Cloud Standby: Disaster Recovery of Distributed Systems in the Cloud

Alexander Lenk[1], Stefan Tai[2]

[1] FZI Forschungszentrum Informatik, Berlin, Germany
lenk@fzi.de
[2] Technische Universität Berlin, Berlin, Germany
tai@tu-berlin.de

Abstract. Disaster recovery planning and securing business processes against outtakes have been essential parts of running a company for decades. As IT systems became more important, and especially since more and more revenue is generated over the Internet, securing the IT systems that support the business processes against outages is essential. Using fully operational standby sites with periodically updated standby systems is a well-known approach to prepare against disasters. Setting up and maintaining a second datacenter is, however, expensive. In this work, we present Cloud Standby, a warm standby approach for setting up and updating a standby system in the Cloud. We describe the architecture of Cloud Standby and its methods for deploying and updating the standby system. We show that by using Cloud Standby the recovery time and long-term costs of disaster recovery can significantly be reduced.

Keywords: Cloud Standby, IaaS, Warm Standby, Disaster Recovery, Distributed Systems

1 Introduction

Since the industrial revolution, protecting critical business processes against potential risks like earthquakes, fire, power outages, theft, illness, floods, and similar events has been a major concern of companies. Therefore, disaster recovery planning and preparing contingency plans for disaster preparedness have always been an integral part of running a company. To be prepared for these worst-case scenarios, disaster recovery plans are made in order to resume operations as soon as possible. These measures to keep up critical business processes in case of an emergency are often referred to as Business Continuity Management (BCM) [5] or, in the context of IT, as IT Service Continuity Management (ITSCM) [13]. The effectiveness of BCM can be controlled via the key figures Recovery Time Objective (RTO) and Recovery Point Objective (RPO) [5]. RPO refers to the maximum acceptable time between two backups whereas RTO defines the maximum reasonable time a business process may be interrupted (see also Fig. 1).

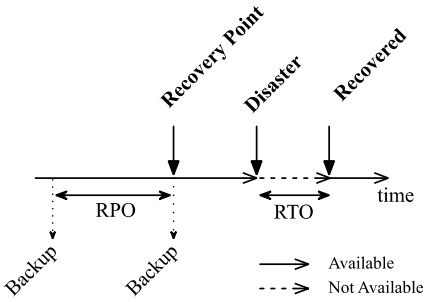

Fig. 1. Recovery Point Objective and Recovery Time Objective in the context of IT

To reduce the downtime of a system after an outage, it is common to replicate the whole system to another standby site [18] or even better, to another provider [14]. This is an established but very expensive approach that comes in different types: *hot standby* is a failover mechanism where all relevant data is consistently and continuously mirrored to a second data center with equivalent infrastructure almost in real-time. The main disadvantage of this procedure is the high costs. In addition to the operating costs of both systems, there are also the costs for mirroring. In contrast, there is *Cold standby*. Cold standby sites are updated only at times of low load such as nights or weekends and no standby systems are prepared. Therefore, in a case of a disaster, the standby system has to be ordered, deployed and equipped with the last backup. Therefore an RPO of days or weeks and an RTO of days or months are common. The third replication mechanism, warm standby, is positioned between cold and hot standby. A warm standby system has, similar to a hot standby system, an identical copy of the primary systems infrastructure but does not mirror data immediately. Instead, warm standby systems replicate the data periodically in short timeframes. Thus the primary and the secondary systems can have small amounts of different data that needs to be recreated in case of an outage. In general, warm standby systems have an RTO and RPO between minutes and hours.

To reduce operating costs of infrequently used IT components like warm standby systems, it is possible to use Cloud Computing [4, 16]. Cloud Computing provides the user with scalable, configurable IT resources over the internet with a pay-per-use pricing model [8, 12]. This means that the user only pays for resources he actually needs and unused resources can be used by other users. In the case of warm standby systems this pricing model makes Cloud Computing an ideal platform for hosting replication sites at reasonable prices with high availability [20]. Using a Cloud Computing datacenter as a standby site is especially interesting for small and medium companies that have all their servers in a single datacenter and do not have the possibility to run their own colocation center. Many of these small and medium-sized companies, however, should prepare for disaster with a standby system at another location or provider: according to recent studies [17], downtime costs in small and medium companies sum up to $12,500-23,000 per day and even data centers that are considered as highly available have reported downtimes [11].

In this work we introduce a novel approach for securing a distributed system against provider outages by using Cloud technology. We present Cloud Standby as a new method for disaster recovery of distributed systems in the Cloud. This method is composed based on a disaster recovery process for monitoring the standby site, updating the standby system, and initiating the emergency operation. Our focus thereby is on technical aspects of the IT service continuity process rather than on regulatory aspects or risk management (see **Fig. 2**).

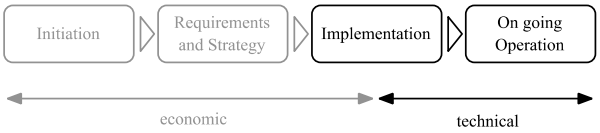

Fig. 2. IT service continuity process (cf. [13]) – the focus of this paper lies on the parts marked in black

The subsequent chapters of this paper are structured as followed: In Chapter 2, we refer to existing works. Chapter 3 introduces the novel Cloud Standby approach. We present the results of an evaluation with the company "barcoo" in Chapter 4. The final Chapter 5 concludes the findings and provides an outlook for our future work.

2 Related Work

In the following we describe state of the art warm and hot standby approaches that are using Cloud Computing as standby sites. We exclude cold standby in this discussion, because in cold standby no standby system is prepared.

2.1 Warm standby in the Cloud

Wood et al. [20] analyze the cost reduction by having a Cloud provider as standby site. In contrast to our work, this paper focuses on the economic part and does not present a concrete warm standby system. The authors, however, identify that the pay-as-you-go Cloud computing billing is especially effective in the warm standby. Pokharel et al. [14] reach the same conclusion. They also recommend that the primary system and replication system should be deployed geographically apart. This ensures that if a whole datacenter goes down, one of the systems will still be available and the business can continue with only a short interruption. In their approach they describe an algorithm that allows identifying outages and initiating the emergency operation.

Klems et al. [6] present a concrete warm standby approach that is using Cloud technology to run the replication system. Also, Klems at al. point out that using Cloud Computing in the context of warm standby systems can lead to a reduction of costs and deployment time. In their work they focus on single servers rather than on distributed systems and present a mechanism for backing up the primary virtual machine. Thus, for every primary virtual machine they provision two virtual machines in the

Cloud, leading to an unnecessary overhead if there are already data backup mechanisms in place.

2.2 Hot Standby in the Cloud

The hot standby approach PipeCloud [21] replicates virtual machines to the cloud by copying all writing operations to the virtual hard disk. To get access to these writing operations, this approach needs access to the hypervisor. All the writing operations are asynchronously sent to the cloud and stored in a queue where they are applied to the virtual images stored there. This allows this approach to be used if the RPO is very small; however, it also introduces the problem that the primary system cannot run on a public cloud since there is no access to the hypervisor and the disk-writing operations themselves in the public cloud. It also introduces a huge traffic overhead as every single writing operation has to be packaged and sent to the Cloud over the Internet. Similar approaches are called Remus[3] and SecondSite [15]. Both of them rely on access to the hypervisor for copying the writing operations and sending them over the network. Remus, though, is not focusing on the Cloud but rather on replicating the virtual machine to another hypervisor within the same datacenter, and with the additional SecondSite it is possible to copy one or more virtual machines to the hypervisor of another datacenter. None of the presented approaches, however, are applicable in the public Cloud due to their requirement to have low level access to the hypervisor.

In conclusion it can be noted that in the related work there are approaches for warm and hot standby but none of them can be used to easily setup and update a standby system of a distributed system in the public Cloud.

3 Cloud Standby

In this section we describe Cloud Standby, a novel approach for replicating a distributed system in the Cloud to another Cloud. In the following subsections we describe the components and possible states of the Cloud Standby system, and the process for replicating the primary system to the Cloud.

3.1 Components and methods

The Cloud Standby method consists of several different components and methods. An overview of the Cloud Standby method can be found in Fig. 3.

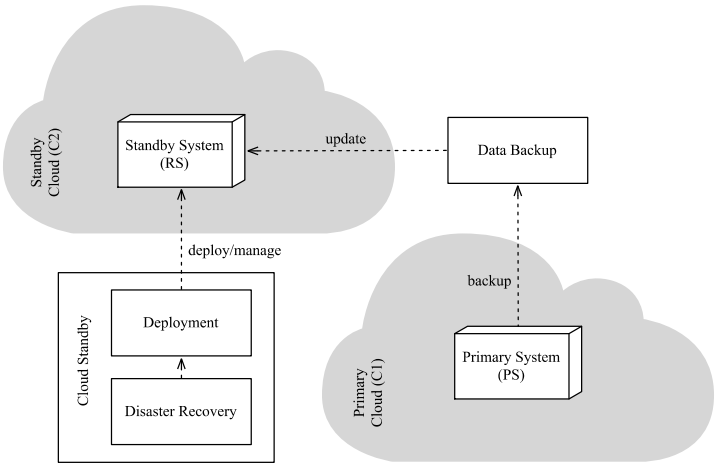

Fig. 3. Overview of the combination of methods

- **Primary System (PS)** – The primary system is the distributed system that needs to be secured by the warm standby approach. This system, with its given architecture and features like failure tolerance is secured as a whole, so that the same architecture and features are still present after the disaster. The primary system is backed up repeatedly using a state of the art data backup method as to meet a desired RPO.
- **Standby System (RS)** – The standby system is a copy of the primary system, which takes over the operation in the case of an emergency. The Cloud Standby approach aims to deploy this system and keep it up to date. In order to enable the standby system to take over the operation, the requests of the users of the distributed system must also reach the emergency system. There are various methods for handing over external requests to the standby system, for example the use of virtual IP addresses or dynamic DNS entries. These methods are not part of this work, but established methods of the state of the art like the work of Ayari et al. [2] can be used in this case.
- **Cloud (C1 and C2)** – The exact location of resources like virtual machines or storages are often veiled in Cloud Computing. In this paper, "Cloud" describes the location of a Cloud data center and refers to a logical unit where computing power and memory is supplied. Within this Cloud, transaction costs like traffic are not charged. The Cloud is a runtime environment for virtual machines and provides storage. These Cloud resources can be maintained with the Cloud Management Interface. Typical management tasks are creating, reading, updating and deleting resources.

In the following we describe the methods that are applied to the components. The tools and implementations supporting these methods can be located anywhere. We, however, recommend having them hosted in the Standby Cloud for availability and cost reasons: If the primary cloud becomes unavailable the Standby System can still be started with the last version of the backup. Also many Cloud providers do not

charge data transfer within a single datacenter, so the data transfer during the update cycle of the standby system is free of charge.

- **Data Backup** – The data backup method includes a central database that contains all backups for the distributed system and is filled by backup software of the respective virtual machines of the primary system. The backup interval has to match the RPO. The RPO applies to the superordinate business process, hence to every system and virtual machine involved in the process. We assume that the current backup status is adequate for the RPO. Therefore, the backup ensures that the replicated data is consistent. Only after every virtual server has saved its data, is the backup completed and unlocked for restoration. The backup serves as a consistent central data source that can be accessed from the primary as well as from the standby system. By using the backup as a common component, existing solutions are integrated and it is ensured that the standby system can be updated to a condition that matches the RPO.
- **Deployment –** Besides the Disaster Recovery Method, the Deployment Method is a key part of Cloud Standby. It coordinates and maintains the infrastructure like virtual servers and network configurations that are involved in the distributed system[1]. The Deployment Method has all information about the distributed system like IP addresses or access data and serves as the information source for other components. As a deployment method, a state of the art Cloud provider-independent deployment method like TOSCA [23] can be used. We, however, recommend to use our tailored Cloud Standby deployment model [7, 9] that has native support of Cloud federation and data backup and is thereby ready for the usage with the disaster recovery method presented in this paper. Our deployment method is available online[2].
- **Disaster Recovery** – Tasks of the Disaster Recovery Method are updating the standby system, and initiating and terminating the emergency mode. Therefore, it uses the deployment method for every task that effects the administration of single entities. The disaster recovery method consists of the emergency backup process and the update protocol.

In the following, we further describe the disaster recovery method by detailing valid state transitions and the disaster recovery process. We design our system as a warm standby approach. That is, the standby systems gets updated periodically with the data backup available within the data backup store as part of the data backup method. So, setting up the data backup correctly is a preliminary requirement for our method. Fig. 4 gives an overview of the backup and update process and illustrates the independence of the two. While the backup process depends on the RPO, the update of the standby system happens afterwards and is defined as $t_{updateInt}$.

---

[1] Cloud infrastructure specific features like availability zones, firewall configurations, or external storage are specified with the deployment method and are not in the focus of this paper.

[2] https://github.com/alexlenk/CloudStandby/tree/master/org/cloudstandby/model

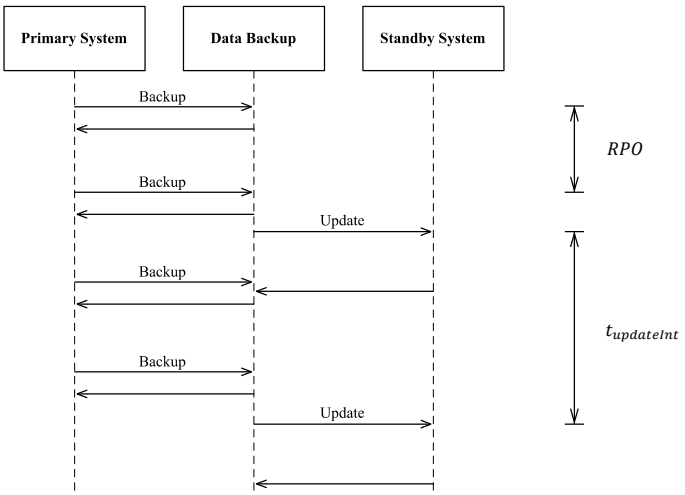

Fig. 4. Exemplary representation to show the independence of the update and backup interval as a UML sequence diagram

In order to do the periodic updates necessary in a warm standby approach we start the system with the deployment method, restore the last backup, and save the updated images. Therefore, in the next run, only the changed data has to be updated on the image which reduces the time until the instance is fully available. In the following section we describe the states of a Cloud Standby system during the disaster recovery process.

3.2 Disaster Recovery States

Cloud Standby is based on a warm standby approach where a Primary System (PS) running in the Primary Cloud (C1) is periodically synchronized as a Standby or Replica System (RS) to Standby Cloud (C2). The states and state transitions of our Cloud Standby approach are depicted in Fig. 5.

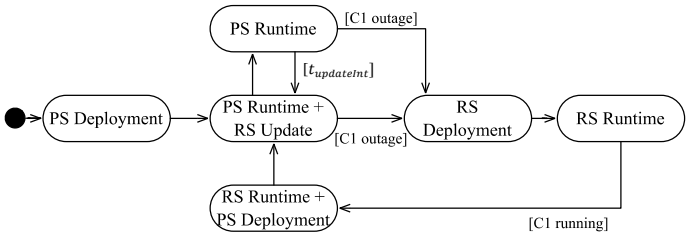

Fig. 5. UML state chart of a Cloud Standby system (c.f. [9, 10])

- **PS Deployment –** The PS is deployed on C1 at first. The deployment time depends highly on the structure of the deployment. For each use case, the deployment time

can be determined by experimentation. It depends, however, on the amount of data that has to be copied to the virtual machine. If the deployment contains a stateful server, causing a lot of data changes over time, the deployment time can rise quickly (see section 4.2). After the initial deployment a RS update is performed to ensure the RS can take over immediately in the case of a disaster.

- **PS Runtime + RS Update –** Periodically (after $t_{updateInt}$) the RS is updated. This ensures that the deployment time of the RS is reduced when an actual disaster occurs.
- **PS Runtime –** During PS runtime, the RS is turned off and generates no costs. The PS data are, however, backed up using standard backup methods. This ensures the RPO can be met in case of a disaster (see Fig. 4).
- **RS Deployment –** When C1 fails the RS is started and takes over the service. The time for the deployment is $t_{replDepl}$. The deployment time varies with the amount of data that needs to be installed or stored during the deployment process. This means that $t_{replDepl}$ decreases with a decreasing $t_{updateInt}$. The correlation between $t_{replDepl}$ and $t_{updateInt}$ can be determined through experiments or monitoring over time.
- **RS Runtime –** In the case of an outage on C1, the RS takes over and only if during this time an outage also takes place on C2 is whole system unavailable.
- **RS Runtime + PS Deployment –** As soon as C1 is up again, the PS can be redeployed and then takes over the service.

In the following section we describe a high level process that implements the states described in this section.

3.3 Disaster Recovery Process

The main purpose of the process described in this section is to provide all the functionality and parallel tasks that are necessary for the Cloud Standby approach. The process ensures that the primary Cloud is monitored, the emergency mode is activated and deactivated, and that the update of the standby system is triggered. The disaster recovery process is depicted in Fig. 6 as an UML activity diagram.

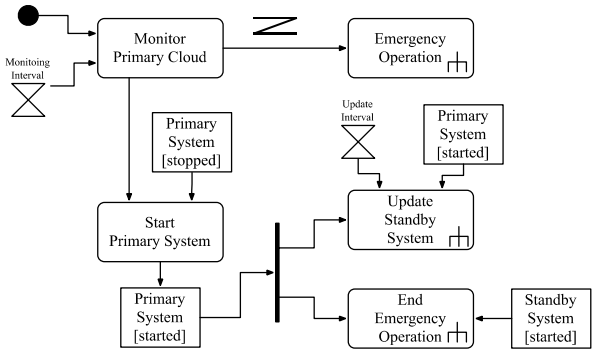

Fig. 6. Disaster recovery process

There are two parallel activities that are executed while the primary system is running: monitoring of the primary Cloud, and the update of the standby system. Both of these activities run continuously and are periodically restarted. Each of them has its own timer and can be externally controlled, but both depend on the state of the standby system - for example, updating the standby system is only possible if the primary system is in the state ‘started’. Once the primary system is stopped, this activity can’t be executed, but the failure of the monitoring activity also automatically triggers the emergency operation. Once the primary Cloud is available again, but the primary system is still stopped, the primary system is started and the operation is switched back to normal mode so that the update can be executed.

The updating process is depicted in Fig. 7. After the standby system is started, the update of the system is initiated by restoring the last backup with the standard backup method. Once the restoring process has finished, the current state is saved and the deployment is stopped again. When using our model-based deployment method [7] the update of the standby system can be modelled within the deployment. Therefore, it can be ensured that the update cycle is always executed and no data is lost. Even if all presented processes and methods are applied correctly, it is however crucial to test the disaster recovery process from time to time. We therefor recommend including live testing during ongoing operation as part of the ITSCM (see section 1).

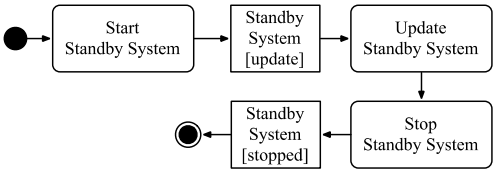

Fig. 7. Update Standby System Process

So far, we described the Cloud Standby approach comprising a set of methods and the disaster recovery process. In the following section, we evaluate this new method with a real world use case.

4 Evaluation

For the evaluation we implemented[3] the Cloud Standby system and used a real world use case in order to do an experimental evaluation. Further, a long term simulation based on a Markov chain approach has been carried out (for method and further information on the simulation please see [9, 10]).

[3] Implementation available at: https://github.com/alexlenk/CloudStandby

4.1 Use Case

For our evaluation, we use the infrastructure of the barcoo[4] application. Barcoo is a German startup with over 10 million app downloads. Barcoo is one of the most famous apps in the German Apple App store. The barcoo infrastructure consists of four components: a load balancer, a set of application servers, a MySQL database, and a NoSQL database (for caching). The components and their dependencies are shown in Fig. 8.

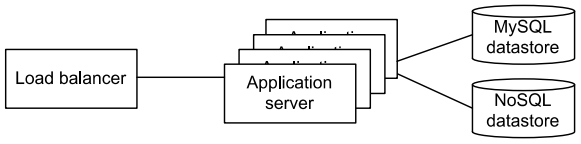

Fig. 8. Barcoo application components (c.f. [19])

The load balancer is used as an access point for all clients (browser, smartphone app, etc.). Here, the requests are forwarded to each application server. Each application server uses data from the MySQL database or NoSQL-cache in order to make the appropriate information available to the user. Data in the NoSQL store comes mainly from external providers (price service provider, warnings about food scandals, etc.) and are cached only for performance and bandwidth reasons.

Besides non-critical cache data, the production system of barcoo also includes mission-critical and privacy-related data. For the unlimited use of this data, special precautions and consents to third parties are required, which cannot be obtained in the evaluation. For this reason, not all data of barcoo is available for evaluation and will be replaced by non-critical data for the evaluation of this paper. Thus, it is also possible to reproduce the findings of this work without access to the internal data of barcoo.

Data and application changing rate. The business data of barcoo is subject to permanent changes: new products are constantly added to the database, prices are updated, new users are included, and so on. The requests that are addressed to the database are a mixture of insert, update, and delete operations. The examination of the data in the MySQL database for several years revealed a trend towards a continuous increase of the amount of data. Therefore, only insert operations are considered in this evaluation. Due to the historical data available it is assumed that the database is growing by 400 MB within one week.

The barcoo application is subject to a continuous development. As part of the agile Scrum development process, there are a number of minor and major releases of new versions. In the context of this evaluation, it is assumed that, as usual for Scrum, a new and bigger version is published every two weeks, which makes it also necessary to update the underlying software packages of the operating system. Fig. 9 once more

[4] http://www.barcoo.com

illustrates the relationship between the change of the MySQL data and the application server updates.

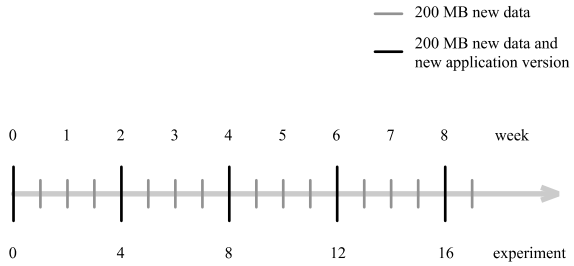

Fig. 9. Sequence of the experiments

Simulation of critical business data. In the context of this evaluation, sensitive business and customer data are replaced with publicly available data sets and applications. However, to receive realistic results, the amount and the structure of the data were adjusted in close relation to the real data. More precisely, the MySQL database is filled with data of the Ensembl gene database [22]. These data include both large quantities of numbers that are comparable to the prices stored in the barcoo database as well as longer texts that correspond to other data stored.

The Ensembl database contains the data in the form of databases, where each database comprises the genetic code of a form of life. Initial analyses of the data showed that parts of the data can be faster imported than others. This can be due to the fact that some tables only consist of numbers whereas others consist of larger amounts of text. In order to prevent that these different import times distort the measurements, all data of an Ensembl database were converted into MySQL import commands and written into a large MySQL file. Each line corresponded to an import command in the data file. Subsequently, the lines of the large, multi-gigabyte file were randomly mixed. The result was divided into small files, which can be individually imported into an existing schema and enlarge the final database to 100 MB. For reasons of better transportation, these 100 MB parts were packed with GZip[5].

The data of the NoSQL database are kept in the main memory and are not of crucial importance for the operation, it just reduces the number of requests to external providers over the time. In an emergency, the NoSQL database is deployed empty and gradually fills itself.

As the barcoo application is the company's main core business, its publication would definitely harm the business. Thus, for this evaluation, every installation step for setting up a Ruby-on-Rails server that is capable of running the barcoo-application has been available but the application itself has not. To simulate the application, the github project "diaspora" was used. We chose diaspora because it is an open-source Ruby-on-Rails application and there are several versions available, so the development of several versions over a certain time can be simulated.

[5] https://www.gnu.org/software/gzip/

4.2 Results

With the given use case we set up the Cloud Standby system and by applying the changed data according to the description in the use case, we simulated a 24 week runtime of the system by repeating it 48 times within a short period of time. To evaluate the deployment times, we measured the time when the first startup call was sent to the cloud provider until the last installation package of the last server was installed. The results are shown in the following sections.

We first compared the startup time of the distributed system with increasing business data and new application versions over time with and without Cloud Standby replication. By measuring the starting time without replication we created a reference on how the startup time develops with increasing data. We used a curve estimation regression on the measured data point to fit a function that we can use in our simulation later. Since the Cloud Standby replication comes with a price that has to be paid to the Cloud provider when updating the standby system, we used the data we gained from the experiments to calculate the overhead in costs for a given RTO. With this information we then simulated the long term costs of the Cloud Standby system using a Markov chain based approach that also takes the outages costs into account and can thereby estimate in which cases our Cloud Standby approach is useful and in which it is not.

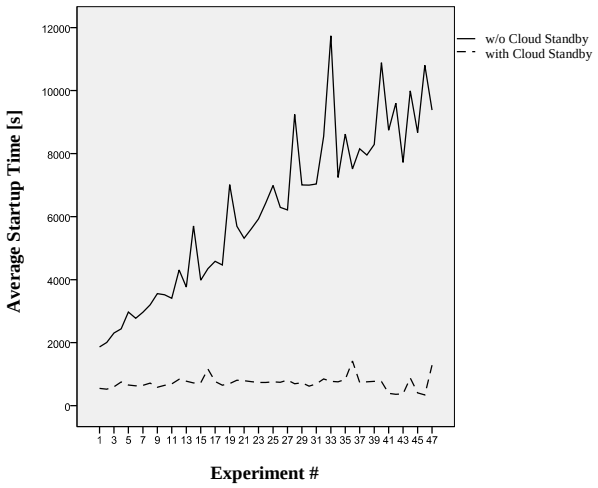

Fig. 10. Comparison between the deployment time with and without Cloud Standby

Reduction of the RTO. Considering the deployment time of a system with and without Cloud Standby Fig. 10 shows that the difference between the deployment time of the respective system rises with the size of the backup.

By doing a fit on the measured data points we determined the following function:

$$f(x) = 1234.742x^{0.519} \quad (1)$$

Additional Costs. The costs for this approach arise from the starting of the standby system consisting of 7 virtual machines[6] for update reasons. Thereby the standby costs are linked to the configured update interval (see section 3.3). The update interval also defines what RTO can be achieved. So, we used interpolation on the startup times in Fig. 10 and with the resulting function and a cost function for periodic Cloud updates (see [9, 10]) we calculated the costs of running a standby system. Apart from breakdown costs, these costs are solely overhead. In our case, the overhead for RTO is between 2-5 % of the primary system hosting costs (see Fig. 11).

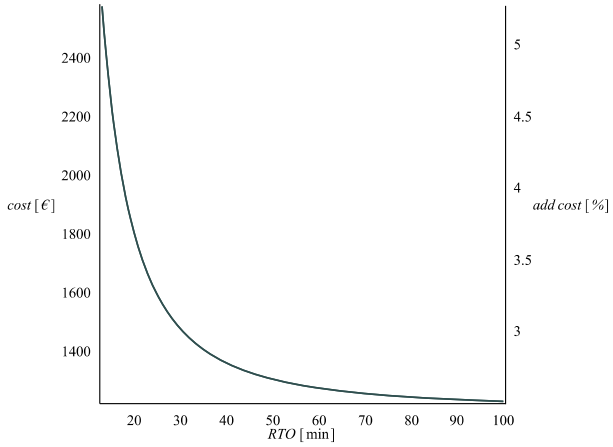

Fig. 11. Costs for Cloud Standby in relation to the recovery time objective as an absolute value and percentage additional overhead costs

Long-term savings. Taking the outage probability of the primary provider, the outage costs ($cost_e$) of the business processes, and the startup time (see formula 1) into account, we can calculate for the given use case when Cloud Standby should be used and when not[7]. As shown in Fig. 12, Cloud Standby is not useful in the case of very high or very low outage costs. When these costs are very high a hot standby approach should be considered and when they are very low it might be better to not use any standby approach at all. However, as shown in Fig. 12, in many cases our approach is beneficial (grey area). In our use case this approach should be used when the outage costs are between 4.88€ and 2989.97€ per hour.

[6] Usage cost for the used virtual machines types on Amazon EC2: 0.68€/h [1]

[7] For calculating the long term savings we used the calculation method presented in [9, 10]

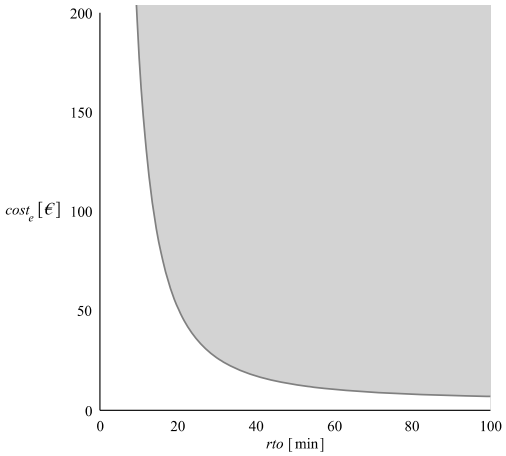

Fig. 12. Area (grey) in which Cloud Standby is cheaper than risking an outage (c.f. [9, 10])

5 Conclusion

In this paper we presented a novel approach for warm standby in the cloud. We introduced Cloud Standby and its methods and processes for preparing, monitoring, and updating the standby system. We evaluated the system using a real-world use case where we deployed a multi-tier application several times with increasing data in the cloud. We showed that by using Cloud Standby, the startup time, and thereby the smallest possible RTO, can be significantly reduced. We also showed that in this use case, the cost overhead of the approach is between 2-5% and that when taking the outage costs of the business process into account, our approach should be used when these costs are between 4.88€ and 2989.97€ per hour.

In our future work we will lay additional focus on the deployment method and describe how a model-driven approach can be used to complement our disaster recovery method. We also plan to extent the implementation of our approach.

Acknowledgements. We would like to thank Tobias Bräuer, CTO of barcoo, for his help and the insights he gave us on the barcoo infrastructure.

References

1. AWS Inc.: Amazon Web Services, Cloud Computing: Compute, Storage, Database, https://aws.amazon.com/.
2. Ayari, N. et al.: Fault tolerance for highly available internet services: concepts, approaches, and issues. Commun. Surv. Tutor. IEEE. 10, 2, 34–46 (2008).
3. Cully, B. et al.: Remus: High Availability Via Asynchronous Virtual Machine Replication. Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation. pp. 161–174 (2008).

4. Henderson, C.: Building scalable web sites. O'reilly (2008).
5. Hiles, A.: The definitive handbook of business continuity management. Wiley (2010).
6. Klems, M. et al.: Automating the delivery of IT Service Continuity Management through cloud service orchestration. Network Operations and Management Symposium (NOMS), 2010 IEEE. pp. 65–72 (2010).
7. Lenk, A.: Cloud Standby Model Implementation, https://github.com/alexlenk/CloudStandby/tree/master/org/cloudstandby/model.
8. Lenk, A. et al.: What's inside the Cloud? An architectural map of the Cloud landscape. Software Engineering Challenges of Cloud Computing, 2009. CLOUD'09. ICSE Workshop on. pp. 23–31 (2009).
9. Lenk, A., Pallas, F.: Cloud Standby System and Quality Model. Int. J. Cloud Comput. IJCC. 1, 2, 48–59 (2013).
10. Lenk, A., Pallas, F.: Modeling Quality Attributes of Cloud-Standby-Systems. Service-Oriented and Cloud Computing. pp. 49–63 Springer (2013).
11. Li, Z. et al.: The Cloud's Cloudy Moment: A Systematic Survey of Public Cloud Service Outage. ArXiv Prepr. ArXiv13126485. (2013).
12. Mell, P., Grance, T.: The NIST definition of cloud computing. NIST Spec. Publ. 800, 145 (2011).
13. Menken, I. et al.: Itil V3 Malc-Managing Across the Lifecycle Full Certification Online Learning and Study Book Course: The Itil V3 Intermediate Malc Complete Certification Kit. Emereo Pty Limited (2009).
14. Pokharel, M. et al.: Disaster Recovery for System Architecture Using Cloud Computing. Presented at the July (2010).
15. Rajagopalan, S. et al.: SecondSite: disaster tolerance as a service. Proceedings of the 8th ACM SIGPLAN/SIGOPS conference on Virtual Execution Environments. pp. 97–108 (2012).
16. Schmidt, K.: High availability and disaster recovery. Springer (2006).
17. Symantec: 2011 SMB Disaster Preparedness Survey - Global Results. (2011).
18. Whitman, M. et al.: Principles of incident response and disaster recovery. Cengage Learning (2013).
19. Wittern, E. et al.: Feature-based Configuration of Vendor-independent Deployments on IaaS. 18th IEEE International Enterprise Distributed Object Computing Conference (EDOC). (2014).
20. Wood, T. et al.: Disaster recovery as a cloud service: Economic benefits & deployment challenges. 2nd USENIX Workshop on Hot Topics in Cloud Computing. (2010).
21. Wood, T. et al.: PipeCloud: using causality to overcome speed-of-light delays in cloud-based disaster recovery. Proceedings of the 2nd ACM Symposium on Cloud Computing. p. 17 (2011).
22. Ensembl Genome Browser, http://www.ensembl.org/index.html.
23. Topology and Orchestration Specification for Cloud Applications Version 1.0, http://docs.oasis-open.org/tosca/TOSCA/v1.0/cs01/TOSCA-v1.0-cs01.pdf.