How good are Classic Distributed Algorithms for Replica Management in Cloud Services?

Position Paper

Rafael Pereira, Flávio Ribeiro
WebMedia, Globo.com
Rio de Janeiro, Brazil
{rafael.pereira, flavio.ribeiro }@corp.globo.com

Markus Endler
Departamento de Informática, PUC-Rio
Rio de Janeiro, Brazil
endler@inf.puc-rio.br

Abstract— **Almost every cloud platform has some dependable core services based on replicated data/state and which require strong consistency among the replicas. And as these replicas may be hosted in geographically distributed data-centers, cloud platform's consistency preserving algorithms are challenged by unpredictable communication latencies and temporary network partitions. On the other hand, in the last three decades, much research on algorithms for distributed replica consistency has been done, and some have been successfully incorporated into practical systems. In this work we analyze the specific consistency requirements and the common distributed deployments of cloud platforms, and discuss to which extent classic distributed algorithm design has contributed to the solution of the internet-cloud problems.**

Keywords - Distributed Architecture, Cloud Computing, Distributed Algorithms

I. INTRODUCTION

Cloud computing is a paradigm shift following the change from mainframe to client–server in the early 1980s. Details are abstracted from the users, who no longer have need expertise in, or control over, the technology infrastructure "in the cloud" that supports them. Cloud computing describes a new supplement, consumption, and delivery model for IT services based on the Internet, and it typically involves over-the-Internet provision of dynamically scalable and often virtualized resources.

An intrinsic characteristic of cloud computing platforms is the intense use of distributed computing resources, with services that employ distributed storage and processing across multiple servers, sometimes geographically dispersed, which requires the implementation of solutions for data/state replication that preserve consistency of different replicas.

However, there are numerous challenges associated with the implementation of those, since a significant part of the communication necessary for replica and consistency management is accomplished through the Internet, where network connectivity is neither predictable nor reliable, with frequent variations of latency, jitter and temporary network partitions, which directly impact the efficiency of communication. Consequently, maintaining consistent replicas becomes a challenge that must be effectively addressed by cloud computing services, so they can provide the reliability required for different types of applications hosted on them. For example, a cloud based storage service must tolerate multiple failures without data loss or unavailability. Amazon Simple Storage Service (Amazon S3), is one that provides 99.999999999% durability and 99.99% availability of objects over a given year [1], as consequence of their replicated and fault tolerant architecture.

In the last three decades, much research on distributed algorithms for replica consistency has been done, and some have been successfully incorporated into practical systems. Algorithms for fault detection, election and consensus, are examples of key components for the construction of reliable and fault tolerant distributed storage and processing services, which main characteristic is the existence of consistent replicas to ensure data and state redundancy. However, much of which was developed and researched in this area did not take into account the current state of cloud computing, where, again, much of the communication is performed through the Internet, with links that are unreliable and susceptible to different types of errors, and where most services are in data centers geographically distributed around the globe.

Thus, the main objective of this paper is to analyze how the fundamental algorithms used to maintain consistent data/state replicas of distributed systems behave with the new paradigm posed by cloud computing, discussing how they contributed to the evolution of this new computing model and what are the main challenges that this new scenario introduces to the existing algorithms, clarifying issues which still need to be addressed so that we have solutions even more efficient and reliable.

Paper Outline. Next Section presents the evolution of research in key distributed algorithms used for replica management, discussing details about Leader Election, Consensus and Failure Detection. In Section 3, the replication and consistency challenges introduced by the cloud computing paradigm are analyzed, and the recent work regarding distributed algorithm in the Cloud is presented and discussed. Section 4 presents cases of distributed algorithms usage and limitations on cloud-based services, and, in Section 5, the emergent demands from cloud services are

discussed. Finally, in Section 6 we highlight some future research and development opportunities in distributed algorithms for cloud computing services, concluding with an overall analysis in Section 7.

II. EVOLUTION OF RESEARCH IN RELATED DISTRIBUTED ALGORITHMS

In this section, we summarize the past developments of research in following key problems associated with the two classes of replica management - active replication and primary-backup approach: Leader Election, Consensus/Agreement and Failure Detection. Instead of embracing the impossible task of highlighting the most notorious algorithms that have been proposed for each problem, we decided to focus at the discussion of the gradual shift towards more practical system hypothesis and concerns addressed by the research community.

Leader Election: The standard model for communication in the leader election problem is that all communication among the nodes is by broadcast. Moreover, each node has a unique identity (known by all), the identity of the broadcast senders known by all nodes, and the leader selection is based on the total order among the node ids. Most algorithms, however, handle dynamic sets of nodes, intermittent network partitions and node failures. Although many algorithms do not make any assumptions about communication latencies (using asynchronous rounds for synchronize the node's roles), most of them only converge when the network has reached a period of stability [2]. More recently, the scalability problem [3], as well as stability performance and several other QoS parameters have been considered important requirements of leader election [4,5].

Consensus/Agreement: The Consensus and Byzantine Agreement are two closely related fundamental problems of agreement on a common value in distributed systems. These two problems have been extensively researched and applied during the last decades, but most of the proposed algorithms are suited for either fully connected, broadcasting networks and/or generalized connected networks. But since cloud systems do not have any such standard network topology, most of the proposed algorithms cannot be directly adapted to cloud systems without deep analysis on how cloud computing resources are distributed, interconnected and managed.

Failure Detection: Since the seminal work by Chandra and Toueg [6], Failure Detectors have been recognized and applied as an important abstraction in the design and implementation of Agreement and Leader Election algorithms, as well as for many other distributed protocols and services. Research on FD has then focused both on alternative abstractions -e.g. the Phi-*Accural FD* [7], on analysis of the networking conditions under which some QoS properties such as: detection time, mistake recurrence time and mistake duration, can be guaranteed [8], as well as on efficient and scalable implementations.

These key algorithms were extensively and successfully used for a wide variety of distributed applications, in order to address replication and consistency issues, guaranteeing scalability and fault tolerance according to application's requirements. However, a different computing paradigm is emerging, and the assumptions used to guide the development of such algorithms may be different in the light of this new model. In the next section, we introduce the challenges and replication/consistency requirements of cloud-based services, presenting what had been developed regarding distributed algorithms in cloud computing.

III. CHALLENGES IN CLOUD COMPUTING

Cloud computing infrastructures are typically broken down into three layers [9,10]: "software" (SaaS), "platform" (PaaS) and "infrastructure" (IaaS). Each layer serves a different purpose and offers different products to businesses and individuals around the world, and, conversely, every layer can be perceived as a customer of the layer below [10].

Despite being classified in different ways, and serving different purposes, the layers of cloud computing (SaaS, PaaS and IaaS) have similar requirements with respect to replication, consistency and fault detection, however, applied to distinct goals and subject to different requirements. In general, all three layers use replication and fault detection to ensure reliability and availability of services provided by them to the upper layers and users. However, each layer deals with different levels of abstraction (e.g. the lowest level, IaaS, considers only bare physical and virtual resources, whereas the highest level, SaaS, handles application data and functionality, etc.). Thus, replication and fault detection solutions at the three levels employ somewhat different monitoring and replica management mechanisms and also have quite different availability, consistency and performance requirements.

In the case of IaaS, there is the provision of infrastructure services, particularly with respect to storage, structured or unstructured, and processing. In this scenario, replication and fault detection are used to ensure data and states consistency. One example of structured storage service in the Cloud that has strong replication and availability requirements is Amazon Dynamo. The Dynamo was designed to be a highly scalable key-value store that is highly available to reads but particularly for writes. This system is designed to make progress even in the presence of network partitions. The high write availability is achieved through an asynchronous replication mechanism, which acknowledges the write as soon as a small number of replicas have written it. The write is eventually propagated to other replicas. In the presence of failures, high availability is achieved at the cost of lower consistency [11].

Replication and consistency also are requirements for the PaaS layer. This middle layer usually uses services provided by the IaaS platforms to implement software components and architectural stack for applications. In such scenario, replication is used to achieve redundancy and scalability. Heroku is an example of PaaS service that is built over Amazon AWS services, and provides web server stack

management and deployment solutions to web applications. Heroku uses replication to provide scalability, increasing its reads capacity. Google App Engine, another PaaS service, launched two years ago the Google App Engine Datastore. Its Master/Slave replication architecture was designed to support quick, strongly consistent reads, while enabling fast writes that are immediately available. Google also have the High Replication Datastore, which provides a higher level of availability for both read and writes. This was achieved at the expense of increased latency for writes and changes in consistency guarantees. The High Replication Datastore increases the number of data centers that maintain replicas of data by using the Paxos algorithm [12] to synchronize that data across datacenters in real time. One of the most significant benefits is that all functionality of applications will remain fully available during planned maintenance periods, as well as during most unplanned infrastructure issues.

Finally, since the SaaS layer provides software to end users, at this level, the replication concerns are basically centered around application data, and the main goal is to achieve highest availability and in a scalable way. Furthermore, the consistency restrictions are usually dependent on the application's specific purpose. Twitter search engines, for example, are scaled up by partitioning the indexes and replicating the partitions [13], but in this case the replicated partitions do not have strong consistency requirements, but are well-served with some eventual consistency model. In fact, for most social networking applications, content portals, or media distribution services, weak- of eventually consistency is widely accepted and used.

Despite their different abstraction levels and requirements, at the three levels, all replication management and consistency maintenance approaches essentially deal with some of the three classic distributed system problems: Leader-Election, Consensus and Failure Detection. However, the most of the algorithms addressing these problems that have been proposed in literature cannot be applied to cloud systems, but just serve as an inspiration for the design of customized algorithms that take into account the specific characteristics of cloud infrastructure deployments, cloud platforms, and applications. In the following section we present some previous works towards this idea.

A. LITERATURE ABOUT DISTRIBUTED ALGORITHMS IN CLOUDS

There have been a number of proposed consensus algorithms for cloud computing. Yan et al [14] present a Dual Consensus Protocol that handles also malicious and dormant communication links. The underlying network topology has two levels: where links connect nodes of the same level as well as nodes of different levels. A similar topology was used in [15] which proposed a protocol for the agreement problem with fallible processes in a cloud system. In [16] the authors also propose a specific cloud computing topology and a consensus protocol for this topology that reaches agreement with minimum round of message exchange. Cheng *et al* [17] present a consensus algorithm for a distributed system where consensus servers, processors and communication links can be faulty at the same time. Their approach is based on a hierarchical organization, where the consensus servers perform most of the communication and computation. In [18] the authors report their very interesting experience of using (and adapting) the Paxos consensus algorithm to implement the distributed fault tolerant locking service Chubby, which is used by several Google elements such as GFS and Bibtable. The main conclusions of the authors were that "there were significant gaps between the [...] Paxos algorithm and the needs of real-world system", and that "the fault-tolerance computing community has not developed the tools to make it easy to implement their algorithms."

In regard to Failure Detectors, there is no significant difference between its use in a general network setting and in Cloud computing, since processor and link failure modes are independent of their deployments. Nevertheless, there are some interesting extensions. Gabel et al. [19] presents a technique for latent fault detection in cloud systems. The proposed method is based on statistical tests and compares only machines performing the same task at the same time. The key idea is to tag a machine as faulty when it deviates from the normal behavior. The proposed failure detection is centralized, but was tested on several services of various sizes and natures. Regarding Leader Election, we could not find papers proposing/discussing specific algorithms/protocols for Cloud systems. Instead, it seems that simple variations or adaptations of well-known Leader Election algorithms are used for specific centralized coordination task within a cloud platform, such as mentioned in [20].

IV. REPORTED ISSUES ON THE USE OF CLOUD-BASED SERVICES

From 1980s to today, the manner in which the Internet is used has dramatically changed. The access and passive consumption of simple static content on web portals gave rise to a lot of central points where the information is exchanged dynamically by millions of users. With this increasingly active participation of users and high content generation arising from them, the Internet and its content delivery model needed to be remodeled.

Given this scenario, some techniques have been developed and evolved to make it possible to manage and store data in a scalable way. The most notorious solution were the creation or adoption of Cloud computing services, which attempt to combine a dynamic allocation of hardware resources and algorithms based on distributed computing for data persistence and scalability.

The fact is that although many algorithms and techniques have been applied successfully in the cloud computing environment, there are scenarios in which the management of a large mass of content is impaired. Specifically for the largest companies were the core business reaches users in a scale of thousands or millions like multimedia streaming and

social networks, issues related to the application of techniques for reliability and consistency of the data/state exchanged on cloud computing services were reported.

Globo.com, the internet branch of Globo Corporation, produces live broadcasts of events related to sports, entertainment and news within its web portal and reaches up to 200,000 concurrent users at its peak. Globo.com's stream distribution is supported by an internal CDN (Content Delivery Network) composed of a set of geographically distributed servers that are strategically located in different parts of Brazil.

There are events transmitted by Globo.com only available to premium users. Thus, a security solution based on Consensus/Agreement fundamentals that controls access of these users must be propagated across the servers infrastructure. Since some parts of the infrastructure are connected through the Internet, maintenance and propagation of this data is directly related to network conditions and this leads directly to race condition problems for delays in the timing and synchronization between the servers.

Another problem reported also in Globo.com's multimedia distribution system happened in the recommendations service for videos. The fact is that the system runs on a hybrid infrastructure consisting of internal servers and Amazon Web Services EC2, the Amazon's elastic computing platform. In a period of unavailability of Amazon Web Services, the synchrony between the replicas of the database of the recommendations and watched videos were lost. When Amazon's EC2 became available, the attempt to sync data between replicas from internal servers and Amazon's EC2 to maintain the consistency of the database at all caused a high demand for the service as a whole increasing its response time and causing instability for the systems that consume the service.

Netflix, an on-demand video distribution company with more than 20 million subscribers and responsible for 30% of all residential downstream internet traffic in North America during peak times [21], followed its philosophy of openness and published an open letter pointing out the lessons learned by the problems that has happened with the use of cloud services from third parties. A very interesting concept used by Netflix is that their services are designed to fail, so where possible, it serves stateful content in stateless services which any node can respond to any request for content, routing users to other nodes when the node required fails [22].

In other periods of unavailability of Amazon Web Services, many other companies and their services were hampered as the recommendations service of Globo.com and streaming service of Netflix. Chartbeat, a service for monitoring and reporting visits in websites used by huge companies around the world had a major problem. Besides the unavailability of the service, it reported a lack of approximately 11 hours of customer data [23] due to the instability and failures in storage services from Amazon. Just to mention, another well-known company called Foursquare, a location-based social network which have more than 10 million users and is hosted at Amazon [24, 25], also reported several downtime and latency issues directly related to problems of unavailability in Amazon EC2 and Amazon S3, the Amazon Simple Storage Service [26].

Apart from the cases related on this section, many other users of cloud services have reported problems of inconsistency of data applications that are often aligned with synchrony problems and availability of the cloud as a whole, and initiatives to alleviate these problems have already been created.

One that can be cited is how the request for data is made in the Amazon's SimpleDB, where the user can choose between consistent or eventually consistent read on the method call. Another important concept is the *read-your-write,* a technique used to improve the performance of read data. Basically, the node in the cloud that wrote the information has a higher performance in reading it. Several other concepts are being created but even these initiatives are still based on traditional algorithms for failure detection and consensus [27]. Thus, a new field of research for this development has opened.

V. EMERGENT DEMANDS OF CLOUD SERVICES

As reported in the previous section, numerous problems are emerging due to the use of cloud-based services. Companies lose large amounts of money and directly impact consumer satisfaction. Apparently, as pointed on the *Literature About Distributed Algorithms in Clouds* there are evidences where studies and research done in the field of science over the years has necessitated some adjustments or even real changes to work in this scenario.

The first question to be analyzed is the minimum necessary for a reasonable functioning of distributed systems. Twelve years ago, Eric Brewer has created an interesting concept called CAP Theorem. CAP is an acronym of Consistency, Availability and Network Partition, and it asserts that any networked shared-data system can have only two of three desirable properties. This year, Brewer found that explicitly handling partitions, designers could optimize consistency and availability, thereby achieving some trade-off of all three [28]. This can be seen as an opportunity to research, since it opens a range of possibilities with regard to the way the partitions need to be handled to optimize the achievement of the three variables - Consistency, Availability and Partitions - considered as the most important by the theorem.

Regarding the first variable of the CAP Theorem, the consistency, different versions of eventual consistency methodologies like *read your writes* have been created and applied frequently. Other interesting concepts, such as the *Monotonic Read Consistency* which if the process has seen a particular value for the object, any subsequent accesses will never return any previous values, and *Session Consistency* that can be seen as a version of *read your writes* in the context of a session also are already described in the literature [29]. Other ways of classifying data consistency and the way they are fetched is still an open field and new methodologies and concepts can be applied.

As the CAP Principle, other approaches to mapping requirements to good service in a cloud have also been

empirically established. This is the case of Werner Vogels, who associated some variables with conditions of a distributed system and established some relations and formulas upon these variables [29]. Just to symbolize one of the hypothesis that Vogels cited, if **N** is the number of nodes that store replicas of the data, **W** the number of replicas that need to acknowledge the receipt of the update before the update completes and **R** the number of replicas that are contacted when a data object is accessed through a read operation, **W+R>N** indicates a scenario where strong consistency can be achieved. Others cases are cited, but the fact is that many other empirical hypotheses can and should be created in coming years.

Lastly, users are investigating the approach of create more layers on the cloud computing scheme. Although it seems initially ambiguous, research in this direction has justified the investment. Wingu, The Synchronization Layer For Safe Concurrent Access to Cloud [27] is nothing more than a locking-based system using the theory of mutual exclusion, but in distributed systems area. In this case, Wingu generates a time overhead due to the extra layer, but this extra time eliminates the window of inconsistency and it prevents conflicting updates. Proposals for another new layers of abstraction are an unexplored field.

Is important to note that some features that needs a large processing power and massive data management on the cloud are being questioned by companies their real necessity of real-time response. It is possible to notice that some social networks are making use of workarounds supported by techniques in interface design and human computer interaction to disguise what would be a problem of rapid spread of consistent data [30, 31]. An interesting field that has evolved is the way the user interacts with the system so that does not require the engines to return results in real time without harming the experience of the visit.

VI. IMPACT OF NEW REPLICA CONSISTENCY REQUIREMENTS ON ALGORITHM DESIGN

As can be noticed, the demands of distributed cloud services are somewhat different from the problems typically addressed by classic distributed algorithms. In fact, there are important situations in which cloud computing developers depend upon data or service replication, since applications are tuned for rapid response, elasticity, and scalability by exploiting a mixture of aggressive replication and very loose coupling between replicas [32]. However, the considerations that guide algorithms design to achieve replica consistency are different in such environment. Thus, concepts like connectivity model, membership model, failure mode, between others, have to be redefined for the design of cloud-based replication consistency algorithms.

Firstly, for cloud-based services, distributed algorithms should consider non-uniform (heterogeneous) network and failure models, and be designed for two-level hierarchical networks which are common to cloud systems, i.e. at the basic level, data-centers with large amounts of processing nodes with a dependable common power supply and interconnected within reliable and high-performance network, and the un within data-centers at one level, and at the top level, the widely distributed mesh of interconnected data-centers, which must rely on conventional Internet routing for synchronization of data across countries and continents. In this model, the distributed algorithms should consider specific behavior of its nodes, as well as particular communication frequencies between any two nodes, according to the nodes' location. In such scenario, the node connectivity model used in consistency algorithm design is composed simultaneously by groups of reliable and unreliable links, where unreliable links present delays, data loss, and can isolate a group of reliably interconnected nodes.

Furthermore, the membership model from the replication scheme is different for cloud-based replication algorithms. Basically, cloud platforms vary their set of nodes elastically, where the actual collection of members changes over time, perhaps rapidly. Full replication necessitates tracking the set, having a policy for initializing a newly launched service instance, and ensuring that each update reaches all the replicas, even if that set is quite large [32]. Therefore, these membership dynamics must be factored into the model, and coordination with cloud management infrastructure is required to avoid disruptive elasticity events.

An important issue that should be considered more at the design of distributed algorithm for clouds is the their common requirement for updates/synchronizations of large data sets, as opposed to small and frequent data updates, as assumed by conventional distributed algorithms. This means that algorithms must be designed to handle concurrent *bulk data transfers,* to/from key nodes of the cloud system. Although this data-volume problem may be simply regarded as a capacity planning issue, in our understanding algorithm design will most often have a big impact on this issue. Moreover, it will be very important to carefully determine on which node/s of a data center, and in which data-center/s, some processes of the algorithm should be executed. And the available resources versus the expected peak volumes of data transfers may largely determine this choice. In other words, cloud-specific distributed algorithms should use system models that also consider the available resources (for processing, storage, and communication) of the underlying hardware and network infrastructure.

Finally, the failure mode used for cloud-based algorithms has to assume that network packages can be lost and that applications may fail by crashing. If a network partition occurs, the affected servers are treated as if they had crashed, and, when the partition is repaired, affected nodes are restarted.

A. FUTURE RESEARCH AND DEVELOPMENT

With different demands from distributed cloud services it is possible to highlight several interesting directions of future research and development.

As also noted in [18], there is still lack of adequate tools and environments to facilitate the implementation, debugging and testing of distributed algorithms for real-world systems, and more precisely, for large cloud infrastructures. In spite of a multitude of FIRE (Future

Internet Research and Experimentation)-like test bed initiatives and platforms (OpenLab, TEFIS, etc.), apparently there is still a huge demand for development and test platforms for distributed algorithms, and which support algorithmic design, their stepwise refinement, validation and tuning, as well as their testing for large- and wide-scale systems.

VII. CONCLUSIONS

The increasing demand for efficient processing of large volumes of information promotes the research on architectures and techniques that optimize the use of available resources. Cloud computing is a new paradigm that is becoming increasingly popular, since it allows computing resources to be used on demand, increasing flexibility, scalability, and reducing costs.

In this paper we discussed how design of fundamental distributed algorithms, such Leader Election, Consensus and Fault Detection, are being applied to cloud computing. The paper highlights that, since a significant part of the communication in a Cloud infrastructure is done through the Internet, where frequent variations of communication latency, jitter or temporary network partitions directly impact the efficiency of communications, network resources are an important element that needs to be considered when implementing a cloud service that requires high availability and replica consistency.

We also pointed out that despite having different layers with different purposes, i.e. IaaS, PaaS and SaaS, cloud computing employs replication management and consistency maintenance solutions that essentially deal with some of the three classic distributed system problems. However, for practical implementations, there are still significant differences between the models typically used by classic distributed algorithms and the reality of cloud systems, such as non-uniform networks, manipulation of large data sets, frequent variations of the computing infra-structure as consequence of continuously resource provisioning and un-provisioning, node failures, etc.

These limitations became clear with issues reported by existing large cloud services, where carefully-designed replica consistency maintenance algorithms apparently are unable to handle bursts of interdependent replica synchronizations. In fact, to achieve high availability and scalability, some cloud services are trying to avoid strong replica consistency, and instead offer different flavors of weak or eventually consistency.

Finally, the last important remark is that there are several opportunities of future research and development regarding replication and consistency in cloud based services, and also in tools that support implementation, debugging and testing of distributed algorithms for real-world systems.

REFERENCES

[1] Amazon Simple Storage Service - http://aws.amazon.com/s3/

[2] Garcia-Molina, H.; Elections in a Distributed Computing System,IEEE Trans. on Computers, C-31(2),48-59, 1982.

[3] King, V.;, Saia, J.: Sanwalani, V.; Vee, E.; Scalable leader election. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm* (SODA '06). ACM, pp. 990-999, 2006.

[4] Schiper, N.; Toueg, S.; , A robust and lightweight stable leader election service for dynamic systems," *Dependable Systems and Networks With FTCS and DCC, 2008. DSN 2008. IEEE International Conference on* , vol., no., pp.207-216, 24-27 June 2008.

[5] Aguilera M.K.;, Delporte-Gallet, C.; Fauconnier, H.; Toueg, S.: Stable Leader Election. Proceedings of DISC 2001: pp. 108-122.

[6] Chandra T.D.; Toueg, S.; Unreliable Failure Detectors for Reliable Distributed Sysems, Journal of the ACM, vol. 43, no. 2, pp. 225-267, 1996.

[7] Hayahibara, N.; Défago, X.; Yared, R.: Katayama, T., The Phi-Accrual Failure Detector, *Technical Report, IS-RR-2004-10, Japan Advanced Institute of Science and Technology*, 2004.

[8] Chen, W.; Toueg, S.; Aguilera, M.K.; On the Quality of Service of Failure Detectors, *IEEE Trans. on Computers*, Vol, 51, No. 5, May 2002.

[9] Vaquero, L. M., Rodero-Merino, L., Caceres, J., Lindner, M. (2009) A Break in the Clouds: Towards a Cloud Definition, ACM SIGCOMM Computer Communication Review, Volume 39, Number 1, January 2009

[10] Q. Zhang, L. Cheng, R. Boutaba. Cloud Computing: State-of-the-Art and Research Challenges. Journal of Internet Services and Applications(JISA). Springer London, Vol. 1(1), pp. 7-18, May 2010.

[11] Data Management Challenges in Cloud Computing Infrastructures

[12] L. Lamport, Paxos Made Simple, 01 November, 2001.

[13] Florian Leibert, Jake Mannix, Jimmy Lin, and Babak Hamadani; Automatic Management of Partitioned, Replicated Search Services; SOCC 2011

[14] S. C. Wang, K. Q. Yan, and C. P. Huang, "Consensus Under Cloud Computing Environment within Malicious Faulty Transmission Media", The E-Learning and Information Technology Symposium, Tainan, Taiwan, 1 April, 2009.

[15] S. C. Wang, S. S. Wang, K.Q. Yan and C. P. Huang,, The Anatomy Study of Fallible Processes Agreement for Cloud Computing, *International Journal of Advanced Information Technologies (IJAIT)*, Vol. 4, No. 2, December 2010.

[16] Chanchary, F.H.; Islam, S.; Challenges of Using Consensus Protocol in Cloud Computing, ICCIT, 2012.

[17] C. F. Cheng, S. C. Wang and T. Liang, "Investigation of Consensus Problem over Combined Wired/Wireless Network", *Journal of Information Science and Engineering 25*, pp 1267-1281, 2009.

[18] Chandra, T.D.; Griesemer R.; Redstone, J.; Paxos Made Live - An Engineering Perspective, *Proc. of ACM Princ. of Distributed Computing, pp. 398--407, 2007.*

[19] Gabel, M.; Gilad-Bachrach, R.; Bjorner, N,; Schuster, A.; Latent Fault Detection in Cloud Services, Microsoft Research, Tech. Rep. MSR-TR-2011-83, 2011.

[20] Bonvin, N.; Papaioannou, T.G.; Aberer, K.; A Self-Organized, Fault-Tolerant and Scalable Replication Scheme for Cloud Storage, SoCC 2011.

[21] Janko Roettgers, Infographic: Netflix by the Numbers. August 9, 2011. http://gigaom.com/video/netflix-by-the-numbers/ visited on July 4, 2012.

[22] Adrian Cockroft, Cory Hicks, Greg Orzell, Lessons Netflix Learned from The AWS Outage. The Netflix Tech Blog, April 29, 2011. http://techblog.netflix.com/2011/04/lessons-netflix-learned-from-aws-outage.html visited on July 4, 2012.

[23] Henry Blodget, Amazon's Cloud Crash Disaster Permanently Destroyed Many Customers' Data. Business Insider. April 28, 2011. http://articles.businessinsider.com/2011-04-28/tech/29958976_1_amazon-customer-customers-data-data-loss visited on July 4, 2012.

[24] Cody Barbierri, Foursquare: 5 million users, 25.000 new ones a day. Venturebeat Mobile. December 10, 2010.

http://venturebeat.com/2010/12/08/foursquare-25000-new-users/ visited on 4 July, 2012.

[25] http://aws.amazon.com/solutions/case-studies/foursquare/

[26] Andrew Hickey, New Amazon Cloud Outage Takes Down Netflix, Foursquare. CRN. August 09, 2011. http://www.crn.com/news/cloud/231300459/new-amazon-cloud-outage-takes-down-netflix-foursquare.htm visited on July 4, 2012.

[27] Craig Chasseur, Allisson Terrell, Wingu - A Synchronization Layer For Safe Concurrent Access to Cloud. December 22, 2010.

[28] Eric Brewer, CAP Twelve Years Later: How the "Rules" Have Changed. IEEE Computer Society. February 2012.

[29] All Things Distributed: Eventually Consistent - Revisited. December 22, 2008. http://www.allthingsdistributed.com/2008/12/eventually_consistent.html visited on July 4, 2012.

[30] Adil Aijaz, The Engineering behind LinkedIn Products "You May Like", LinkedIn Blog. March 2, 2011. http://blog.linkedin.com/2011/03/02/linkedin-products-you-may-like/ visited on July 4, 2012.

[31] The Engineering Behind Twitter's New Search Experience. Twitter Engineering Blog. May 31, 2011. http://engineering.twitter.com/2011/05/engineering-behind-twitters-new-search.html visited on July 4, 2012.

[32] Kenneth P. Birman, Daniel A. Freedman, Qi Huang, Patrick Dowel. Overcoming CAP with Consistent Soft-State Replication; IEEE Computer, 2012