AJIT-e: Online Academic Journal of Information Technology
2016 Fall/Güz – Cilt/Vol: 7 - Sayı/Num: 25
DOI: 10.5824/1309-1581.2016.4.001.x
http://www.ajit-e.org/?menu=pages&p=details_of_article&id=232

Received : 03.05.2016 *Editorial Process Begin: 15.06.2016* *Published: 11.10.2016*

Investigation of Amazon and Google for Fault Tolerance Strategies in Cloud Computing Services

Shereen Al-RAHEYM, *Human Development University (UHD), Sulaimania, Iraq, shereen.al-raheym@uhd.edu.iq*

Sinan Can AÇAN, *METU, Computer Center, Ankara, Turkey, canacan@metu.edu.tr*

Ö. Tolga PUSATLI, *Cankaya Univ., Dept. of Mathematics, Information Technologies, Ankara, Turkey, pusatli@cankaya.edu.tr*

ABSTRACT *Cloud computing has recently become an attractive topic due to its ability to offer information technology solutions through virtual machines as on-demand services to share and consume resources over the Internet. As a result of rapid development in such services, the necessity of fault tolerance in the cloud is a major concern with reliability, availability and dependability which are more critical to this new service type. This work investigates techniques and means of tolerating cloud services as well as cloud customers' systems/enterprises execution over the cloud safe from failures. Failures in cloud enabled services should be expected to occur hence they should be handled. The essential features of implementing fault tolerance strategies guarantee the business continuity, avoid financial lost, recovering systems from failures, and provide disaster recovery as well. The specific focus is to explore scenarios of avoiding/recovering from failures through redundancy, checkpoint and replication. Commercial IaaS providers such as Amazon's AWS and Google's GCE are taken as examples as they tolerate their infrastructure from failures; in this way a robust architecture with fault tolerance property could be built for a system/enterprise. Hence, general conceptual steps with fault tolerance considerations have been proposed.*

Keywords *: cloud computing, fault tolerance, reliability, availability, dependability, redundancy, checkpoint, replication, AWS, GCE*

AJIT-e: Online Academic Journal of Information Technology
2016 Fall/Güz – Cilt/Vol: 7 - Sayı/Num: 25
DOI: 10.5824/1309-1581.2016.4.001.x

Amazon ve Google Bulut Bilişim Servislerinin Hata Dayanıklılığı Stratejileri Açısından İncelenmesi

ÖZ *Bulut bilişim, bilgi teknolojileri çözümlerini, talep üzerine sanal cihazlarla aracılığı ile İnternet üzerinden sunarak kaynakları paylaşma ve tüketme yeteneği sayesinde, yakın zamanda cazip bir konu haline gelmiştir. Bahsedilen bu hizmetlerin hızlı gelişimi sonucunda, bulut bilişimde daha kritik olan güvenilirlik ve bulunurluk hizmetleriyle, hata dayanıklılığı gerekliliği, bu hizmet tipinde önemli bir kaygı konusudur. Bu çalışmada, bulut hizmetlerinin hata dayanıklılığı yöntem ve teknikleriyle birlikte bulut hizmeti alan müşterilerin sistemlerinin ve/veya işletmelerinin bulut üzerinde hatadan etkilenmeyecek şekilde çalışması incelenmiştir. Bulut üzerinde çalışan sistemlerde hata beklenmelidir ve hata oluştuğunda da giderilmelidir. Hata dayanıklılığı stratejilerini oluşturmaktaki temel amaç iş sürekliliğini sağlamak, parasal kayıplardan kaçınmak, sistemlerdeki arızaları gidermek ve felaketlerden kurtarmaktır. Çalışmanın özel odağı, yedeklilik, denetim noktası kullanma ve çoklama kullanarak hatalardan kaçınma ya da hataları giderme senaryoları üzerindedir. Ticari altyapı hizmeti (IaaS) sunan Amazon'un AWS ve Google'ın GCE hizmetleri, altyapılarının hatalara karşı dayanıklı olduğu için örnek olarak alınmıştır. Bu sayede bir sistem ya da işletme için hata dayanıklılığı olan güçlü bir mimari yapı kurulabilir. Bu çalışmada hata dayanıklılığı için gerekli genel kavramsal adımlar önerilmiştir.*

Anahtar Kelimeler: *bulut bilişim, hata dayanıklılığı, güvenilirlik, bulunurluk, yedeklilik, denetim noktası, çoklama, AWS, GCE*

Introduction

Nowadays, cloud computing is a popular paradigm. It can be identified as a business model that offers online computing resources. It aims to deliver on-demand large amount of IT infrastructure such as (servers, storage applications, network, services) for multi-users to share and consume as a utility in a virtualized manner and highly abstracted from cloud service providers (e.g. AWS, Azure, GCP). However, virtualization considers as the backbone of cloud computing Kepes (2011) in which cloud technology could never be without it. The concept of virtualizing a computer system's resources is based on adding a layer between the hardware and operating system that allows diverse operating system instances working simultaneously upon a single server. Physical resources, including: processors, memory, storage, network, and I/O devices are dynamically partitioned and shared. Carlin and Curran (2012)

Fault tolerance is an important key issue in cloud computing. It is the means to guarantee the availability and reliability (dependability) of critical cloud services as well as applications' executions and fulfilling their functions correctly even in the presence of failure affecting a system resources (e.g. hardware, software, network, overflow, timeout, power, or database lose). Basically, fault tolerance techniques are employed through the procurement or the development level of the system, so that, it is a survival attribute of cloud computing systems to satisfy the quality of service (QoS) requirement which are offered in service level agreement (SLA). (Ganga, Karthik, & Paul, 2012; Latchoumy & Khader, 2011; Pullum, 2001)

The remainder of this paper is organized as follows: next section surveys the relation between fault tolerance and dependability in a distributed system. We addressed fault tolerance approaches, methods, and techniques on the cloud in this section. Next section explores fault tolerance within commercial infrastructure cloud computing providers. Following that, we proposed a general conceptual model with fault tolerance considerations. Next section addresses related work. Finally, the conclusion of the paper and our future work are presented in the last section.

Background

Fault tolerance aims to achieve dependability in a system. Hence, system dependability is an essential objective of fault tolerance. Generally speaking, dependability is justified the trustworthiness of a system to deliver services to its customers Dubrova (2013). Figure 1 clarifies the tree of the major dependability characteristics

Dependability attributes

The dependability tree illustrates its attributes; two primary attributes are reliability, and availability:

- Reliability: is the continuity of delivering correct services without disruption like loss of data or code reset during execution; it is a function of time so that it is related to the mean time to failure (MTTF) and the mean time between failure (MTBF). Moreover, mean time to repair (MTTR) is the difference between MTTF and MTBF. As a result, MTBF=MTTF+MTTR (Latchoumy & Khader, 2011).

DOI: 10.5824/1309-1581.2016.4.001.x

- Availability: can be defined as the immediate readiness of the system to perform the services or the tasks when it is asked to do it. Availability = (MTTF) / (MTTR + MTTF) (Selic, 2006). Basically, availability in the cloud can be attained by redundancy throughout the replication of services or data and spreading them across various resources (Patel, Taghavi, Bakhtiyari, & Junior, 2013). System availability is measured by downtime per year. Table 1 shows standard values of availability and their corresponding downtime (Dubrova, 2013).

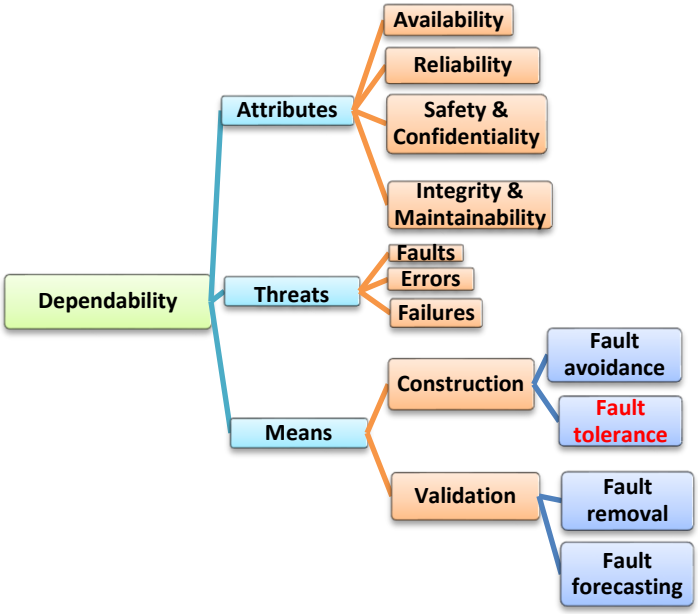


Fig. 1. Dependability tree compiled from (Latchoumy & Khader, 2011; Pullum, 2001)

Dependability threats

On one hand, a fault is software or hardware failure, defect, shortcoming, or flaw that leads to an incorrect or an inaccurate value on the computational level, which is called an error. On the other hand, a failure is a system problem that is caused by the error. In other words, faults are the sources of errors and errors are the sources of failures. So that, a fault is the root of failure as defended in (Dave & Raghuvanshi, 2012).

Dependability means

Four techniques are used to achieve dependability. On one hand, means that are deployed during the process of software configuration or construction with the goal of providing trust services to customers (Avizienis, Laprie, Randell, & Landwehr, 2004), they are:

- Fault tolerance, which is the subject of this work that aims to avoid an application or service failures' events if a fault happens. Early examples include (Laprie, 1995).

• Fault avoidance/prevention, which aims to prevent or reduce faults from occurring or introduce as possible. (Lussier et al., 2005)

On the other hand, techniques are deployed after the software development with the ability to reach confidence such as presented in (Pullum, 2001):

• Fault removal, which detects and eliminates the existence of fault during the development and operational life of the system.

• Fault forecasting, that evaluates, estimates or ranks the system behavior during the present or activation of faults.

Fault tolerance solutions are based on redundancy. So that, redundancy is one of the keynote principle for supporting availability and reducing the risk of single points of failure (VMware, 2009). A diversity of resource redundancy techniques and structures exist for fault tolerance mechanism which can be categorized into four essential categories (H. Li, Shang, Dang, & Jin, 2009):

• Spatial redundancy: Extra copies of the hardware computing resources are added to find out and overcome the impact of component failure. In advance, static, dynamic, or hybrid hardware redundancy can be used depending on the system structure complication (Runge, 2012).

• Temporal redundancy: Also called time redundancy, by which additional time can be used to re-execute the same task several times on the same resources in the event of a failure. Time redundancy has an advantage for detecting fault effectively with a low hardware cost (H. Li et al., 2009).

• Information redundancy: Additional information, which is called check bits are added to the original to data help protecting from soft error. This form of redundancy needs hardware redundancy to process check bits.

• Software redundancy: Extra software is added to override software failure.

Interestingly, managing and recovering failure in cloud form is different from what is happening in traditional datacenters. The redundancy in the cloud is managed and formulated in the software instead of hardware components. For example, Amazon EC2 offers a wide variety options such as direct failed virtual machine to another virtual machine image VMI, replications, or live migration and all failed virtual machine configurations are same to original (Babcock, 2010).

In most current cloud, checkpointing and replication are the two common fault tolerance redundancy strategies which are used in case of a system outage.

Table1. Standard availability values

DOI: 10.5824/1309-1581.2016.4.001.x

Availability	Downtime/ year	Downtime/ month
90%	36.5 days	72 hours
95%	18.25 days	36 hours
97%	10.96 days	21.6 hours
98%	7.30 days	14.4 hours
99%	3.65 days	7.20 hours
99.5%	1.83 days	3.60 hours
99.8%	17.52 hours	86.23 minutes
99.9%	8.76 hours	43.2 minutes
99.95%	4.38 hours	21.56 minutes
99.99%	52.56 minutes	4.32 minutes
99.999%	5.26 minutes	25.9 seconds
99.9999%	31.5 seconds	2.59 seconds
99.99999%	3.15seconds	0.259 seconds

Checkpointing/Restart (C/R) recovery

C/R is the typical technique to tolerate failure on unreliable systems. By saving a snapshot of running application on a stable storage periodically so as to restart the application from a latest checkpointing image in case of a crash (Y. Li & Lan, 2011).

Various types of checkpointing strategies had been investigated by researchers, but there are three popular checkpointing fault tolerance strategies

Full checkpointing: It is the traditional mechanism, which saves the total state of the application or the system periodically to a storage platform. The drawback of this mechanism is the time which is consumed to make a snapshot of a whole system. And also the consumed of a large storage to save the whole system running states (Gokuldev & Valarmathi, 2013).

Incremental checkpointing: Typically, the first checkpoint is full while the subsequent checkpoints only save pages that have been modified. This procedure produces a large recovery overhead due to the system must recover from the starting checkpoint (Garg & Singh, 2011).

Hybrid checkpointing: It is a combination between the full checkpointing and the incrementing strategies. Hence, a balance between the checkpointing overhead and the fault recovery overhead should achieve (Sun, Chang, Miao, & Wang, 2013).

Replication

The availability of replicated resources is a key requirement for the forming of fault tolerant systems in the cloud. Simply, replication means several copies of an application with the same input-set are executed simultaneously on alternative sites (Ghoreyshi, 2013). For example, proxy server and the caching in web browser can be considered as a form of replication. The essential goal of replication is to guarantee that at least one replica can complete the task correctly in case the others fail (Latchoumy & Khader, 2011). More than one replication mechanisms such as active, passive, or semi-active have been used in the cloud computing.

Basically, Hadoop, Ha proxy (high availability proxy) and Amazon EC2 are the common tools that are implemented in order to manage replicas on different locations (Ganga et al., 2012). Moreover, three important problems should be addressed to achieve efficient replication: "Which data should be replicated?", "When to replicate in cloud systems", "How many suitable new replicas should be created in the cloud?", and "Where the new replica should be placed?" (Sun et al., 2013).

Fault tolerance collective studies for commercial cloud IaaS providers

To understand how fault tolerance nowadays operates in cloud computing services, Figure 2 and Figure 3 explore Amazon web services (AWS) and Google Compute Engine (GCE) infrastructure as a service. Many tools and features are provided within the service that capable of designing fault tolerance (FT), disaster recovery (DR), and high available (HA) systems. In other side there are further infrastructure building blocks include fault tolerance solutions by default.

2006 is the start point of Amazon Web Services. At the core of Amazon virtual computing resources is an EC2 instance "virtual machine server". Amazon EC2 is constructed over multiple (eleven) geographical locations known as regions. In December 2013 Google Cloud launched Google Compute Engine GCE as a new entry to the market providing IaaS. Google group datacenters are distributed in (three) regions worldwide. Both providers overcome failure in a region, throughout splits each region into two or more Zones which are geographically isolated from each other in the same region, but they are connected with low latency network connection.

AWS first step in EC2 is to launch a VM instance throughout, creating an Amazon Machine Image (AMI), it is a master template that helps to define instances such as (web server, application server, etc...). From one AMI multiple instances can be established. Moreover, AMI's instances can be scaled up/down (Amazon Web Services, 2016). In GCE different virtual machine servers can be launched and all resources within a region can be accessed by zones within it throughout static IP address that is provided by default and GCE promise to launch VMI with instances management property (Ferraioli, 2014). It is safety for critical application/system to keep a spare of an instance in different AZ and keeps it running, in case a hot instance fails, the activity is taken by just redirecting users' requests to a new instance. Moreover, VM instances are not replicated automatically in the same or different regions, it is customers' mission and need their interaction.

DOI: 10.5824/1309-1581.2016.4.001.x

An instance local storage is not persistent, so that other infrastructure storage component such as Amazon Elastic Block Storage (EBS) in AWS and Persistent Disk PD in GCE are required. As a result, the root device volume of an instance is Amazon EBS/PD volume. Also, an instance hot data should store in Amazon EBS/Amazon S3/GCE PD when the instance dies a customer can replace it by attaching the volume to a new instance (Varia & Mathew, 2014).

Checkpointing "snapshot" strategy is Amazon EBS and GCE redundantly feature that reduces the possibility of failure throughout automatically take an image of the instance volume. Replication feature is implemented in Amazon EBS, EBS volume data can be replicated with ease across multiple servers in a region AZs. Automatically, Amazon EBS snapshots are reserved in Amazon S3. While GCE snapshot is a global resource, this mean PD snapshot can be applied to new PD volume in the same zone, different zone, or different region. Many VM instances can be connected to one PD volume with read-only mode (Amazon Web Services, 2016; Google, 2016a).

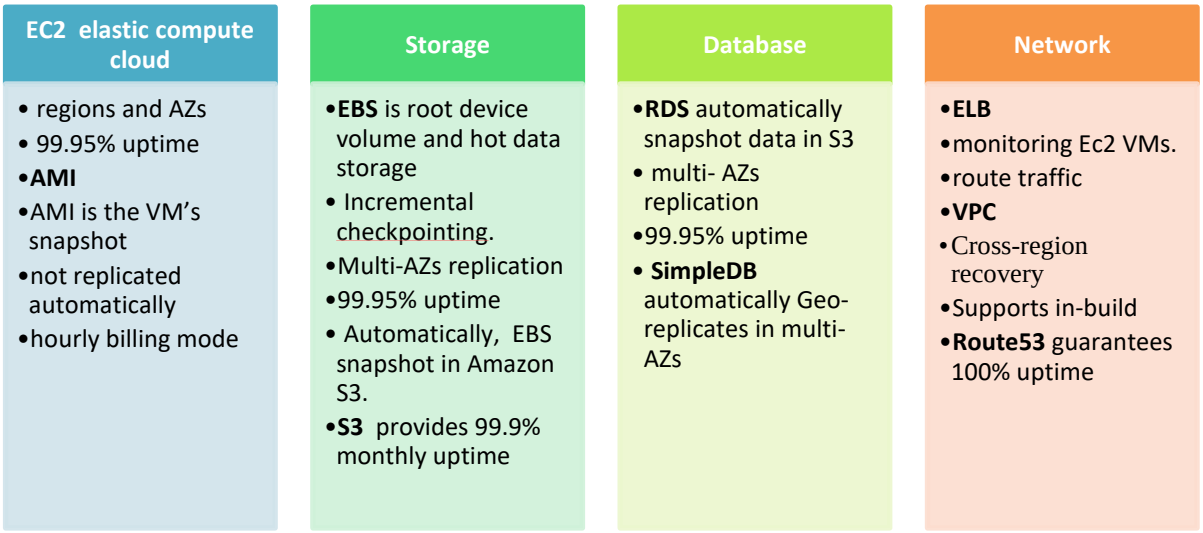
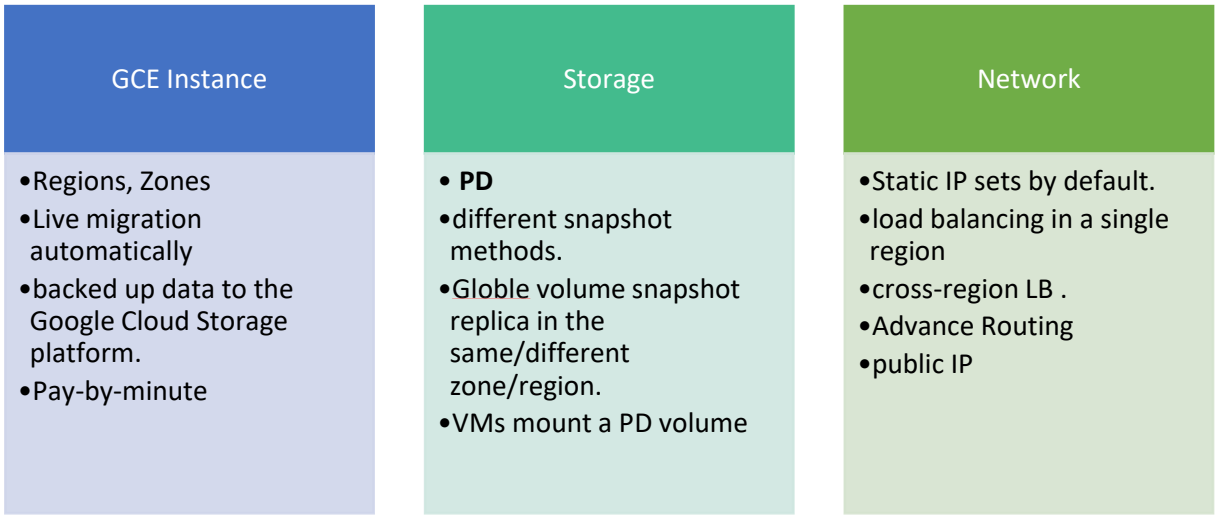


Fig. 2. Amazon Web Service (AWS) Infrastructure Building Block

Amazon Relational Database Service (RDS) is an infrastructure component that provides fully featured virtual database server on the cloud with the capability of online access and use customers' databases such as MySQL, Oracle, SQL server, or PostgreSQL). Amazon RDS automatically snapshot data which are stored on Amazon S3 for more protection and availability. A customer's database can be replicated in multi-availability Zone AZs within the same region. There is no direct service to replicate to another region. Moreover, Amazon SimbleDB is a non-relational database functionality that creates, stores, and queries varied sets of data, with automatic administration infrastructure, hardware, software, etc. Geo-replication of data among multiple Availability Zones AZs within the same region is done automatically (Baron & Kotecha, 2013). Whilst, Google database services are deployed over Google Cloud Platform and google developer are working for launching database services within their IaaS.

Finally, networking features apply within Amazon and Google IaaS, Amazon Elastic Load Balancing (ELB) could distribute concurrently arriving traffic from users to multi-EC2 instances. Better fault tolerance architecture in AWS can be achieved throughout ELB. Amazon ELB route traffic across different Availability Zones within specific region. Amazon Virtual Private Cloud (VPC) "Virtual Datacenter" supports launching AWS resources in a virtual network topology, which is constructed by customers with entire governing over this network (IP address ranges, subnets, route table configuration, ports, VLANs, and gateway). As a result, cross-region recovery of customer's application is provided. Amazon Route53 is the cloud base DNS server, which automatically answers users request to cloud EC2 instances, ELB, Amazon S3, or routes users to infrastructure outside AWSs with Geo DNS, strong fault tolerance architecture can be configured, the healthy-check services which globally monitor and manage traffic to the just healthy and reachable resources by returning the IP address of healthy resources (Amazon Web Services, 2014; Varia & Mathew, 2014). Advanced Networking Feature launched within Google IaaS. Per-region static IP address for an instance is specified by default as well as reserved static IP feature so when a zone fails, it is easy to route traffic to another zone.

Fig. 3. Google Compute Engine (GCE) Infrastructure Building Block



Moreover, Global resource (public IP) determines how VM instances communicate with each other, with other network and with the outside world as well. Service-side load balancing technology (Network load balancing in a single region, HTTP such as cross-region LB and content-based LB) support heavy traffic, detect unhealthy VM instances, and route request to the closet VM. Advance Routing is a new feature that helps connect in-premise datacenter and the cloud, build multi-cloud deployment, and VPN (Google, 2016b).

Comparing these features make sense which provider covers a customer requirement. It will be interesting to see that Amazon AWS and Google Compute Engine are similar in their main infrastructure as service components. AWS and GCE try to offer high-quality and high-

DOI: 10.5824/1309-1581.2016.4.001.x

availability service, resilient infrastructure, redundant infrastructure, easy access to resources, independent storage, create VMs snapshot, and high-level of network options, at least offer 99.95% monthly uptime as well. Note, however, that they use common fault tolerance strategies which are checkpointing and replication strategies to survive customers' applications and enterprises. Some differences are in their efficiency of infrastructure resources, for example (Google PD volume has the capability to be attached to multiple VM instances with read-only mode, billing scheme mode is applied in GCE versus hourly billing scheme within AWS, live migration of VM instances in case of a datacenter maintenance is done automatically in Google CE, AWS datacenters spreads across eleven regions, while GCE has three regions

Clearly, failure happens all the time and the cloud vendors design geo-location to tolerate regions' failure, more facilities help surviving AZs, network problem can be solved by load balancing technique, IPs, DNS routing, and VPNs, and deploy multiple instances overcome instance failure. Therefore, the critical key point to success deploying services/enterprises over the cloud IaaS is that how can customers design and deploy effective architecture that faults mitigate and avoid real risk for their businesses. Accordingly, in the following section, general conceptual steps are proposed to help designing and architecting a solution with fault tolerance consideration for customers' applications over the cloud infrastructure.

Fault tolerance consideration for IaaS cloud computing application

In reality, few works concentrate on a good understanding to build and setup cloud strategy of a solution in the cloud computing environment. In the cloud context, design a solution strategy is a complex decision- making process which needs strategic and well-ordered methods and consideration to be taken into account.

Build fault tolerance strategy within a workload definition

The workload is a key process for understanding service/enterprise architecture and describing exactly what will be deployed over the cloud virtual infrastructure. According to NetApp research (Villatore-Silva, 2012), a successful solution needs to address the real application's workload elements, requirements, technical sides, and metrics such as:

- Availability,
- Design solution costs such as resource cost.
- Security, regulatory, and privacy demands.
- Capacity,
- Infrastructure resources are the essential requirement to run the enterprise in pay-as-you-go method.
- Financial consideration affects workload, such as the IT capital budget.

- Business services such as Enterprise Resource Planning (ERP), finance and accounting, Customer Relationship Management (CRM), Human Resources (HR), Payroll, as well as Project Management System.

- Data structure as SQL or NoSQL.

- Fault tolerance strategy setup. This characteristic depends on the type of the enterprise; whether it is a traditional solution or a new generation solution.

In essence, the workload should be built on the idea that a web application failure happens all the time, thus enterprise developers should think about how can they choose the convenient methods that fit the workload as well as they should determine the workload type is it for a client-server/traditional application or a new generation application (gaming, mobile app, HPC, social app). Moreover, a cloud application has other sub-workloads, for example, a web site may have search and browse workload, and registration workload.

Build up life-cycle development model

Clearly, after establishing an application workload operational side should be designed within a specific period of time. According to Marks and Lozano research in their book (Marks & Lozano, 2010) life-cycle describes the process of plan activity in a systematic manner throughout establish the milestones in which key scope decision are made for start and end point of the system, team responsibilities, gives direction, and identifying the availability of resource demands. As a result, the life-cycle includes tracking expected behavior of an application at different time and life-cycle stages such as planning, analysis of the system, system design, implementation, testing, and maintenance as well. Next step is to draw charts (weekly, annually) for a time against usage, availability, or capacity by taking entities, aspects, and conditions in their consideration. In this way, recognizing the life cycle will help determine the next step.

Identify availability plan

The major important part is to assess availability plan in business infrastructure. What is the level of availability demanded over the life cycle of the workload should be defined? In terms of the SLA, studying carefully the cloud provider's SLA and identifies the applicable SLA which is needed. Which resource requires 99.9999% or 99.95% uptime because 99.9999% availability requires high-cost investment.

Recognize and identify failure mode

The availability of workload is affected by the failure. Therefore, understanding, identifying, and documenting failure will precisely insure building the map of failure points.

Understand the architecture patterns

Establishing robust architecture help determining strategy for high availability so as to deliver a health application throughout many facets:

- Communication between and within different workload parts result in handle and manage failures.

DOI: 10.5824/1309-1581.2016.4.001.x

- Identify critical resources expose to fail, which components need to replicate, snapshot, change/modify software, improve, or upgrade such as OSs.

- Trust cloud services that are provided by third parties.

- Automated everything is valuable, so as to minimize the developers' mistake which minimize business risk, improve efficiency.

- Implement right redundancy strategies. Several scenarios are clearly architected by Amazon Web Services and Google Compute Engine. Either provide full redundancy solution, however, it increases 100% of the cost, or partial redundancy in another region (cost-effective) solution.

- Traffic management is valuable to ensure that a request is directed to the appropriate node.

Related works

In (Das & Khilar, 2013), authors discussed VFT, a model which addresses fault tolerance in a hypervisor virtualization architecture inside a cloud datacenter. The VFT has the capability of decreasing the service time, also ensuring and raising the system availability. The Reactive fault tolerance mechanism with the replication and redundancy techniques is used to build the model. The models' scheme implicates two stages. Cloud manager (CM) stage carries out hypervisor virtualization, load balancing, and performance recording (success rate parameter), then detect and repair faults throughout fault handler. The second stage is the decision maker (DM) which responsible for checking the nodes status and task deadline, then the algorithm will give the final decision to create checkpoint if the all checking gives correct result. The proposed model was evaluated throughout the analyzing of the success rate SR metric; whenever, SR is increased the scheme will achieve perfect performance.

In another work (Gómeza, Carril, Valin, Mouriño, & Cotelo, 2014), Gomeza and colleagues suggested virtual cluster architecture to tolerate faults in the cloud IaaS platform. Cloud sites failure can be recovered through deploying of virtual clusters on two different geographical locations. It is a significant method in high performance computing (HPC) applications, in case the whole or part of the site fails. This architecture is suitable for executing a particular application of a customer. It is totally independent, and can be deployed in one or several cloud providers. Another technique is added which monitoring the application performance periodically.

There are works on storage architecture for cloud reported to the literature. Examples include Magicube (Feng, Han, Gao, & Meng, 2012), a cloud storage architecture model, which has the ability to reduce the space cost of redundancy, improve performance and guarantee the reliability of the system. Unlike Amazon S3, Hadoop Distributed File System (HDFS), and Google GFS storage system that needs three replications of each file as a default, Magicube system is conserved just 1 replica of a file.

In (Egwutuoha, Chen, Levy, & Selic, 2012), researchers defined a new cloud computing capability of employing fault tolerance for computation intensive application which needs a lot of computations and acceleration to perform the task such as medical imaging, bioscience, or financial trading. A framework of a high performance computing system which depends on a proactive FT strategy to predict the fault, reactive FT uses checkpointing to reduce the cost of execution time, live migration and FT protocol for communication as well.

Fault Tolerance as a Service (Nandi, Paul, Banerjee, & Ghosh, 2013) is another service proposed as a formal model. Abbreviated as FTaaS, it presents a service by a cloud vendor to provide fault tolerance guarantees to customers' applications. FTaaS functions as an agent to the lowest level of service in cloud computing, which is IaaS. It is provided as a part of SLA with two fault tolerance strategies: spatial redundancy which depends on majority-voting and temporal redundancy, which is based on checkpointing mechanism.

Additionally, in (Cully et al., 2008) researchers report their work on virtual machine replication, Remus. They propose this model to provide OSs high-availability as a service which are based on virtualization infrastructure platform. Virtualization guaranties the capability of creating a copy of a running machine as well as migrates running VMs to other hosts. Remus offers protection similar or better than commercial providers' cost throughout replicates snapshot of the entire running operating system approximately every 25ms to another location.

Conclusion and future work

In conclusion, cloud computing provides many features to small and medium business enterprises such as cost-effective of infrastructure resources, managing infrastructure, availability, and scalability. From a technical point of view, cloud services are commercial and their services are not designed to ensure the continuity of customers' application. In fact, they ensure the availability of their infrastructure and components which are offered to customers. As a result, fault tolerance property should be addressed so as to guarantee customer system continuity over the cloud. The material put in the background section and the discussion of general conceptual steps show a pre-define strategy in which how an enterprise can be appointed into a workload so as to specify getting availability and failover. Accordingly, a life-cycle model and milestone could be established. After this step, specifying the nature of availability plan and establishing the map of expected failures are needed. All these key points should be considered to design a highly available and resilient architecture, where the customer could achieve a better fault tolerance plan. Section on fault tolerance consideration discusses in detail the key lessons learned.

The future research path in the direction of fault tolerance will introduce possible evaluation and analyses the cost of implementing FT within Amazon Web Service AWS and Google Compute Engine on-demand infrastructure.

DOI: 10.5824/1309-1581.2016.4.001.x

References

1. Kepes, B. (2011). Revolution Not Evolution How Cloud Computing Differs from Traditional IT and Why it Matters - White paper: Diversity Limited.

2. Carlin, S., & Curran, K. (2012). Cloud Computing Technologies. International Journal of Cloud Computing and Services Science (IJ-CLOSER), 1(2), 59-65.

3. Latchoumy, P., & Khader, P. S. A. (2011). Survey on Fault Tolerance in Grid Computing. International Journal of Computer Science & Engineering Survey (IJCSES), 2(4), 97-110.

4. Ganga, K., Karthik, S., & Paul, A. C. (2012). A Survey on Fault Tolerance in Work flow Management and Scheduling. International Journal of Advanced Research in Computer Engineering & Technology (IJARCET), 1(8), 176-179.

5. Pullum, L. L. (2001). Software Fault Tolerance Techniques and Implementation: Artech House.

6. Dubrova, E. (2013). Fault Tolerant Design: Springer Science+Business Media.

7. Selic, B. (2006). Fault Tolerance Techniques for Distributed Systems: IBM.

8. Patel, A., Taghavi, M., Bakhtiyari, K., & Junior, J. C. (2013). An intrusion detection and prevention system in cloud computing: A systematic review. Journal of Network and Computer Applications, 36, 25-41.

9. Dave, S., & Raghuvanshi, A. (2012). Fault Tolerance Techniques in Distributed System. International Journal of Engineering Innovation & Research, 1(2), 124-130.

10. Avizienis, A., Laprie, J.-C., Randell, B., & Landwehr, C. (2004). Basic Concepts and Taxonomy of Dependable and Secure Computing. IEEE Transactions on Dependable and Secure Computing, 1(1), 11-33.

11. Laprie, J.-C. (1995). Dependable Computing Concepts and Fault Tolerance: Terminology. Paper presented at the 25th International Symposium on Fault-Tolerant Computing, Highlights from Twenty-Five Years, Pasadena, California

12. Lussier, B., Chatila, R., Guiochet, J., Ingrand, F., Lampe, A., Killijian, M.-o., & Powell, D. (2005). Fault Tolerance in Autonomous Systems: How and How Much? Paper presented at the 4th IARP/IEEE-RAS/EURON Joint Workshop on Technical Challenge for Dependable Robots in Human Environments, Nagoya, Japan.

13. VMware. (2009). Protecting Mission-Critical Workloads with VMware Fault Tolerance - White Paper: VMware.

14. Li, H., Shang, L., Dang, J., & Jin, H. (2009). Fault Recovery Approach in Fault-Tolerant Processor. Paper presented at the International Conference on Scalable Computing and Communications; 8th International Conference on Embedded Computing, Dalian, China.

15. Runge, A. (2012). Reliability Enhancement of Fault-prone Many-core Systems Combining Spatial and Temporal Redundancy. Paper presented at the 14th International Conference on High Performance Computing and Communications, Liverpool, UK.

16. Babcock, C. (2010). Management Strategies for The Cloud Revolution: McGraw-Hill.

17. Li, Y., & Lan, Z. (2011). FREM: A Fast Restart Mechanism for General Checkpoint/Restart. IEEE Transactions on Computers, 60(5), 639-652.

18. Gokuldev, S., & Valarmathi, M. (2013). Fault Tolerant System for Computational and Service Grid. International Journal of Engineering and Innovative Technology (IJEIT), 2(10), 236-240.

19. Garg, R., & Singh, A. K. (2011). Fault Tolerance in Grid Computing: State of The Art and Open Issues. International Journal of Computer Science & Engineering Survey (IJCSES), 2(1), 88-97.

20. Sun, D., Chang, G., Miao, C., & Wang, X. (2013). Analyzing, modeling and evaluating dynamic adaptive fault tolerance strategies in cloud computing environments. The Journal of Supercomputing, 66(1), 193–228.

21. Ghoreyshi, S. M. (2013). Energy-Efficient Resource Management of Cloud Datacenters Under Fault Tolerance Constraints. Paper presented at the International Green Computing Conference (IGCC), Arlington, Virginia.

22. Amazon Web Services. (2016). Amazon Elastic Compute Cloud: User Guide for Windows Instances: Amazon Web Services.

23. Ferraioli, J. (2014). Blurring the IaaS PaaS Divide with Julia Ferraioli [Press release]. Retrieved from https://www.youtube.com/watch?v=tmhGlaXuIn8&list=PLXI5ri9BGtlG12MaiClkARwv_p6n7xJn6

24. Varia, J., & Mathew, S. (2014). Overview of Amazon Web Services (pp. 22): AWS.

25. Google. (2016a, 4.October.2016). Creating Persistent Disk Snapshots. Retrieved 16.Nov.2016, 2016, from https://cloud.google.com/compute/docs/disks/create-snapshots

26. Baron, J., & Kotecha, S. (2013). Storage Options in the AWS Cloud - White Paper: AWS.

27. Amazon Web Services. (2014). Elastic Load Balancing Developer Guide API Version 2012-06-01: AWS.

28. Google. (2016b, 8.November.2016). Using Networks and Firewalls. Retrieved 14.Nov.2016, 2016, from https://cloud.google.com/compute/docs/networking

29. Villatore-Silva, T. (2012). Creating an Enterprise-Wide Cloud Strategy - White Paper: NetApp.

30. Marks, E. A., & Lozano, B. (2010). Executive's Guide to Cloud Computing: Wiley.

31. Das, P., & Khilar, P. M. (2013). VFT: A Virtualization and Fault Tolerance Approach for Cloud Computing. Paper presented at the Conference on Information and Communication Technologies (ICT 2013).

32. Gómeza, A., Carril, L. M., Valin, R., Mouriño, J. C., & Cotelo, C. (2014). Fault-tolerant virtual cluster experiments on federated sites using BonFIRE. Future Generation Computer Systems, 34, 17-25.

DOI: 10.5824/1309-1581.2016.4.001.x

33. Feng, Q., Han, J., Gao, Y., & Meng, D. (2012). Magicube: High Reliability and Low Redundancy Storage Architecture for Cloud Computing. Paper presented at the 7th International Conference on Networking, Architecture, and Storage, Xiamen, Fujian, China.

34. Egwutuoha, I. P., Chen, S., Levy, D., & Selic, B. (2012). A Fault Tolerance Framework for High Performance Computing in Cloud. Paper presented at the 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, Ottawa, Canada.

35. Nandi, B. B., Paul, H. S., Banerjee, A., & Ghosh, S. C. (2013). Fault Tolerance as a Service. Paper presented at the 6th International Conference on Cloud Computing, Santa Clara, California.

36. Cully, B., Lefebvre, G., Meyer, D., Feeley, M., Hutchinson, N., & Warfield, A. (2008). Remus: High Availability via Asynchronous Virtual Machine Replication. Paper presented at the NSDI '08: 5th USENIX Symposium on Networked Systems Design and Implementation, San Francisco, CA.