Technical University of Denmark

02228 Fault-Tolerant Systems

Fault-Tolerant Cloud Computing Architectures

Authors:

Andreas Hallberg Kjeldsen
s092638@student.dtu.dk

Morten Chabert Eskesen
s133304@student.dtu.dk

December 9, 2014

Contents

List of Figures

Chapter 1

Introduction

In this report we will describe what cloud computing is, further we will give a detailed description of the architecture and fault-tolerant features of two cloud system, at last we will compare how the systems handle failures and discuss the pros and cons of these methods. As a result of the comparison, we will be able to conclude on what the systems do well and where they might be able to improve.

1.1 Scope

We will focus on the fault-tolerant features of the cloud computing architecture within the two selected cloud computing systems. We have chosen to focus on Amazon Web Services and Google Cloud Platform. We have chosen these cloud computing systems because both systems are among the most popular[1] cloud computing systems [16].

1.2 Cloud Computing

The National Institute of Standards and Technology is a federal technology agency in the United States of America. They define cloud computing by the following:

"Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort or service provider interaction." [17]

This definition states that shared networks, servers, applications, services etc can easily be distributed globally and quickly maintained by using cloud computing.

There are five essential characteristics of the cloud computing model

[1] Popular means that they are among the most commonly used platforms for enterprise cloud developers.

On-demand self-service No required human interaction when needing more or less computing capabilities

Broad network access Capabilities are accessed through standard mechanisms and available over the network

Resource pooling Computing resources are pooled in order to serve multiple consumers

Rapid elasticity Capabilities can be elastically released to scale rapidly according to the demand

Measured service In an automatic way the cloud computing systems control and optimize resource use

There are four different deployment models. One cloud infrastructure is for exclusive use by a single organization comprising the multiple consumers - the *private cloud*. Another cloud infrastructure is for exclusive use by a specific community of users from organizations with shared concerns - the *community cloud*. The *public cloud* is an infrastructure open for use by the general public. The last cloud infrastructure is a mixture of two or more distinct cloud infrastructures that remain unique entities - the *hybrid cloud*.

1.3 Fault tolerance in cloud computing

Fault tolerance is a key factor for cloud computing systems due to the rapid exponential growth in use of cloud computing [18]. The purpose of fault tolerance in any system is to achieve robustness and dependability. Fault tolerance policies and techniques allow us to classify this techniques into 2 types

Proactive fault tolerance policy aims to avoid recovering from fault, errors and failure by predicting them and replacing the suspicious component. This means detecting problems before they actually occur.

Reactive fault tolerance policy reduces the effect of failures when the failure actually occurs.

These policies can be divided into two further sub techniques error processing and fault treatment. The aim of error processing is to remove errors from the computational state and the aim of fault treatment is to prevent faults from reoccurring.

Chapter 2

Amazon Web Services

The Amazon Web Services, henceforth *AWS*, is a collection of remote computing services which make up a cloud computing platform. AWS was launched by Amazon.com in 2006. This chapter will outline the architecture of their platform and describe the fault-tolerant features the platform provides.

2.1 Architecture

AWS offers a wide range of services, which can be seen as components of a system. The services they provide are available in multiple continents. The services can be used to build entire systems which are highly fault-tolerant. Using AWS the hassle of obtaining servers in various locations is removed, instead the user just has to select which region to deploy to. Though it might be tempting to just push an entire system out into the cloud, it is not enough to make it fault tolerant. The architecture of the system is important. Below is example architectures for a website and an application.

2.1.1 Website architecture

In Figure 2.1 is a system with two availability zones, each hosting the web server, application server and a database server. Replication is setup between the databases. Dependencies between the availability zones has been avoided. The load balancing makes sure to distribute the traffic properly. If one of the web servers or application servers fail, then the load balancer could redirect all traffic to availability zone without errors. While the traffic is redirected, the failed instance will be replaced by instantiating a new instance from a preconfigured image.

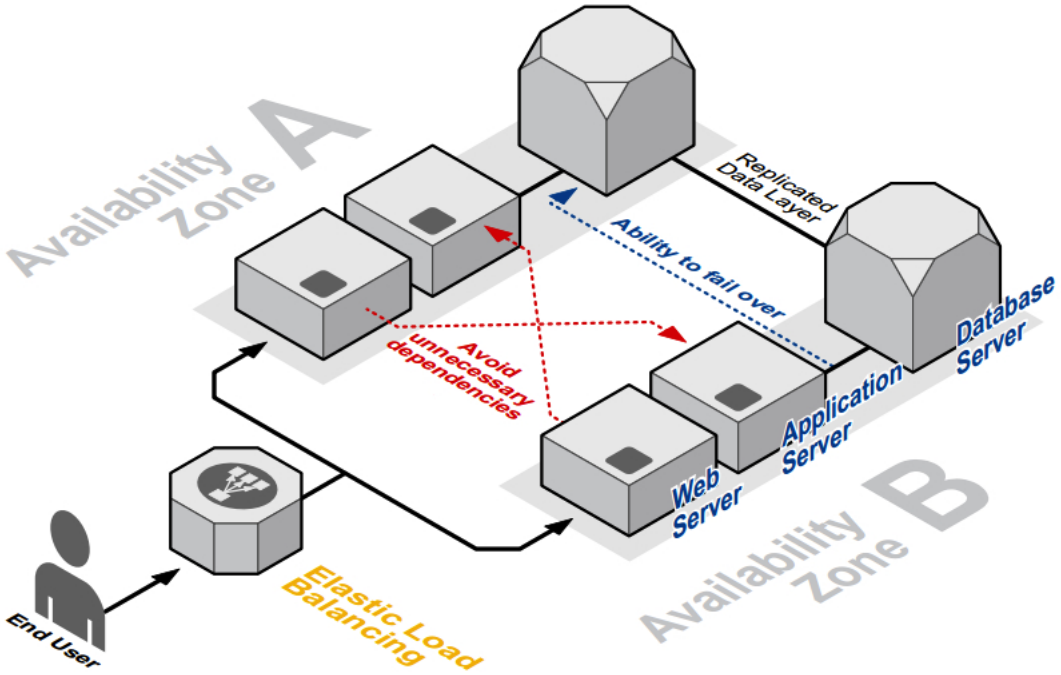

Figure 2.1: Example of fault-tolerant website architecture hosted on AWS[1].

2.1.2 Application architecture

In Figure 2.2 is a system with two availability zones, each hosting an application instance and a storage instance. At specified time intervals, snapshots of the storage instances are saved to another storage system, which also has fault-tolerance implemented. In the event of an availability zone becoming unavailable, the Elastic IP can easily be mapped to use another availability zone. In the event of an application instance crashing, the instance can easily be replaced using a preconfigured images of the instance. When the application instance has been replaced, the storage instance used is re-attached. If the storage instance fails, a new one can be instantiated from the saved snapshots.

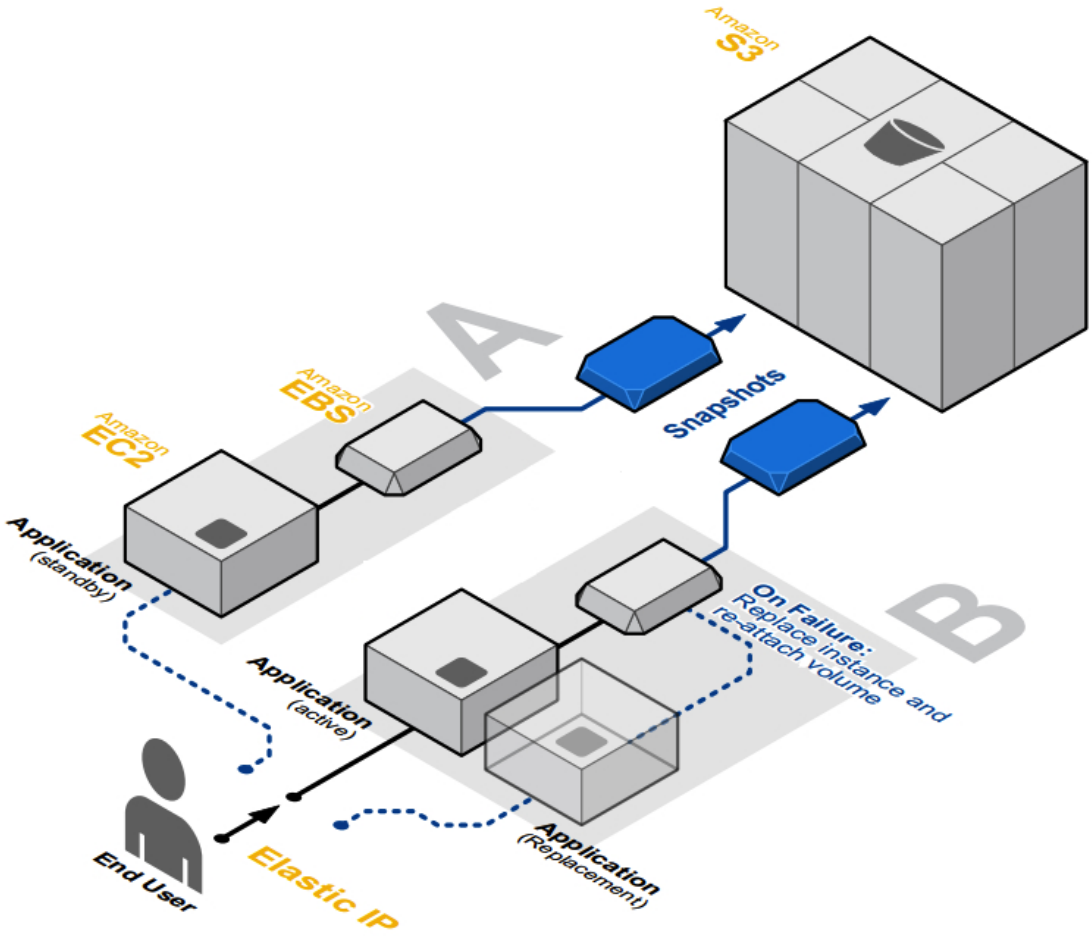

Figure 2.2: Example of fault-tolerant application architecture hosted on AWS [1].

2.2 Availability

AWS have regions in various parts of the world. Each region has multiple Availability Zones, henceforth *AZ*. When using AWS, the user has to specify in which region they wish to have their resource[1] located. Within the region, a resource may then be distributed to multiple AZ.

[1] A resource can be a computing resource, storage resource etc.

2.3 Redundant storage

AWS provides multiple ways of storing data. They have key/value based storage (S3), block storage (EBS) and physical SSDs.

2.3.1 Simple Storage Service

The Amazon Simple Storage Service, henceforth *S3*, is a highly-scalable, reliable and low-latency key/value data storage infrastructure. With S3 you define buckets in which you wish to store data. Each bucket range from 1 TB to 5TB in capacity and has a maximum file size of 5 GB. Within a region, the data stored in S3 is replicated to various AZ, hence redundancy is implemented. Storing the data within various AZ, allows S3 to fetch the data requested from whichever AZ that contains the requested data with the lowest latency. Though the data is available from multiple AZ in the region, it can be setup to also be replicated to different regions, which in turn replicates the data to the various AZ in the region.

2.3.2 Elastic Block Storage

The Amazon Elastic Block Storage, henceforth *EBS*, is block storage for use with their Elastic Compute Cloud. The blocks can be used just like an ordinary hard disk. There EBS is backed by either SSD disks or magnetic disks. It is recommended to use the SSD backed EBS for services and applications and only use the magnetic disks for data that is rarely accessed. EBS can be run in four different RAID[2] configurations. The configurations are RAID0 (striped disks), RAID1 (mirroring), RAID5 (striping with distributed parity) and RAID6 (striping with double distributed parity). Amazon recommend RAID0 for use when I/O performance is more important than fault-tolerance, RAID1 when fault-tolerance is more important than I/O performance and they discourage the use of RAID5 and RAID6 [7]. Snapshots are created of the EBS volumes and stored on S3, thus the EBS volumes can be recreated at any time.

2.3.3 Physical SSDs

The Amazon Elastic Compute Cloud, henceforth *EC2*, are computing instances. The EC2 instances comes in various versions ranging from small instances with limited capabilities to large extravagant instances that are fully equipped. The smaller instances have no hard disk but are using EBS instead. The larger instances can both use EBS and hard disks. The hard disks used are SSDs and if more than one is attached to the instance, then they can be run in either RAID0 or RAID1. Some instances are optimized for EBS, which means that the overhead of using EBS instead of an attached SSD is very limited. All instances

[2]RAID stands for Redundant Array of Independent Disks.

are able to access EBS, this is due to the hard disks attached to a computing instance are formatted when the instance is terminated, thus important long-term data should be stored on EBS [8].

2.4 Application Recovery

At AWS, the computing instances are started by launching an Amazon Machine Image, henceforth *AMI*. The AMIs are ready to deploy templates of the root volume (for example an operating system, an application server and applications), launch permissions and mapping to EBS volumes. Using AMIs it is easy to launch a new instance when server load is high or quickly recover from instance failures. The AMIs are stored on S3.

2.5 Elastic Load Balancing

The Amazon Elastic Load Balancing, is a service for distributing the load between running EC2 instances. Further it also allows for greater levels of fault-tolerance by routing traffic away from faulty EC2 instances. Elastic Load Balancing can be setup to automatically detect if EC2 instances are performing poorly or if they are dead, in this case the EC2 instance will no longer receive incoming traffic. While the traffic is being routed to other EC2 instances, the faulty EC2 instance will be replaced by a newly launched EC2 instance or a reserved EC2 instance. When the new EC2 instance is up and running, the Elastic Load Balancer can start routing traffic to it. If setup properly, the detection and replacement of faulty EC2 instances can be done automatically without the need of any manually interaction with the AWS. Elastic Load Balancing also allows for auto scaling, meaning if more computing power is required, a new EC2 instance can be launched and then the load balancer will start to even out the traffic. An example use for Elastic Load Balancing with auto scaling is shown in Figure 2.3. The example show two load balancers, where the first one spreads the traffic between the EC2 instances running the web server, the next load balancer spreads the traffic between the EC2 instances running the application serving. Using this setup, it's possible to scale the EC2 instances asymmetrically, this is useful when only the EC2 instances running the web server is in need of more resources etc.

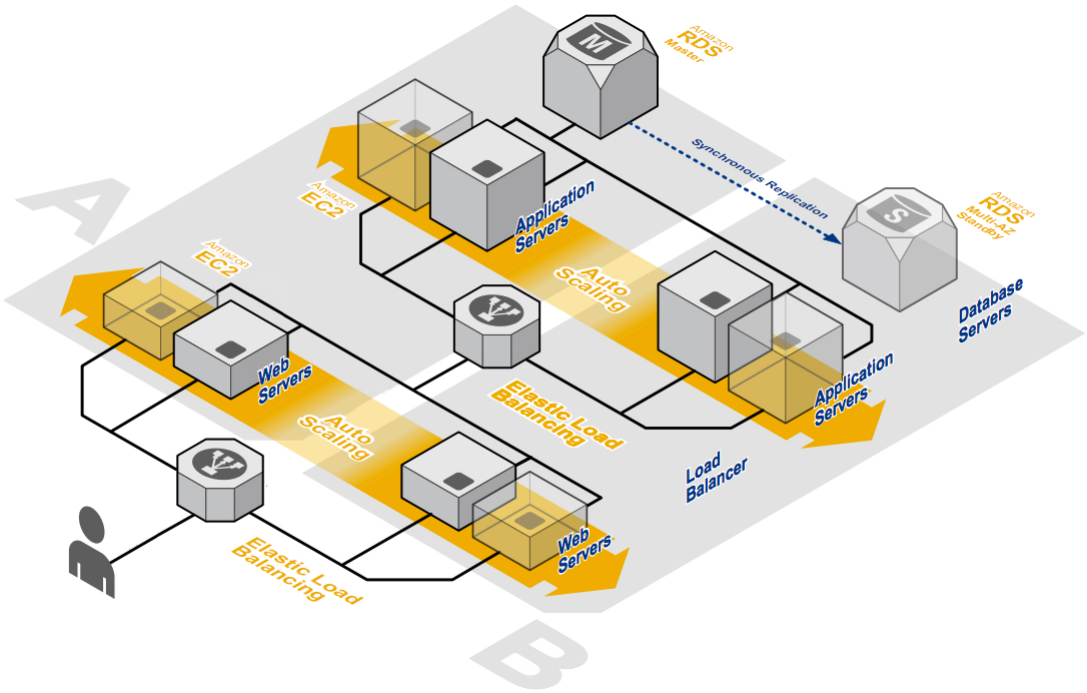

Figure 2.3: Example of using AWS Elastic Load Balancing for fault-tolerance and auto scaling.

2.6 Elastic IP

Amazon offers Elastic IP, which is a publicly accessible IP that is linked with an EC2 instance. When an EC2 instance is launched, it will get an IP assigned to it. There are no guarantee that an EC2 instance will be assigned the same IP if it has been stopped and then started again. To circumvent this problem, the Elastic IP can be used. The Elastic IP is static, hence other applications or end users will only have to access the Elastic IP and not worrying about being routed to a destination. The Elastic IP can also be used in case of an instance failure. When an instance fails, a new instance can be launched thereafter the Elastic IP is linked with the new instance. Same procedure can be applied if an availability zone becomes unavailable.

Chapter 3

Google Cloud Platform

Google Cloud Platform, henceforth *GCP*, is collection of cloud computing products made by Google. This chapter will outline the architecture of GCP and describe the fault-tolerant features the platform provides.

3.1 Architecture

Google App Engine, henceforth *GAE*, is a service for developing and hosting web applications in Google-managed data centers. In Figure 3.1 the architecture of a web application on GAE is depicted. The Google Load Balancer makes sure to distribute the traffic properly. The load balancer is a built-in feature of GAE that provides transparent Layer 3 and Layer 7 load balancing to applications. GAE has an automatic scaling feature. When the number of request increases for an application GAE will automatically allocate more resources such that the web application is able to handle the additional demand.The Memcache and Task Queue services are integrated within GAE. The Memcache is an in-memory cache shared across all of the GAE instances. This means that there is extremely high speed access to information cached by the web-server. Task Queue provides a mechanism to free the front end servers to service new user requests by offloading longer running tasks to backend servers. Google Cloud Storage is an online storage service for files, i.e. the static content. Meanwhile the dynamic content from databases is handled by the Google Cloud SQL and Google Cloud Datastore. Google Cloud SQL is a fully managed MySQL database that lives in the Google Cloud infrastructure. Google Cloud Datastore is a fully managed, highly available NoSQL data storage for non-relational data that includes a REST API.

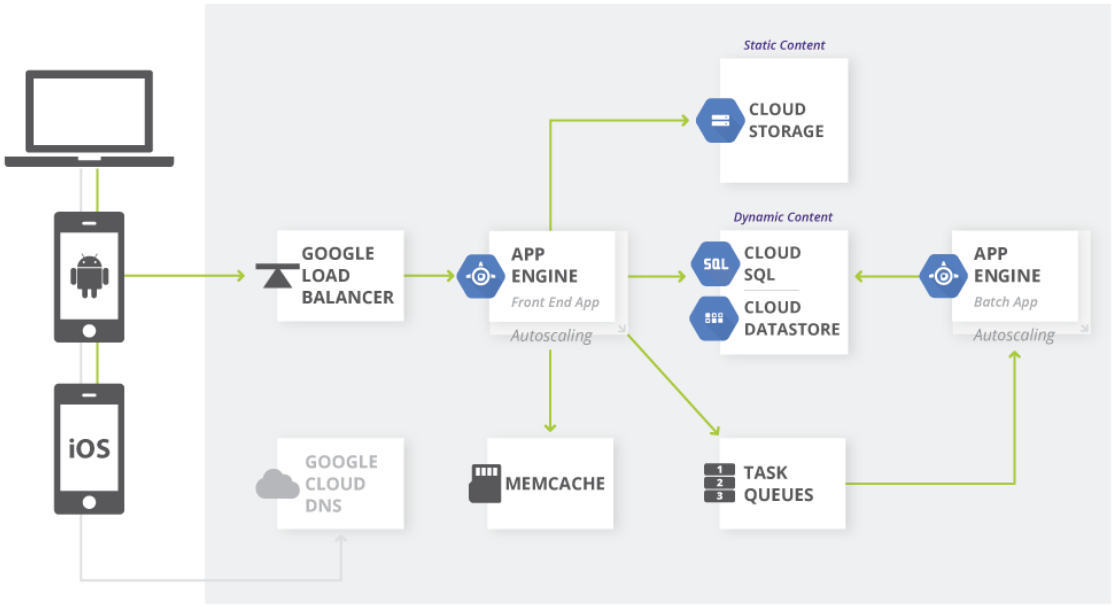

Figure 3.1: Google Web Application Architecture on Google App Engine [11].

3.2 Availability

GCP has regions in various parts of the world. Each region has multiple availability zones. When using GCP the customer has to specify in which region their resource should be located. Only certain resources are region- and zone-specific. Resources that are specific to a zone or a region can only be used by other resources in the same zone or region. Disks and instances are examples of zonal resources. Other resources are global resources that can be used by any other resource across any location. An example of a global resource is an image. In order to build fault-tolerant applications that have high availability the recommendation from Google is to deploy applications across multiple zones in a region. The goal of this is to protect against unexpected failures of components, up to and including a single zone. When putting resources in different zones in a region it provides isolation for many types of infrastructure, hardware and software failures. Even higher degree of failure independence is achieved by putting resources in different regions.

3.3 Redundant storage

GCP provides multiple ways of storing data. They have storage for files (CS), NoSQL data storage (CD) and MySQL database (CSQL). These ways of storing data will explained in this section.

3.3.1 Cloud Storage

The Google Cloud Storage, henceforth *CS*, is an online file storage web service for storing and accessing your data on Google's infrastructure. All data in CS belongs inside a project. Project consists of a set of users, a set of APIs, billing, authentication and monitoring settings for those APIs. Buckets are the basic containers that holds the data. Everything stored in CS must be contained in a bucket. Buckets belong to a project. Objects are the individual pieces of data stored in CS. CS provides a set of features and capabilities which are designed to store, share and manage data in an efficient and reliable way.

High capability and Scalability CS is able to store objects that are terabytes in size.

Strong Data Consistency CS provides strong read-after-write consistency for all upload and delete operations. In availability terms upload operations to CS are atomic, i.e. when uploading an object it is not accessible until it is completely uploaded.

Bucket Locations CS provides users with the ability to specify where the buckets are geographically stored.

REST APIs CS provides two RESTful programming interfaces.

3.3.2 Cloud Datastore

The Google Cloud Datastore, henceforth *CD*, is a schemaless NoSQL datastore providing robust, scalable storage for applications. CD has the following features

- No planned downtime
- Atomic transactions
- High availability of reads and writes
- Strong consistency for reads and ancestor queries
- Eventual consistency for all other queries

CD replicates data across many datacenters using a system which is based on Paxos algorithm. Paxos is a family of protocols for solving consensus in a network of unreliable

processors. Consensus is the process of agreeing on one result among a group of participants. Google has used the algorithm to build a fault-tolerant database, i.e. CD. CD can execute multiple operations in a single transaction. A transaction can not succeed unless every operation in the transaction succeeds. If one of the operations fail the transaction is automatically rolled back.

3.3.3 Cloud SQL

The Google Cloud SQL, henceforth *CSQL*, is a MySQL database that lives in Google's cloud. It has all the capability and functionality of MySQL with a few additional features and a few unsupported features. According to Google, CSQL is ideal to use in a small to medium-sized application. CSQL offers fully managed, highly available relational databases. It is possible to create, configure and use MySQL databases that live in Google's cloud. Google handles the replication, encryption, patch management and backups. All the data is encrypted when stored and in flight on Google's network. Only authorized IP addresses are accepted when connecting to instances and can be secured with SSL. All the data is replicated in multiple locations to achieve durability and high availability.

3.4 Load balancing

Load balancing is a service for distributing the load across multiple virtual machine instances. Google Compute Engine, henceforth *GCE*, is the service that allows users to launch virtual machine instances. The load balancing can be done with either network load balancing or HTTP load balancing. The load balancing provides the following benefits:

- Scaling of application
- Supporting heavy traffic
- Detection of faulty virtual machines
- Balance load across regions
- Route traffic to closest virtual machine
- Support content-based routing

This means that faulty virtual machines will not cause downtime of an application as the traffic can be routed to other healthy running virtual machines or a new virtual machine can be launched to replace the faulty one. The scaling of application is done by launching new virtual machines if in need of computing power due to heavy traffic. When launching new virtual machines the load balancer will distribute the load in a way such that the traffic for each virtual machine is even.

Chapter 4

Comparison of Failure Handling

List of faults that the systems handle along with a description of how it's handled and why it works. If the methods for handling the failure differ, we will discuss the methods, highlighting their pros and cons.

Chapter 5

Conclusion

Conclude on our findings, focus on what the systems do well and where it might be possible to improve.

Bibliography

[1] Amazon Web Services Reference Architectures, *Fault Tolerance & High Availability*, viewed October 2014. http://media.amazonwebservices.com/architecturecenter/AWS_ac_ra_ftha_04.pdf

[2] Amazon Web Services Whitepapers, *Building Fault Tolerant Applications*, October 2011. http://media.amazonwebservices.com/AWS_Building_Fault_Tolerant_Applications.pdf

[3] Amazon Web Services, *Designing Fault-Tolerant Applications*, Slides, July 2011. http://www.slideshare.net/AmazonWebServices/base-camp-awsdesigningfaulttolerantapplications

[4] Amazon Web Services, *Designing Fault-Tolerant Applications*, YouTube, July 2011. https://www.youtube.com/watch?v=9BrmHoyFJUY

[5] Amazon Web Services, *Amazon S3*, viewed October 2014. http://aws.amazon.com/s3/

[6] Amazon Web Services, *Amazon EBS*, viewed October 2014. http://aws.amazon.com/ebs/

[7] Amazon Web Services, *RAID Configuration*, viewed October 2014. http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/raid-config.html

[8] Amazon Web Services, *Instance Storage*, viewed October 2014. http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/InstanceStorage.html

[9] Google, *Google Cloud Platform Documentation*. https://cloud.google.com/docs/

[10] Google Patents, *Data placement for fault tolerance*, February 2006. http://www.google.com/patents/US7000141

[11] Google Cloud Platform, *Architecture: Web Application on Google App Engine* https://cloud.google.com/solutions/architecture/webapp?hl=da

[12] Google Cloud Docs, *Regions & Zones* `https://cloud.google.com/compute/docs/zones`

[13] Google I/O, *App Engine: Scalability, Fault Tolerance, and Integrating Amazon EC2*, YouTube, June 2006. `https://www.youtube.com/watch?v=p4F62q1kJ7I`

[14] Google, *Google Cloud Platform Blog*. `http://googlecloudplatform.blogspot.dk/`

[15] Todd R. Weiss, *Google Cloud Platform Gets Developer Enhancements*, August 2013. `http://www.eweek.com/cloud/google-cloud-platform-gets-developer-enhancements`

[16] Larry Dignan, *Amazon Web Services, Windows Azure top cloud dev choices, says survey*, August 2013. `http://zd.net/1vw5pzE`

[17] National Institute of Standards and Technology, *The NIST Definition of Cloud Computing*, October 2011. `http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf`

[18] Prasenjit Kumar Patra, Harshpreet Singh and Gurpreet Singh, *Fault Tolerance Techniques and Comparative Implementation in Cloud Computing*, February 2013. International Journal of Computer Applications.

[19] Terrell Herzig et al., *Implementing Information Security in Healthcare: Building a Security Program*, HIMSS, February 2013