T_{\text{replication}} = \frac{V_{\text{update}}(t) \times r}{B} \tag{2}$$

$$\frac{V_{\text{update}}(t)}{N_{\text{obj}} \times \mu} \times r - r_{\text{e}} \leq \frac{B}{N_{\text{svr}} \times \rho} \approx 0 \tag{3}$$

Let $\alpha = V_{\text{update}}(t)/(N_{\text{obj}} \times \mu)$. α must be in the interval (0,1], and $\alpha = 1$ represents data immutable applications. The smaller α is, the more advantages replication takes. Besides, the above comparison given by (3) is made from the perspective of encoding speed and network constraints, the disadvantage of erasure-coding would be more obvious if the workflow is also considered. Another consideration is that the network within a datacenter is much more stable and efficient compared to WAN (Wide Area Network). Hense, network volume control is not at our first priority. We choose simple replication in datacenter scale.

2) Replica Degree: To prevent data loss, the safest way would be to put a replica on every node. However, the storage capacity of a node is limited. To determine the replica degree, the cost of data repair and consistency maintenance must be taken into consideration. While satisfying system constraints and application features, the replica degree which guarantees better data durability is preferred.

Consider a datacenter v being deployed with applications a_1,a_2,...,a_n, the total data size and replica degree of application a_i is Z_i and r_i, respectively. N storage nodes are employed with storage capacities $s_1, s_2, \ldots, s_N$. Then vector $< r_1, r_2, \ldots, r_n >$ must satisfy overall storage capacity constraint:

$$\sum_{i=1}^{n} Z_i r_i \leq \sum_{i=1}^{N} s_i \tag{4}$$

No matter how network traffics are scheduled, the maximal network bandwidth V_{Max} allowed for application a_i at one time point is limited. V_{Max} is a pay-up-front value decided by the owner of a_i. We divide network volume into two parts: V_{rp},volume for data repair, and V_{wl}, volume for normal workload. Apparently,

$$V_{\text{rp}} + V_{\text{wl}} \leq V_{\text{Max}} \tag{5}$$

Assume the life time distribution φ of a node is already known, then the failure probability of this node at time t can be calculated as $\int_0^t \varphi(x)dx$ (failed disk will be replaced immediately). Therefore the expected size of data loss for application a_i at time t is:

$$DataLoss(a_i, t) = \sum_{j=1}^{N} \frac{Z_i}{N} \int_0^t \varphi_j(x)dx \tag{6}$$

Let $\beta(t)$ be the available bandwidth for repair at time t. If μ_i is the expectation of a_i's record size, the worst repair workload V_{rp} can be formulated as follows: if $\beta(t) \leq \mu_i$,

$$V_{\text{rp}}(t) = \sum_{i=1}^{n} \frac{\beta(t) \times DataLoss(a_i, t)}{\mu_i} \tag{7}$$

if $\beta(t) \geq \mu_i$,

$$V_{\text{rp}}(t) = \sum_{i=1}^{n} DataLoss(a_i, t) \tag{8}$$

For each application a_i, its operation type and corresponding approximate network volume can be given by the owner of a_i, with $opr(a_i, t)$ and $opw(a_i, t)$ respectively denotes the reading and writing volume of a_i, V_{wl} is obtained by:

$$V_{\text{wl}}(t) = \sum_{i=1}^{n} (opr(a_i, t) + opw(a_i, t)) \tag{9}$$

Another constraint on the replica degree is the data consistency requirement. To guarantee system throughput, most Cloud systems only apply updates to one replica at a time, and a time window is allowed to propagate updates to all the other replicas. Assume the inconsistent window of application a_i is W_i, we can have:

$$opw(a_i, t) \times (r_i - 1) \leq \int_t^{t+W_i} \left(V_{\text{Max}} - V_{\text{rp}}(x) - V_{\text{wl}}(x)\right) dx \tag{10}$$

Resolving the inequalities formed by (4),(5) and (10) for a_i, we obtain,

$$r_i \leq V_{\text{Max}} \left(\lambda Z_i + \frac{Workload(t)}{W_i}\right)^{-1} \tag{11}$$

where λ is a constant parameter indicating the data loss rate. Applying this to each application, then the vector of $< r_1, r_2, \ldots, r_n >$ can be obtained. For application a_i, the probability of data loss can be calculated with

$$PDL(a_i, t) = \prod_{j=1}^{r_i} \frac{r_i - j + 1}{N - j + 1} \int_0^{t+T_{\text{rec}}(t)} \varphi_i(x)dx \tag{12}$$

which gives the probability of losing all the replicas of a record during the minimal reconstruction time $T_{\text{rec}}(t)$. Based on the network assumption given in section 3.1, $T_{\text{rec}}(t)$ on the datacenter scale is considered only proportion to replica size.

3) Replica Placement.: The way of placing replicas among nodes has effects on how fast a replica can be reconstructed if some hard disks fail permanently. Even with the network assumption given in section 3.1, the more switches involved between a connection of two nodes, the more traffic scheduling overhead may be introduced and the system is more likely to suffer from a link failure or network re-partitioning. Thus replicas should be placed as near as possible. However, correlated failures can take big portion of total failures. Nodes that plug into the same power bar or on the same rack or in the same room, are more likely to fail simultaneously due to unpredictable physical damages. In this case, data can never be reconstructed if all of its replicas are placed on nearby nodes. Therefore, preferred replica placement strategy should lower the risk of losing all replicas as much as possible and minimize underlying network burden as well.

Given a datacenter implementation, we can always partition it into k non-overlapping groups so that any two nodes from different groups are supposed not to suffer from correlated failure. We define the distance between two nodes or groups of nodes as the number of switches involved in their connection. If application a_i's replica degree $r_i \leq k$, then all the replicas should be put in r_i's nearby groups. If $r_i \geq k$, it implies that more than one replica will be assigned to the same group. In that case, within each group, the same less-correlation partition can be adopted recursively.

For better availability and durability of data, a node that has lighter workload and less failure history records is chosen to host data. However, continuously assigning replicas to the same node can easily result in imbalanced workload. So rescheduling of hosts to place replicas from time to time is necessary. Fortunately, this can be easily done with the help of some workload monitoring and scheduling techniques[25][26].

4) Popularity Based Dynamic Repair Strategy.: Proactive repair can effectively eliminate the peak of network volume when there is a burst of failure.However, proactively creating more replicas for each data record increases total replica degree and may bring more consistency maintenance cost. Doing popularity selection based proactive repair not only makes a better use of network bandwidth, but also greatly eliminates the negative impact of losing popular replicas.

Assume the available bandwidth of system at time t is $B(t)$, top p popular records are grant with different bandwidths to do the proactive repair. We assign bandwidth with respect to the popularity, so the bandwidth for record R_i to conduct proactive repair at time t is:

$$B_{\text{pro}}(R_i, t) = \frac{B(t) \times P(R_i)}{\sum_{i=1}^{p} P(R_i)} \tag{13}$$

The time interval to re-pick top p popular records is a system configuration.

IV. Experiments

We use simulation to analyze the implications of Cloud-based data storage techniques on data durability. Specifically we examine how the replica degree affects data loss and data repair volume on a datacenter scale; how different replica placement strategies affect data loss with CCF considered.

For a datacenter, we assume it is deployed in 4 rooms, each room has 16 racks and each rack contains 16 nodes. Nodes on the same rack have a higher risk of suffering from some common failures, which implies that nodes in different rooms are much less correlated and generally do not fail together. We leave the network implementation details out, and only focus on the network traffic volume and data loss rates under different fault tolerant configurations.

As Cloud is featured with various types of services or applications, which may significantly differentiate with each other in record size, I/O operation or CPU consumption, etc. The combination of all these features determines the final fault tolerant strategy for either single application or entire datacenter. In experiments, we have 3 applications deployed in the system, which are varied in record size, total storage consumption and workload (number of requests as well as updating volume).

A. Replica Degree vs. System Constraint in Datacenter

We simulate an application of 1000000 records running on a datacenter. Record size follows a $N \sim$(1MB, 0.1) distribution. Figure 1 shows the repair traffic volume introduced under different replica degree configurations.

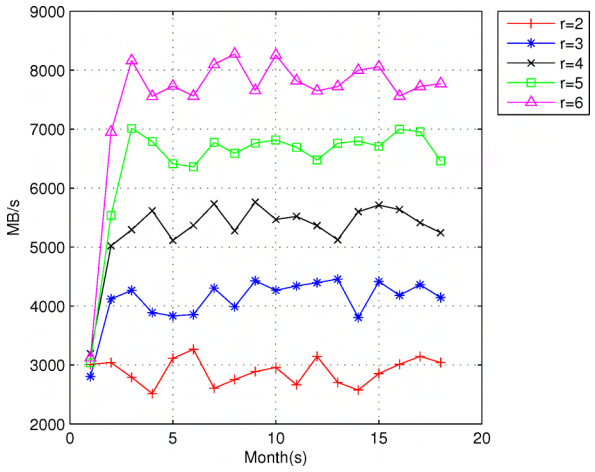

Fig. 1. Replica Degree vs. Repair Network Traffic Volume

As indicated in Figure 1, for certain replica degrees, the network volume brought up by repair is generally stable. Obviously the higher the replica degree the more network traffic volume required for repairing. Therefore, with system constraints specified, we can derive the maximal replica degree allowed for this application.

B. Replica Placement vs. Data Loss in Datacenter

As discussed in the previous section, a better option would be less-correlated based cost-effective placement. Simulation result shown in Figure 2 compares random placement with

our strategy. (We set CCF follow an exponential distribution as given in [27]).

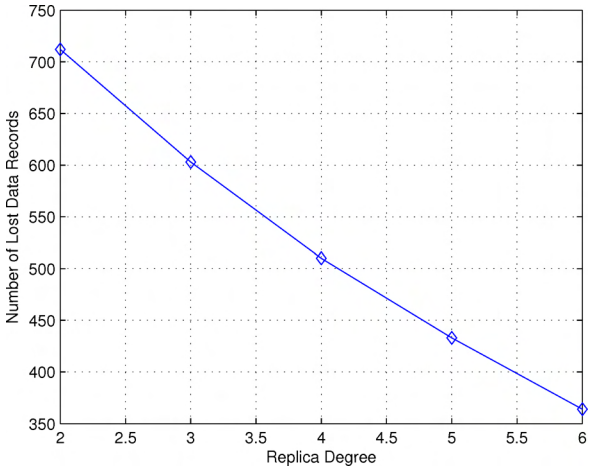

Fig. 2. Replica Degree vs. Data Loss

Apparently less-correlated placement well outperforms the others. Interestingly, as shown in Figure 2, there seems a linear relationship between replica degree and number of record loss. Thus, for different replica degrees, the mean network traffic volume required for repairing construct an arithmetic progression, which has been tested and verified in Figure 1. Also we can observe that with our proposal, the number of data loss drops faster when replica degree increases. This is because, even with more replica involved, our proposal guarantees that they are still less correlated.

V. Conclusion

By examining all the possible factors regarding data durability in Cloud-based data storage system, we re-explore the redundancy strategy and data repair scheme as well as their implications to data durability on the Cloud. While the simulation verifies our proposal on some level, we would like to carry on our study in some open-platform real-world Cloud systems as our future work.

References

[1] J. Dean, "Designs, lessons and advice from building large distributed systems. LADIS," 2009.

[2] S. Goel, H. Sharda, and D. Taniar, "Failure recovery in grid database systems," in *IWDC*, 2004, pp. 75–81.

[3] S. Venugopal, R. Buyya, and K. Ramamohanarao, "A taxonomy of data grids for distributed data sharing, management, and processing," *ACM Comput. Surv.*, vol. 38, no. 1, 2006.

[4] E. Cecchet, G. Candea, and A. Ailamaki, "Middleware-based database replication: the gaps between theory and practice," in *SIGMOD Conference*, 2008, pp. 739–752.

[5] M. Brantner, D. Florescu, D. A. Graf, D. Kossmann, and T. Kraska, "Building a database on S3," in *SIGMOD Conference*, 2008, pp. 251–264.

[6] S. Ramabhadran and J. Pasquale, "Durability of replicated distributed storage systems," in *SIGMETRICS*, 2008, pp. 447–448.

[7] B.-G. Chun, F. Dabek, A. Haeberlen, E. Sit, H. Weatherspoon, M. F. Kaashoek, J. Kubiatowicz, and R. Morris, "Efficient replica maintenance for distributed storage systems," in *NSDI*, 2006.

[8] K. Kim and D. Park, "Reducing replication overhead for data durability in dht based p2p system," *IEICE Transactions*, vol. 90-D, no. 9, pp. 1452–1455, 2007.

[9] H. Weatherspoon and J. Kubiatowicz, "Erasure coding vs. replication: A quantitative comparison," in *IPTPS*, 2002, pp. 328–338.

[10] R. Bachwani, L. Gryz, R. Bianchini, and C. Dubnicki, "Dynamically quantifying and improving the reliability of distributed storage systems," in *SRDS*, 2008, pp. 85–94.

[11] F. Wang, J. Qiu, J. Yang, B. Dong, X. H. Li, and Y. Li, "Hadoop high availability through metadata replication," in *CloudDb*, 2009, pp. 37–44.

[12] A. Haeberlen, A. Mislove, and P. Druschel, "Glacier: Highly durable, decentralized storage despite massive correlated failures," in *NSDI*, 2005.

[13] A. Duminuco, E. Biersack, and T. En-Najjary, "Proactive replication in distributed storage systems using machine availability estimation," in *CoNEXT*, 2007, p. 27.

[14] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The google file system," in *SOSP*, 2003, pp. 29–43.

[15] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels, "Dynamo: amazon's highly available key-value store," in *SOSP*, 2007, pp. 205–220.

[16] B. F. Cooper, R. Ramakrishnan, U. Srivastava, A. Silberstein, P. Bohannon, H.-A. Jacobsen, N. Puz, D. Weaver, and R. Yerneni, "Pnuts: Yahoo!'s hosted data serving platform," *PVLDB*, vol. 1, no. 2, pp. 1277–1288, 2008.

[17] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in *SIGCOMM*, 2008, pp. 63–74.

[18] E. Pinheiro, W.-D. Weber, and L. A. Barroso, "Failure trends in a large disk drive population," in *FAST '07: Proceedings of the 5th USENIX conference on File and Storage Technologies*. Berkeley, CA, USA: USENIX Association, 2007, pp. 2–2.

[19] J. Borcsok, S. Schaefer, and E. Ugljesa, "Estimation and evaluation of common cause failures," in *ICONS '07: Proceedings of the Second International Conference on Systems*. Washington, DC, USA: IEEE Computer Society, 2007, p. 41.

[20] J. Kubiatowicz, D. Bindel, Y. Chen, S. E. Czerwinski, P. R. Eaton, D. Geels, R. Gummadi, S. C. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Y. Zhao, "Oceanstore: An architecture for global-scale persistent storage," in *ASPLOS*, 2000, pp. 190–201.

[21] P. Druschel and A. I. T. Rowstron, "Past: A large-scale, persistent peer-to-peer storage utility," in *HotOS*, 2001, pp. 75–80.

[22] S. Ratnasamy, P. Francis, M. Handley, R. M. Karp, and S. Shenker, "A scalable content-addressable network," in *SIGCOMM*, 2001, pp. 161–172.

[23] N. Cruces, R. Rodrigues, and P. Ferreira, "Pastel: Bridging the gap between structured and large-state overlays," in *CCGRID*, 2008, pp. 49–57.

[24] J. S. Plank, J. Luo, C. D. Schuman, L. Xu, and Z. Wilcox-O'Hearn, "A performance evaluation and examination of open-source erasure coding libraries for storage," in *FAST*, 2009, pp. 253–265.

[25] E. Elmroth and L. Larsson, "Interfaces for placement, migration, and monitoring of virtual machines in federated clouds," in *GCC '09: Proceedings of the 2009 Eighth International Conference on Grid and Cooperative Computing*. Washington, DC, USA: IEEE Computer Society, 2009, pp. 253–260.

[26] J. Brandt, A. Gentile, J. Mayo, P. Pebay, D. Roe, D. Thompson, and M. Wong, "Resource monitoring and management with ovis to enable hpc in cloud computing environments," in *IPDPS '09: Proceedings of the 2009 IEEE International Symposium on Parallel&Distributed Processing*. Washington, DC, USA: IEEE Computer Society, 2009, pp. 1–8.

[27] L. Xing and W. Wang, "Probabilistic common-cause failures analysis," in *RAMS '08: Proceedings of the 2008 Annual Reliability and Maintainability Symposium*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 354–358.