Journal of Cloud Computing: Advances, Systems and Applications

REVIEW Open Access

High availability in clouds: systematic review and research challenges

Patricia T. Endo[1,2*], Moisés Rodrigues[2], Glauco E. Gonçalves[2,3], Judith Kelner[2], Djamel H. Sadok[2] and Calin Curescu[4]

Abstract

Cloud Computing has been used by different types of clients because it has many advantages, including the minimization of infrastructure resources costs, and its elasticity property, which allows services to be scaled up or down according to the current demand. From the Cloud provider point-of-view, there are many challenges to be overcome in order to deliver Cloud services that meet all requirements defined in Service Level Agreements (SLAs). High availability has been one of the biggest challenges for providers, and many services can be used to improve the availability of a service, such as checkpointing, load balancing, and redundancy. Beyond services, we can also find infrastructure and middleware solutions. This systematic review has as its main goal to present and discuss high available (HA) solutions for Cloud Computing, and to introduce some research challenges in this area. We hope this work can be used as a starting point to understanding and coping with HA problems in Cloud.

Keywords: Cloud computing, High availability, Systematic review, Research challenges

Introduction

Cloud Computing emerged as a novel technology at the end of the last decade, and it has been a trending topic ever since. The Cloud can be seen as a conceptual layer on the Internet, which makes all available software and hardware resources transparent, rendering them accessible through a well-defined interface. Concepts like on-demand self-service, broad network access, resource pooling [1] and other trademarks of Cloud Computing services are the key components of its current popularity. Cloud Computing attracts users by minimizing infrastructure investments and resource management costs while presenting a flexible and elastic service. Managing such infrastructure remains a great challenge, considering clients' requirements for zero outage [2, 3].

Service downtime not only negatively effects in user experience but directly translates into revenue loss. A report [4] from the International Working Group on Cloud Computing Resiliency (IWGCR)[1] gathers information regarding services downtime and associated revenue losses. It points out that Cloud Foundry[2] downtime results in $336,000 less revenue per hour. Paypal, the online payment system, experiences in a revenue loss of $225,000 per hour. To mitigate the outages, Cloud providers have been focusing on ways to enhance their infrastructure and management strategies to achieve high available (HA) services.

According to [5] availability is calculated as the percentage of time an application and its services are available, given a specific time interval. One achieves high availability (HA) when the service in question is unavailable less than 5.25 minutes per year, meaning at least 99.999 % availability ("five nines"). In [5], authors define that HA systems are fault tolerant systems with no single point of failure; in other words, when a system component fails, it does not necessarily cause the termination of the service provided by that component.

Delivering a higher level of availability has been one of the biggest challenges for Cloud providers. The primary goal of this work is to present a systematic review and discuss the state-of-the-art HA solutions for Cloud Computing. The authors hope that the observation of such solutions could be used as a good starting point to addressing with some of the problems present in the HA Cloud Computing area.

*Correspondence: patricia.endo@upe.br
[1] University of Pernambuco (UPE), BR 104 S/N, Caruaru, Brazil
Full list of author information is available at the end of the article

This work is structured as follows: "Cloud outages" section describes some Cloud outages that occurred in 2014 and 2015, and how administrators overcame these problems; "Systematic review" section presents the methodology used to guide our systematic review; "Overview of high availability in Clouds" section presents an overview regarding HA Cloud solutions; "Results description" section describes works about HA services based on our systematic review result; "Discussions" section discusses some research challenges in this area; and "Final considerations" section delineates final considerations.

Cloud outages

Cloud Computing has become increasingly essential to the live services offered and maintained by many companies. Its infrastructure should attend to unpredictable demand and should always be available (as long as possible) to end-clients. However, assuring high availability has been a major challenge for Cloud providers. To illustrate this issue, we describe four (certainly among many) examples of Cloud services outages that occurred in 2014 and 2015:

Dropbox

Dropbox's Head of Infrastructure, Akhil Gupta, explained that their databases have one master and two replica machines for redundancy, and full and incremental data backups are performed regularly. However, on January 10th, 2014[3], during a planned maintenance scheduled intended to upgrade the Operating System on some machines, a bug in the script caused the command to reinstall a small number of active machines. Unfortunately, some master-replica pairs were impacted which resulted in the service going down.

To restore it, they performed the recovery from backups within three hours, but the large size of some databases delayed the recovery. The lesson learned from this episode was the need to add a layer to perform distributed state verification and speed up data recovery.

Google services

Some Google services, such as Gmail, Google Calendar, Google Docs, and Google+, were unavailable on January 24th, 2014, for about 1 hour. According to Google Engineer, Ben Treynor, "*an internal system that generates configurations - essentially, information that tells other systems how to behave - encountered a software bug and generated an incorrect configuration. The incorrect configuration was sent to live services over the next 15 minutes, caused users' requests for their data to be ignored, and those services, in turn, generated errors*".

Consequently, they decided to add validation checks for configurations, improve detection, and diagnose service failure.

Google Apps

The Google Apps Team schedules maintenance on data center systems regularly and some procedures involve upgrading groups of servers and redirecting the traffic to other available servers. Typically, these maintenance procedures occur in the background with no impact on users. However, due to a miscalculation of memory usage, on March 17th, 2014 the new set of backend servers lacked of sufficient capacity to process the redirected traffic. These backend servers could not process the volume of incoming requests and returned errors for about three hours.

The Google Engineering team said that they will "*continue work in progress to improve the resilience of Hangouts service during high load conditions*".

Verizon Cloud

Verizon Cloud[4] is a Cloud provider that offers backup and synchronization data to its clients. On January 10th, 2015 Verizon provider suffered a long outage of approximately 40 hours over a weekend. The outage occurred due to a system maintenance procedure which, ironically, had been planned to prevent future outages.

So, as we can see, Cloud outages can occur from different causes and can be fixed using different strategies. However, in most cases, in addition to the loss of revenue, such service disruptions pushed Cloud providers to rethink their management strategies and sometimes to re-design their Cloud infrastructure design altogether.

Financial losses due to Cloud outages foment studies about HA solutions, in order to minimize outages for Cloud providers. In the next Section, we describe the systematic review approach that we used to undertake research about HA solutions.

Systematic review

In this work, we adapted the systematic review proposed by [6], in order to find strategies that address HA Clouds. Next, we describe each activity (see Fig. 1) in detail and describe how we address it.

Activity 1: identify the need for the review

As stated previously, high availability in Clouds remains a big challenge for providers since Cloud infrastructure systems are very complex and must address different services with different requirements. In order to reach a certain level of high availability, a Cloud provider should monitor its resources and deployed services continuously.

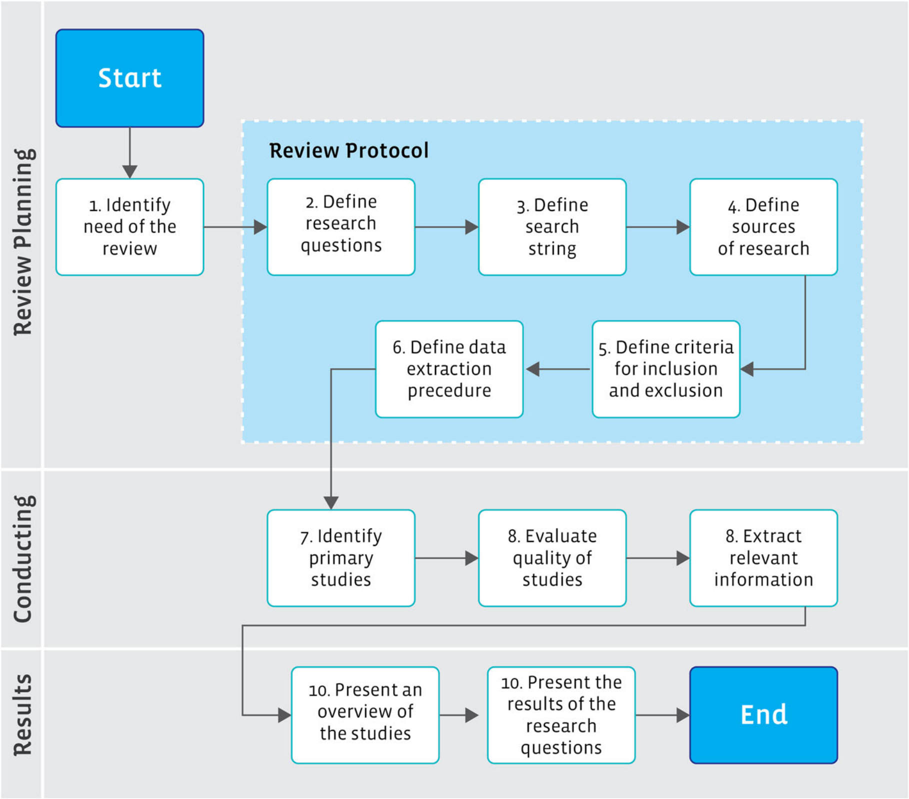
Review Planning
Conducting
Results
Start
Review Protocol
1. Identify need of the review
2. Define research questions
3. Define search string
4. Define sources of research
5. Define criteria for inclusion and exclusion
6. Define data extraction precedure
7. Identify primary studies
8. Evaluate quality of studies
8. Extract relevant information
10. Present an overview of the studies
10. Present the results of the research questions
End

Fig. 1 Systematic review process

With information about resources and service behaviors available, a Cloud provider could make good management decisions in order to avoid outages or failures.

Activity 2: define research questions

In this activity, we need to define which questions we want to answer. The main goal of this work is to answer the following research questions (RQ):

- RQ.1: What is the current state-of-the-art in HA Clouds?
- RQ.2: What is the most common definition of HA?
- RQ.3: What are the HA services implemented by HA Cloud solutions?
- RQ.4: What are the most common approaches used to evaluate HA Cloud solutions?
- RQ.5: What are the research challenges in HA Clouds?

Activity 3: define search string

In this activity, we need to define which keywords we will use in selected search tools. For this work, we used the following expressions: “cloud computing” AND “high availability” AND “middleware”.

Activity 4: define sources of research

For this work, we chose the following databases: IEEE Xplore[5], Science Direct[6], and ACM Digital Library[7].

Activity 5: define criteria for inclusion and exclusion

In order to limit the scope of this analysis, we considered only journals and conferences articles published between 2010 and 2015. The keywords “cloud computing” and “middleware” or “framework” were required to be in the article.

Activity 6: define data extraction procedure

Data extraction is based on a set of items to be filled for each article: keywords, proposal, and future works.

Activity 7: identify primary studies

The search returned 9, 63, and 145 articles in IEEE Xplore, Science Direct, and ACM Digital Library, respectively, totaling 217 works.

By reading all abstracts and using the criteria for inclusion or exclusion, we selected 19 papers for data extraction and quality evaluation. This number is justified because the keyword "high availability" is very common in Cloud Computing, especially in its own definition, and so most of articles had this keyword in them. However, in most cases high availability was not their research focus.

Activity 8: evaluate quality of studies

The quality evaluation was based on checking if the paper is related to some HA Cloud proposal for middleware or framework.

Activity 9: extract relevant information

This activity involves applying the data extraction procedure defined in Activity 6 to the primary studies selected in Activity 7.

Activity 10: present an overview of the studies

In this activity, we present an overview of all articles we selected in Activity 8, in order to classify and clarify them according to the research questions presented in Activity 2. The result of this activity is presented in "Overview of high availability in Clouds" section.

Activity 11: present the results of the research questions

After an overview about studies in HA Clouds, we had a discussion in order to answer the research questions stated in Activity 2. The results of this activity are presented in "Overview of high availability in Clouds" section.

Overview of high availability in Clouds

In this Section, we present an overview about Activity 10, presenting some characteristics of the selected articles in HA Cloud. Figure 2 shows the number of articles published per year from 2010 to 2015.

Concerning research source (Fig. 3), we can see that ACM has more articles published in HA Cloud area.

Some articles define the term "high availability". For instance, authors in [7] say "the services provided by the applications are considered highly available if they are accessible 99.999 % of the time (also known as five 9's)". The Table 1 outlines the various definitions of "high availability" we identified through our research, as well as the source of each definition.

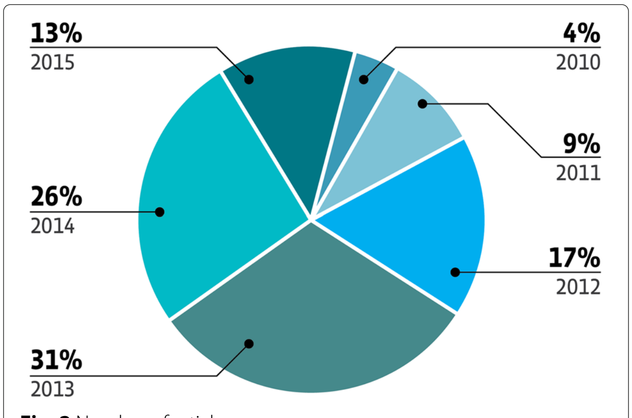

Fig. 2 Number of articles per year

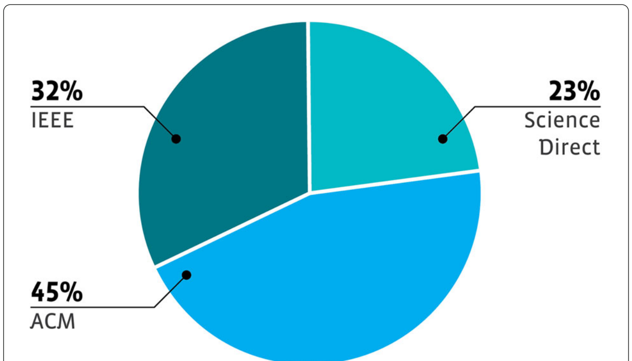

Fig. 3 Number of articles per research source

We also observed that many services are implemented in conjunction in order to offer a HA Cloud. Figure 4 shows monitoring, replication, and failure detection are the most implemented services, identified in 50 % of studies in the research. Please, note that there are more services than published works because it is common to implement more than one service in a proposal.

Figure 5 shows how solutions were evaluated in the studies we analyzed. We can see experimentation is the most popular technique used. These results indicate that research about this topic is working to derive proposals with fast application to the cloud computing industry.

Table 1 High availability definitions

Reference	Definition
Achieving High Availability at the Application Level in the Cloud [7]	The services provided by the applications are considered highly available if they are accessible 99.999 % of the time (also known as five 9's)
Managing Application Level Elasticity and Availability [25]	High availability is achieved when the outage is less than 5.25 minutes per year
Scheduling highly available applications on cloud environments [35]	High availability systems are characterized by fewer failures and faster repair times
Are clouds ready for large distributed applications? [36]	High availability is defined in terms of downtime that is the total number of minutes the site is unavailable for events lasting longer than 5 minutes over a 1-year period
Software aging in the eucalyptus cloud computing infrastructure: Characterization and rejuvenation [37]	Availability is defined as the ability of a system to perform its slated function at a specic instant of time.

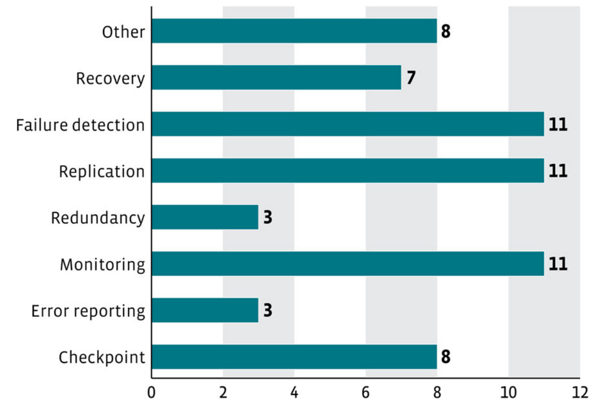

Fig. 4 HA services implemented by solutions

The analysis should be performed based on comparison metrics. Work presented in [8] defines some metrics used to evaluate HA solutions, as shown in Table 2.

Results description

As we found in this systematic review, Cloud providers can make use of several technologies and mechanisms to offer HA services. Authors in [9] classify HA solutions into two categories: middleware approaches and virtualization-based approaches. They propose a framework to evaluate VM availability against three types of failures: a) application failure, b) VM failure, and c) host failure. Authors use OpenStack, Pacemaker, OpenSAF, and VMware to apply their framework, which considers stateful and stateless-HA applications.

However, in our research, we organize solutions into three layers (underlying technologies, services, and middlewares), and keep in mind that layers can be composed of (one or many) solutions from bottom layers to perform their goals (Fig. 6).

Our classification is a simplified view of the framework proposed by Service Availability Forum (SAForum) (Fig. 7). SAForum is focused on producing open specifications to address the requirements of availability, reliability and dependability for a broad range of applications (not only Clouds).

There are three types of services in its Application Interface Specification (AIS): Management Services, Platform Services, and Utility Services. According to [10], Management Services provide the basic standard management interfaces that should be used for the implementation of all services and applications. Platform Services provide a higher-level abstraction of the hardware platform and operating systems to the other services and applications. Utility Services provide some of the common interfaces required in highly available distributed systems, such as checkpoint and message.

SAF also proposes two frameworks: Software Management Framework (SMF), which is used for managing middleware and application software during upgrades while taking service availability into account; and Availability Management Framework (AMF), which provides functions (e.g. a set of APIs) for availability management of applications and middleware [10], such as component registration and life cycle management, error reporting and health monitoring.

We understand our 3-layer classification covers the SAF framework, because SAF specifications can be allocated between our layers. The next sub-sections will present solutions found in our systematic review focusing on services layer.

Underlying technologies

The bottom layer is a set of underlying technologies that enable a Cloud provider offering a plethora of possibilities to provide high availability using commodity systems.

Virtualization is not a new concept but Cloud providers use it as key technology for enabling infrastructure

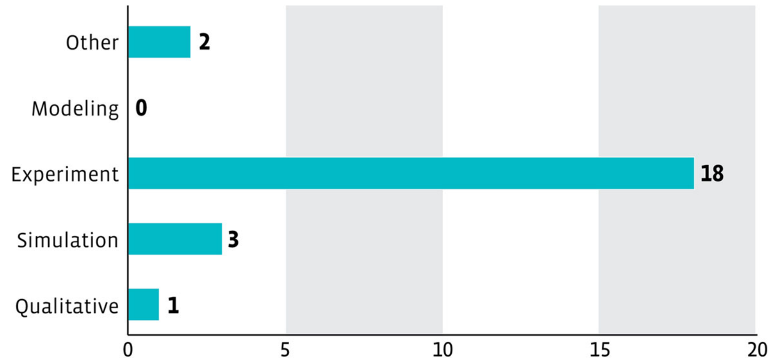

Fig. 5 Approaches used to evaluate HA solutions

Table 2 Metrics for HA evaluation from [8]

Metric	Definition
Reaction time	Delay between the occurrence of the failure and the first reaction of the availability management solution.
Repair time	Duration from the first reaction until the faulty entity is repaired.
Recovery time	Duration from the first reaction until the service is provided again
Outage time	Time between the failure happening and the service recovery. In other words, outage time is the amount of time the service is not provided and it is the sum of the reaction and recovery times.

operation and easy management. According to [11], the main factor that increased the adoption of server virtualization within Cloud Computing is the flexibility regarding reallocation of workloads across the physical resources offered by virtualization. Such flexibility allows, for instance, for Cloud providers to execute maintenance without stopping developers' applications (that are running on VMs) and to implement strategies for better resource usage through the migration of VMs. Also, server virtualization is adapted for the fast provisioning of new VMs through the use of templates, which enables providers to offer elasticity services for application developers [12].

Virtualization can also be used to implement HA mechanisms at the VM level, such as failure and attack isolation, checkpoint and rollback as recovery mechanisms. Beyond that, virtualization can also be used at the network level with the same objectives by virtualizing network functions (see about Network Function Virtualization (NFV) in [13]).

There are several hypervisor options, such as from the open-source community, Xen[8] and Kernel-based Virtual Machine (KVM)[9]. As well, there are those from proprietary solutions, including VMWare[10] and Microsoft's HyperV[11].

Services

The second layer is composed of many services that can be implemented and configured according to Cloud provider requirements or management decisions. For instance, if a provider has a checkpoint mechanism implemented in its infrastructure, it should configure the checkpoint service, which could mean setting it as an active or a passive checkpoint, and configuring the update frequency, for instance. The next subsections describe the main services and report how related studies used them.

Redundancy

The redundancy service can offer different levels of availability depending on the redundancy model, the redundancy strategy, and the redundancy scope (Fig. 8).

The redundancy model refers to the many different ways HA systems can combine active and standby replicas of hosted applications. AMF describes four models: 2N, N+M, Nway, and Nway active [14]. The 2N ensures one standby replica for each active application.

The N+M model is an extension of the 2N model and ensures that more than two system units (meaning a virtual machine, for instance) can handle taking active or standby assignments from an application. N represents the number of units able to handle active assignments and M represents those with standby assignments. It is important to notice that, considering the N+M model, a unit that handles active assignments will never handle standby assignments.

Furthermore, the N-way is similar to the N+M model with the difference that it allows in the N-way model

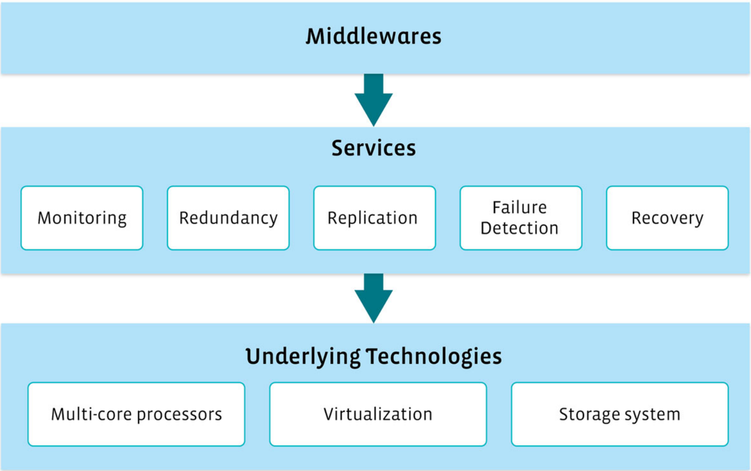

Fig. 6 3-layer classification for HA Cloud solutions

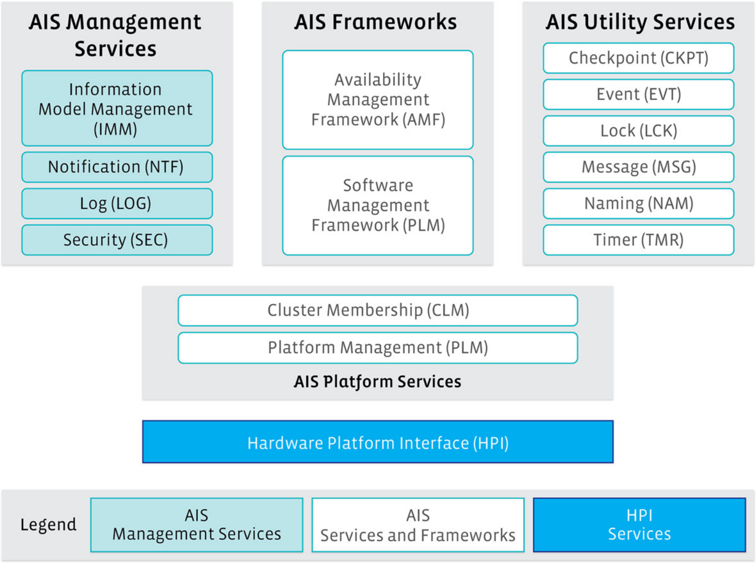

Fig. 7 Overview of SAF framework [10]

unit to handle both active and standby assignments from diverse applications instances.

Lastly, the N-way Active redundancy model comprehends only active assignments from unit applications; it does not allow standby assignments, but permits an application instance to be allocated as active into various units. Due to its simplicity, the 2N model is preferred in terms of implementation [15, 16].

The redundancy strategy is divided in two classes: active and passive redundancy [17]. In active strategy, there are no standby replicas and all application replicas work in parallel. When one node fails, tasks executing at the failed node can be resumed in any remaining node. In passive redundancy, there is one working replica whereas remaining replicas are standby. When the main node fails, any standby replica can resume failed node tasks. Please note that this active strategy helps to provide load balancing to applications. However, maintaining consistency in the passive model is simpler, and so this strategy is used in different proposals [15].

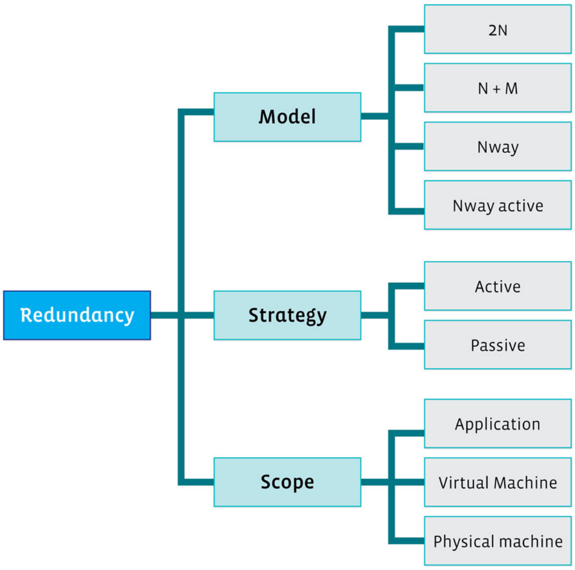

Fig. 8 Redundancy classification

In respect to scope, one can replicate the application itself, the VM that hosts the application, or the complete physical server hosting the application. Authors in [15] propose to use all these approaches in a model-based framework to select and configure High Availability mechanisms for a cloud application. The framework constructs a model of the running system and selects the proper HA services according to the benefits and costs of each service, as well as the required availability level. In contrast, the proposal described in [16] focuses on the VM scope only.

Data replication

Data replication is used to maintain state consistency between replicas. The main problem associated with this service is the question of how to govern the trade-off between consistency and resource usage [18]. In Clouds, the replication may be achieved either by copying the state of a system (checkpoint) or by replaying input to all replicas (lock-step based) [16] (see Fig. 9).

The lock-step strategy is also called "State Machine Replication" and its main goal is to send the same operations to be executed by all replicas of an application in a coordinated way, thus guaranteeing message order and

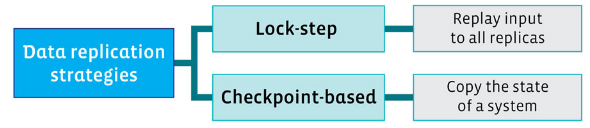
Data replication strategies
Lock-step
Replay input to all replicas
Checkpoint-based
Copy the state of a system

Fig. 9 Data replication classification

state. This strategy can be found in the TClouds plataform [19], which is applied to the state maintenance of application replicas and is also applied to maintain the consistency of objects stored in a set of cloud storage services. The same strategy is applied in the Cloud-Niagara middleware [20] in order to offer a monitoring service to check resource usage and send failure notifications with minimal delay. Following this same strategy, Perez-Sorrosal et al. [21] propose a multi-version database cache framework to support elastic replication of multi-tier stateless and statefull applications. In this framework, application and database tiers are installed at each replica and a multicast protocol maintains data consistency between replicas. The main focus of this proposal is elasticity, but the solution can also cope with failures since the replication protocol uses virtual synchrony to guarantee the reliable execution of the replicas.

Checkpoint-based replication involves propagating frequent updates of an active application to its standby replicas. It is desirable that an application have some checkpoint replicas distributed over different entities to increase reliability, guarding it against failures [10]. Checkpoint service can be implemented in a centralized fashion, when all checkpoint replicas are allocated to the same entity, and in a distributed one, where replicas are located in different entities of a cluster.

Remus is a production level solution implemented at Xen to offer High Availability following this strategy [22]. Authors of that solution point out that lock-step replication results in an unacceptable resource usage overhead because communication between applications must be accurately tracked and propagated to all replicas. In contrast, checkpoints between active and standby replicas occurs periodically, in intervals of milliseconds, providing better tradeoff between resource usage overhead and updates. Taking a similar approach, Chan and Chieu [23] introduce a cost effective solution which utilizes VM snapshots coupled with a smart, on-demand snapshot collection mechanism to provide an HA in the virtualization environment. The main idea behind this proposal is to extend the snapshot service (a common service offered by virtualized infrastructures) to include checkpoint data of a VM.

While Remus and similar approaches fit well to IaaS Clouds because they provide an application-agnostic VM-based checkpoint, Kanso and Lemieux [7] argue that in a PaaS Cloud the checkpoint service must be performed at the application level in order to cope with internal application failures that may remain unnoticed in a VM-based HA system. Therefore, the authors propose that each application send its current state to the HA system through a well-defined checkpoint interface.

In [24], authors propose BlobCR, a checkpoint framework for High Performance Computing (HPC) applications on IaaS. Their approach is directed at both application and process checkpoint levels through a distributed checkpoint repository.

In [16] authors present a solution focusing on HA for real-time applications. The middleware proposed is derived from others technologies, such as Remus, Xen and OpenNebula. For instance, continuous-checkpoint, in which asynchronous checkpoints are made in a security VM to provide HA in case of failures, was inherited from Remus.

Monitoring

Monitoring is a crucial service in an HA Cloud. Through this service, applications' health is continuously observed to support others services. The primary goal of this service is to detect when a replica is down, but robust implementations can also follow the health indicators of an application (CPU and memory utilization, disk, and network I/O, time to respond requests) which will help to detect when a replica is malfunctioning [17]. It can also be done at virtual and physical machine level (Fig. 10).

Papers surveyed showed there are two basic types of monitoring: push-based monitoring and polling-based monitoring. The latter is the most common type of monitoring and involves a set of measuring controllers periodically sending an echo-signal to the hosted applications. This check can be sent to the operating system that hosts the application (through standard network protocols like ICMP or SNMP) or directly to the application through a communication protocol, e.g., HTTP in the case of web applications [17].

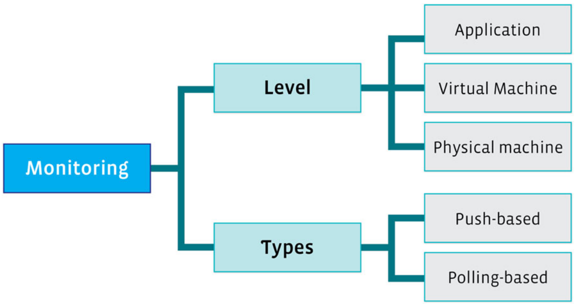

Fig. 10 Monitoring classification

Polling-based monitoring can also be sent from a backup replica to an active replica in order to check its status and to automatically convert it from backup to active when necessary [15] and [20]. This type of monitoring can be made by a monitoring agent that is external to the application or an agent can be implemented directly in the application by a standardized API that handles messages sent by the Cloud. Through this intrusive approach the internal state of the applications can be monitored, enabling the earlier detection of adverse conditions and making it possible to offer services such as checkpointing [7].

Push-based monitoring consists of the application (or a cloud monitoring agent deployed with the application) being the one responsible for sending messages to the measuring controller, when necessary. In this case, the controller is informed when a meaningful change occurs in the monitored application [25]. Push-based monitoring can also be implemented following a publish/subscribe communication model. This type of monitoring is employed by Behl et al. [26] to provide fault-tolerance to web service workflows. The fault monitoring is implemented through ZooKeeper's Watches, which are registered to check if a Zookeper's ephemeral node (an application in this case) is active. In the case of failure, the monitoring controller is notified about the crash. An et al. [16] point out that the highly dynamic environment of cloud computing requires timely decisions that can be achieved by publish/subscribe monitoring. In this case, the monitoring controllers are subscribers and the monitoring agents are publishers.

One important aspect to observe is that both approaches (push and poll) can be implemented in a Cloud environment. The high availability platform proposed by Chan and Chieu [23] uses polling to check periodically for host failures, and monitoring agents running in the hosts push notifications to the monitoring controller. An et al. [16] propose a hierarchical monitoring strategy combining the publish/subscribe communication model for global-level monitoring with polling at the local level.

Failure detection

Failure detection is an important service contained in most HA solutions, which aims to identify systems' faults (application, virtual or physical machine level) and provide needed information for services capable of treating problems to maintain service continuity (Fig. 11).

In [17] the authors list some mechanisms used to detect faults like ping, heartbeat and exceptions. From this perspective, failure detection can be classified in two categories according to detection mechanisms: reactive [23, 26]) and proactive [20]. The first approach waits for KEEP ALIVE messages, but it identifies a failure after a period of time waiting without any KEEP ALIVE message. The second approach is more robust and is capable of identifying abnormal behaviors in the environment, checking the monitoring service and interpreting collected data to verify whether there are failures or not.

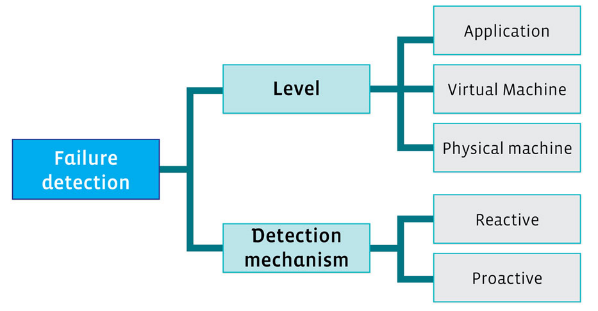

Fig. 11 Failure detection classification

For simplicity, the reactive type is implemented more often. The work presented in [26] proposes a fault-tolerant service through replication processes with BPEL implementation, which means that Zookeeper is responsible for detecting crashed replicas using a callback mechanism called watches. As well [23], authors treat failure detection through heartbeats hosted in each node, and so the absence of heartbeats after a period of time has passed indicates a failure and hence the recovery process begins.

Authors in [20] propose an intelligent system that depends on a proactive mechanism of monitoring and notification, as well as a mathematical model which is responsible for identifying the system faults.

Others studies lack many details about the failure detection process. For instance, in [27], failure detection is implemented together with failure mitigation (recovery) in a process called Fault Injection. This process aims to evaluate the framework capacity to handle failover possibilities. Also, in [7], authors proposed a HA middleware inside VMs for monitoring and restarting in case of failures.

In [16], authors proposed an architecture with an entity called LFM (Local Fault Manager), located in all physical host. It is responsible for collecting resource information such as memory, processes, etc. and transferring it to the next layer, which is responsible for decision making, similar to a monitoring service. Moreover, LFM also runs HAS (High-Availability Service) that keeps synchronization between primary and backup VMs, and is responsible for making backup VM active when a failure is detected in the primary VM.

Recovery

The recovery service is responsible for ensuring fault-tolerant performance through some services like redundancy [17], which means preserving HA even during crashes at application, virtual or physical machine level. It can be classified into smart [15, 16, 20] and simple [23, 28] (Fig. 12). The smart recovery uses other services and mechanisms (such as monitoring and checkpoint) to provide an efficient restoration with minimum losses for the application. Meanwhile, considering simple recovery, the broken application is just rebooted in a healthy node, so that the service continues to be provided, but all state data are lost.

The smart recovery proposed in [15] is guaranteed through a fault tolerant mechanism that keeps an application backup synchronized with active applications but deployed in a different VM. Authors in [16] work in a similar way, starting with the Remus project as base and applying a technique for VM failover using two VMs (primary and backup) that periodically synchronize states and are able to change from primary VM to backup, when needed. In [20], recovery is reached using an active replication technique, where a controller manages a priority list through Backup-ID from resources. Therefore, after a failure, broadcast communication is made and other nodes at the top of the list must assume the execution.

Furthermore, authors in [23] decided to use the simple recovery after a failure by using merged snapshots, in which faulty agent requires the manager any of snapshot available. In addition, work in [28] also uses simple recovery, in which the VMS are monitored by a VM wrapper that identifies unavailability and makes reboots.

Middleware

At the upper layer, we have middleware that uses services to provide HA to applications. The main goal is to manage how these services will operate, configure them, and take decisions according to information acquired.

OpenSAF [10] is an open source project that offers some services that implement the SAForum Application Interface Specification (AIS). For instance, OpenSAF implements the Availability Management Framework (AMF), which is the middleware responsible for maintaining service availability. Is also implements the checkpoint service (CPSv) that provides a means for processes to store checkpoint data incrementally, which can be used to protect applications against failures. For a detailed description of all SAF services implemented by OpenSAF, please see [10].

Since OpenSAF is used for general purpose, some studies use it to implement their Cloud solutions. For instance, authors in [7] propose an HA middleware for achieving HA at application level by using an SAF redundancy strategy. The middleware is responsible for monitoring physical and virtual resources, and repairing them or restarting VMs in case of failure. They also propose an HA integration. Basically, there is an integration-agent, which a Cloud user interacts with in order to provide information about its application and its availability requirements (such as number of replicas and redundancy model); and there is an HA-agent, which is responsible for managing the state of state-aware applications, and abstracting the complexity of APIs needed to execute the checkpoint service.

OpenStack[12] is an open source platform for public and private Clouds used to control large pools of computation, storage and networking resources. OpenStack has several components, and each component is responsible for a specific aspect of the Cloud environment. For instance, the component named Nova is responsible for handling VMs, and providing different flavors and images that describe details about the CPU, memory and storage of a VM. Another component is Neutron, which responsible for network management functions, such as the creation of networks, ports, routers and VMs connections. Considering the HA scope, we highlight the component called Heat that is OpenStack's orchestration tool. Using Heat, one can deploy multiple composite Cloud applications into OpenStack's infrastructure, using both the AWS CloudFormation template and the Heat Orchestration Template

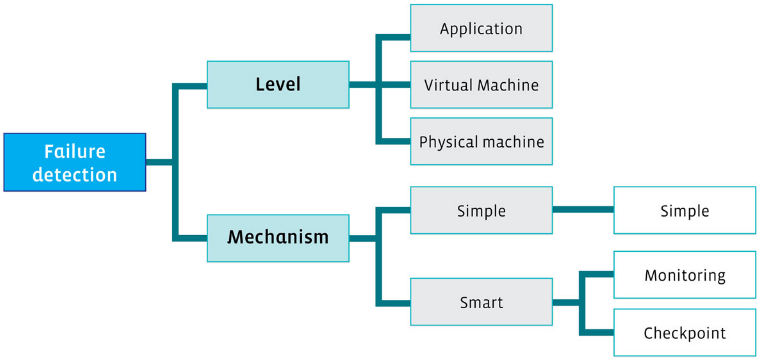

Fig. 12 Recovery classification

(HOT). In terms of HA, with Heat it is possible to monitor resources and applications from three basic levels[13]: 1) application level; 2) instance level; and 3) stack level (group of VMs). In case of failure, Heat tries to solve the problem in the current level. If the problem persists, it will try to solve it in a higher level. However, restarting resources can take up to a minute. Heat can also automatically increase or decrease the number of VMs, in conjunction with Celiometer (which is another OpenStack service) [25].

The paper [28] presents an OS-like virtualization cloud platform. They offers a dual stack API in the shell. One is called "Kumoi" and is used to manipulate data centers directly, while the other is called "Kali" and is used to build up the stack of cloud computing. With this cloud platform authors provide several HA services, such as checkpoint, monitoring, failure detection, recovery and elasticity. One should notice that services are provided at the VM level. They also present a qualitative evaluation between their tool and several others, such as Openstack, Nimbus, and OpenNebula.

The proposed solution in [20] is a high availability and fault tolerance middleware through the checkpoint, watchdog and log services for applications in a cloud environment. The authors claim that two issues are responsible for reaching middleware objectives: notifications without delay and monitoring of resources, which is achieved through an analytic model that identifies the fault nature. The Cloud-Niagara algorithm is shown and performs adjustments at nodes through resources calculation. The mean time to recover of the proposed solution is compared to other systems and evaluated on OpenStack, where Cloud-Niagara operates, by executing processes from real applications (PostgreSQL Database (DB), File Transfer Protocol (FTP), etc). This evaluation shows the CPU usage variation through different loads from the execution of applications processes execution, presenting the importance of monitoring the effective replica instantiation.

Discussions

In the previous sections, we presented a 3-layer classification for HA Cloud solutions that use many techniques to apply HA requirements at the infrastructure level. Since these technologies are key-enablers for Cloud operation and management, it is crucial that we go beyond the advantages to understand their specific challenges.

Regarding the underlying technologies, despite the fact that we presented virtualization as a good alternative for providing HA, some authors do not completely agree that this technology is a good solution for this purpose. In [7], the authors state that virtualization can hide some failures at the software level and that failures at the operating system level can affect both active and standby VMs if running in a lock-step way. Beyond that, virtualization introduces additional software layers imposing additional delays to network datagrams [29]. Consequently, performance measurements can also be affected by virtualization; authors in [29] show that clock-related measurements are affected by CPU load in the host as well as in the network load.

Regarding the offered services - the main focus of this work -, we can find several proposals in the literature for improving them, such as ([5, 30, 31]). Here, we highlight the issues surrounding automatic configuration and test of these services. As it was observed in “Cloud outages” section , Cloud outages can occur due to the misconfiguration of management services. Commonly, enterprises add validation checks for automatic configurations and improve mechanisms for detection and recovery of service failures.

Another important aspect is the feasibility of the service implementation. For instance, authors in [7] implemented their proposal; their algorithms run in polynomial time and the middleware consumes approximately 15MB of RAM and a moderated amount of CPU. On the other hand, Always On solution [32] proposes an HA architecture but does not provides insights on the feasibility of its implementation, nor does it treats how to deploy it. Beyond that, in this solution, applications need to implement their own HA mechanisms because they do not use a modular approach.

In terms of the middleware layer, its main shortcomings are its lack of compatibility to a standard specification and its dependency on a specific technology platform. These characteristics make these solutions inflexible, since once an application is developed to comply with such a middleware, the application cannot be migrated to other alternative solutions without major modifications. In this way, these solutions do not represent the desired interoperability (portability) requirement. The middleware presented in [7] overcomes this problem by offering HA at the application level by using an open-source and standardized implementation, named OpenSAF, which is a flexible and platform-independent solution.

Security is also an essential aspect for HA Clouds; however, none of the presented solutions deals with security mechanisms, such as those protecting against malicious attacks at the VM or application level. This occurs because detection and treatment of security breaches depends on different mechanisms. Even when an attack leads to a failure condition, dealing with this issue can propagate the consequences of the attack to the standby units. For example, in the case of a Denial-of-Service (DoS) attack, the middleware can proactively detect the active unit is out of service, failover to the standby unit, and transfer all requests to the standby unit. This strategy would propagate the denial of service to the standby unit. Therefore, the integration of HA and security services is an essential requirement when implementing a cloud middleware.

The advancement of standardization and improvement of HA strategies in a cloud computing system leads to the concept of HA-on-demand. Authors in [7] discuss this idea. They point out that not all applications need HA requirements during all the time. Thus, users can request HA services for their applications according to their current real needs. An online store can, for example, program different HA levels according to the chronogram of an announced promotion (e.g. Black Friday), changing the robustness of the system in respect to its calendar, clients demand, and allocated budget. In this way, authors state that it is feasible to have HA-as-a-service per applications in the Cloud.

NoPaas: proposal of a high available cloud for PaaS provisioning

Considering all of this related work, we have defined a set of requirements for implementing a high available framework to provide PaaS, which we named NoPaaS (Novel PaaS).

We grouped these requirements into two categories: a) application requirements, and b) framework requirements. **Application requirements** represent mandatory characteristics that all applications require in order to work properly within the NoPaaS framework (Table 3). In turn, **framework requirements** are a set of services and characteristics that the NoPaaS framework itself must provide for applications and/or developers (Table 4).

The application requirements are necessary to provide a unique interface to developers. In this way, the proposal provides strategies to allow developers to handle some HA resources provided by the NoPaaS. At the same time, applications need to be adapted in order to comply with all these requirements, as stated in REQ A.1, in which developers should use NoPaaS API to implement their applications. Despite application requirements making application development a little bit hard, this is a very common requirement in PaaS environments, such as Google App Engine[14]. Those PaaS cloud environments provide user APIs for building scalable web applications and mobile backends. Furthermore, REQ A.2 was defined in order to guarantee the multi-tier and stateful applications handling by the NoPaaS, and REQ A.3 was stated to facilitate and standardize the communication between application' tiers.

Table 3 Application requirements

REQ A.1	Must implement the API as described by the framework
REQ A.2	Must always include the session ID in messages exchanged between tiers of the application
REQ A.3	RESTful communications between tiers

Table 4 Framework requirements

REQ F.1	Must define the API for north bound communication with applications
REQ F.2	Must support different profile configurations
REQ F.3	Must plan resource allocation based on different profile configurations
REQ F.4	Must support multi-tier applications
REQ F.5	Must support stateful applications
REQ F.6	Must deal with sticky sessions
REQ F.7	Must assure HA
REQ F.8	Must provide scaling
REQ F.9	Must provide resource management
REQ F.10	Must rely on Cloud infrastructure compatible with current standards

The framework requirements were defined in order to achieve high availability focused on provisioning multi-tier stateful applications. REQ F.1 allows an unique form of communication with different types of applications, making this process simple for the developer. REQ F.2 and F.3 are related to profile configurations (economy, business, and custom) in order to incoporate different available budgets and requirements into response time and availability levels. These requirements facilitate the resource management from the PaaS provider perspective. From REQ F.4 to F.6, we have determined that the framework must deal with a specific type of application: multi-tier and stateful. It is our big distinction, since we did not find other studies considering such an application type. From REQ F.7 to F.9, we state the main services in order to ensure high availability. These are the big challenges for us, since we are considering multi-tier and stateful applications. The REQ F.10 guarantees compatibility with existing Cloud IaaS providers.

Considering all of these requirements, we propose our NoPaaS framework for high available clouds, shown in Fig. 13. The NoPaaS was designed to support the deployment of multi-tier and stateful applications deployment, providing services that include checkpoint, session migration, and failure recovery.

App deployment module

The App Deployment module is responsible for the interface between the application developer and our NoPaaS framework. NoPaaS proposes a set of modules, in which each module must act as a gateway between the PaaS service and NoPaaS internal services. Applications which will be deployed within NoPaaS must accomplish REQs A.1, A.2, and A.3 regarding the application requirements, and REQ F.1 regarding the framework requirements.

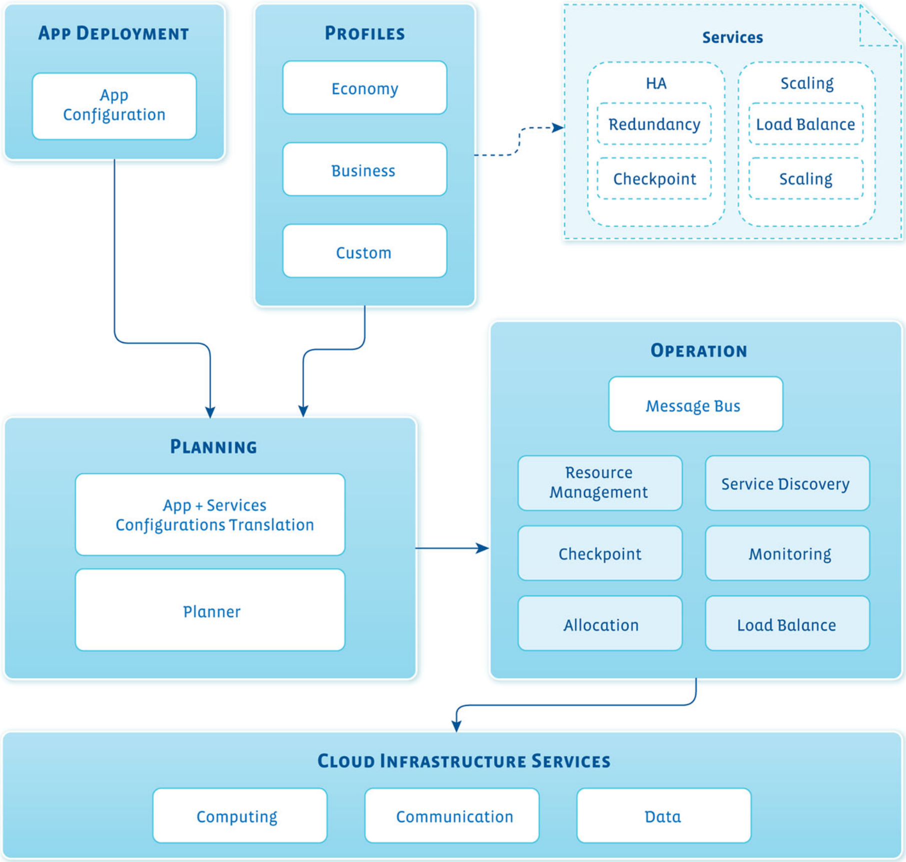

Fig. 13 NoPaaS framework

For the developers, it is mandatory to provide a configuration file specifying all information needed to deploy their applications in NoPaaS. Such a configuration file is very similar to what is usually provided to traditional PaaS in order to deploy a new application (e.g., git repository address and multi-tier architecture).

Profiles module

NoPaas makes use of profiles to represent and map the available budget provided by the developer and application requirements into response time and availability levels. The NoPaaS defines and provides, but it is not limited to, three different profiles: a) economy; b) business; and c) custom. For each profile, there is a specific configuration of load balance, scaling, checkpoint mechanism, and redundancy model based on the Service Availability Forum (SAForum or just SAF) model. REQs F.2 and F.3 are obeyed by this module.

NoPaaS uses the SAF reference because it produces open specifications to address the requirements of availability, reliability and dependability for a broad range of applications. In the SAF specification, there are five redundancy models: no redundancy, 2N, N+M, Nway, and Nway active. These redundancy models differ from each other in the number of active and standby assignments each service has [33], and consequently in terms of the availability level that each model is able to reach.

Planning module

The set of information provided by the developers regarding their applications' configurations and profiles is sent to the App + Services Configurations Translation in the Planning module, which is responsible for translating this information so it can be used by the Planner. The Planner analyses all requirements and available resources on the Cloud infrastructure and plans the resource allocation, choosing the SAF redundancy model in order to satisfy **REQs F.4, F.5, and F.6**. The Planner also communicates with the Resource Management (in the Operation module) in order to ensure information about resource availability is always updated.

The Planner is responsible for executing two main activities: calculating the availability estimation based on SAF redundancy models, and defining the application allocation by trying to minimize the total cost while reaching a minimum availability level defined by the developer.

Each tier of an application is named as Service Instance (SI), and each SI is assigned into a Service Unit (SU). We modeled the solution for mapping SI into SU as an integer program and solved it using algorithms to find the best SAF model (with minimum cost). The Planner uses an analytic worst-case models to estimate the availability of each SAF redundancy model. For a detailed explanation about the analytic models and some simulation results and analysis, please see [34].

Operation module

The Operation module provides many services to deal with the Cloud infrastructure. Resource Management is responsible for supervising the infrastructure, reporting on application failures and generating scaling in/out triggers. The Checkpoint stores backups of deployed applications, recovering their states in case of failure, and also deals with session migrations. The Allocation enforces the reservation of resources designed by the Planner. The Monitoring keeps track of applications and physical resources, maintaining a map of resource usage. The Load Balance is used to distribute the load among multiple tiers of an application, dealing with session stickiness, server failure, and session migration. We define the Message Bus entity for communication purposes, and it is responsible for receiving and delivering messages for all entities. **REQs F.7, F.8, and F.9** should be attended by services of this module.

For instance, we have the **resource management** that handles application failures and is also responsible for issuing alerts regarding scaling needs. **Monitoring** is a basic service responsible for monitoring all applications and (virtual and/or physical) resources. Data generated by the monitor entity is stored and used a posteriori to measure which resources are available and to calculate the ideal configuration needed to deploy a new application (or if scaling is needed).

Cloud infrastructure module

The Cloud Infrastructure services comprise the IaaS services that NoPaaS uses to allocate the developers' applications. The main idea is to use Cloud facilities in order to avoid unnecessary work. For that, NoPaaS needs to contract some IaaS provider or configure our own private IaaS. With this, we comply with **REQ F.10**.

Final considerations

Cloud outages, no matter how long, are responsible for large financial losses. Cloud providers look for solutions that provide high availability even in failure cases. In this paper, we proposed a classification for HA Cloud solutions based on 3 layers. We also described and discussed some existing commercial and non-commercial solutions focused on middlewares.

High availability is a great challenge for Cloud providers due to its complexity (from the infrastructure to the application level). There are many issues to study in order to minimize Clouds outages, such as portability, feasibility, and security. A next step could be the implementation of HA-as-a-service, highlighting even more the importance of this research area for Cloud providers.

Endnotes

[1] http://iwgcr.org/

[2] https://www.cloudfoundry.org

[3] https://blogs.dropbox.com/tech/2014/01/outage-post-mortem/

[4] http://www.verizonwireless.com/solutions-and-services/verizon-cloud/

[5] http://ieeexplore.ieee.org/Xplore/home.jsp

[6] http://www.sciencedirect.com/

[7] http://dl.acm.org/

[8] https://www.xenproject.org/

[9] http://www.linux-kvm.org/page/Main_Page

[10] http://www.vmware.com/

[11] https://www.microsoft.com/en-us/cloud-platform/virtualization

[12] http://docs.openstack.org/developer/heat/

[13] https://wiki.openstack.org/wiki/Heat/HA

[14] https://cloud.google.com/appengine/

Acknowledgements
This work was supported by the RLAM Innovation Center, Ericsson Telecomunicaçõs S.A., Brazil.

Authors' contributions
Our contribution is a systematic review regarding existing high availability solutions for Cloud Computing. We considered studies done from 2010 to 2016; and we provided an overview and description about them based on 3-layer classification. Furthermore, we proposed a framework for providing high availability services, and also presented requirements to deal with multi-tier and stateful applications. All authors read and approved the final manuscript.

Competing interests
Cloud computing, high availability, resource management.

Author details
[1]University of Pernambuco (UPE), BR 104 S/N, Caruaru, Brazil. [2]Federal University of Pernambuco, GPRT, Recife, Brazil. [3]Rural Federal University of Pernambuco, Recife, Brazil. [4]Ericsson Research, Kista, Sweden.

Received: 1 June 2016 Accepted: 4 October 2016
Published online: 18 October 2016

References

1. Dillon T, Wu C, Chang E (2010) Cloud computing: issues and challenges. In: Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference On. IEEE. pp 27–33. http://ieeexplore.ieee.org/document/5474674/?arnumber=5474674&tag=1

2. Puthal D, Sahoo B, Mishra S, Swain S (2015) Cloud computing features, issues, and challenges: a big picture. In: Computational Intelligence and Networks (CINE), 2015 International Conference On. IEEE. pp 116–123. http://ieeexplore.ieee.org/document/7053814/?arnumber=7053814
3. da Fonseca NL, Boutaba R (2015) Cloud Architectures, Networks, Services, and Management. In: Cloud Services, Networking, and Management. Wiley-IEEE Press. http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=7090261
4. Cérin C, Coti C, Delort P, Diaz F, Gagnaire M, Gaumer Q, Guillaume N, Lous J, Lubiarz S, Raffaelli J, et al. (2013) Downtime statistics of current cloud solutions. International Working Group on Cloud Computing Resiliency, Tech. Rep. http://iwgcr.org/wp-content/uploads/2013/06/IWGCR-Paris.Ranking-003.2-en.pdf. Accessed Oct 2016
5. Toeroe M, Tam F (2012) Service Availability: Principles and Practice. John Wiley & Sons. http://www.wiley.com/WileyCDA/WileyTitle/productCd-1119954088.html
6. Coutinho EF, de Carvalho Sousa FR, Rego PAL, Gomes DG, de Souza JN (2015) Elasticity in cloud computing: a survey. Ann. Telecommunications-annales des télécommunications 70(7–8):289–309
7. Kanso A, Lemieux Y (2013) Achieving high availability at the application level in the cloud. In: Cloud Computing (CLOUD), 2013 IEEE Sixth International Conference On. IEEE. pp 778–785. http://ieeexplore.ieee.org/document/6740222/?arnumber=6740222
8. Heidari P, Hormati M, Toeroe M, Al Ahmad Y, Khendek F (2015) Integrating open saf high availability solution with open stack. In: Services (SERVICES), 2015 IEEE World Congress On. IEEE. pp 229–236. http://ieeexplore.ieee.org/document/7196529/?arnumber=7196529
9. Hormati M, Khendek F, Toeroe M (2014) Towards an evaluation framework for availability solutions in the cloud. In: Software Reliability Engineering Workshops (ISSREW), 2014 IEEE International Symposium On. IEEE. pp 43–46. http://ieeexplore.ieee.org/document/6983798/?arnumber=6983798
10. OpenSAF Overview Release 4.4 Programmer's Reference. http://sourceforge.net/projects/opensaf/files/docs/opensaf-documentation-4.4.1.tar.gz/download. Accessed Oct 2016
11. Gonçalves GE, Endo PT, Cordeiro T, Palhares A, Sadok D, Kelner J, Melander B, Mangs J (2011) Resource allocation in clouds: concepts, tools and research challenges. XXIX SBRC-Gramado-RS. http://sbrc2011.facom.ufms.br/files/anais/shortcourses-12.html. http://sbrc2011.facom.ufms.br/files/anais/files/mc/mc5.pdf
12. Marshall P, Keahey K, Freeman T (2010) Elastic site: Using clouds to elastically extend site resources. In: Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing. IEEE Computer Society. pp 43–52. http://dl.acm.org/citation.cfm?id=1845214
13. Cui C, Xie Y, Gao G, Telekom D, Martiny K, Carapinha J, Telecom S, Lee D, Argela TT, Ergen M Network functions virtualisation (nfv). White paper, available at: https://portal.etsi.org/Portals/0/TBpages/NFV/Docs/NFV_White_Paper3.pdf. Accessed Oct 2016
14. Availability Management Framework. http://devel.opensaf.org/SAI-AIS-AMF-B.04.01.AL.pdf. Accessed Oct 2016
15. Wu Y, Huang G (2013) Model-based high availability configuration framework for cloud. In: Proceedings of the 2013 Middleware Doctoral Symposium. ACM. p 6. http://dl.acm.org/citation.cfm?id=2541595
16. An K, Shekhar S, Caglar F, Gokhale A, Sastry S (2014) A cloud middleware for assuring performance and high availability of soft real-time applications. J Syst Arch 60(9):757–769
17. Alexandrov T, Dimov A (2013) Software availability in the cloud. In: Proceedings of the 14th International Conference on Computer Systems and Technologies. ACM. pp 193–200. http://dl.acm.org/citation.cfm?id=2516814
18. Chen T, Bahsoon R, Tawil A-RH (2014) Scalable service-oriented replication with flexible consistency guarantee in the cloud. Inform Sci 264:349–370
19. Bessani A, Cutillo LA, Ramunno G, Schirmer N, Smiraglia P (2013) The tclouds platform: concept, architecture and instantiations. In: Proceedings of the 2nd International Workshop on Dependability Issues in Cloud Computing. ACM. p 1. http://dl.acm.org/citation.cfm?id=2506156
20. Imran A, Ul Gias A, Rahman R, Seal A, Rahman T, Ishraque F, Sakib K (2014) Cloud-niagara: A high availability and low overhead fault tolerance middleware for the cloud. In: Computer and Information Technology (ICCIT), 2013 16th International Conference On. IEEE. pp 271–276. http://ieeexplore.ieee.org/document/6997344/?arnumber=6997344
21. Perez-Sorrosal F, Patiño-Martinez M, Jimenez-Peris R, Kemme B (2011) Elastic si-cache: consistent and scalable caching in multi-tier architectures. VLDB J Int J Very Large Data Bases 20(6):841–865
22. Cully B, Lefebvre G, Meyer D, Feeley M, Hutchinson N, Warfield A (2008) Remus: High availability via asynchronous virtual machine replication. In: Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation, San Francisco. pp 161–174
23. Chan H, Chieu T (2012) An approach to high availability for cloud servers with snapshot mechanism. In: Proceedings of the Industrial Track of the 13th ACM/IFIP/USENIX International Middleware Conference. ACM. p 6. http://dl.acm.org/citation.cfm?id=2405152
24. Nicolae B, Cappello F (2013) Blobcr: Virtual disk based checkpoint-restart for hpc applications on iaas clouds. J Parallel Distrib Comput 73(5):698–711
25. Toeroe M, Pawar N, Khendek F (2014) Managing application level elasticity and availability. In: Network and Service Management (CNSM), 2014 10th International Conference On. IEEE. pp 348–351. http://ieeexplore.ieee.org/document/7014191/?arnumber=7014191
26. Behl J, Distler T, Heisig F, Kapitza R, Schunter M (2012) Providing fault-tolerant execution of web-service-based workflows within clouds. In: Proceedings of the 2nd International Workshop on Cloud Computing Platforms. ACM. p 7. http://dl.acm.org/citation.cfm?id=2168704
27. Ooi BY, Chan HY, Cheah YN (2012) Dynamic service placement and replication framework to enhance service availability using team formation algorithm. J Syst Softw 85(9):2048–2062
28. Sugiki A, Kato K (2011) An extensible cloud platform inspired by operating systems. In: Utility and Cloud Computing (UCC), 2011 Fourth IEEE International Conference On. IEEE. pp 306–311. http://ieeexplore.ieee.org/document/6123513/?arnumber=6123513
29. Dantas R, Sadok D, Flinta C, Johnsson A (2015) Kvm virtualization impact on active round-trip time measurements. In: Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium On. IEEE. pp 810–813. http://ieeexplore.ieee.org/document/7140382/?arnumber=7140382
30. Patel P, Bansal D, Yuan L, Murthy A, Greenberg A, Maltz DA, Kern R, Kumar H, Zikos M, Wu H, et al. (2013) Ananta: cloud scale load balancing. ACM SIGCOMM Comput Commun Rev 43(4):207–218
31. Singh D, Singh J, Chhabra A (2012) High availability of clouds: Failover strategies for cloud computing using integrated checkpointing algorithms. In: Communication Systems and Network Technologies (CSNT), 2012 International Conference On. IEEE. pp 698–703. http://ieeexplore.ieee.org/document/6200714/?arnumber=6200714
32. Anand M (2012) Always on: Architecture for high availability cloud applications. In: Cloud Computing in Emerging Markets (CCEM), 2012 IEEE International Conference On. IEEE. pp 1–5. http://ieeexplore.ieee.org/document/6354593/?arnumber=6354593
33. Availability Management Framework - Application Interface Specification SAI-AIS-AMF-B.04.01. Available at: http://devel.opensaf.org/SAI-AIS-AMF-B.04.01.AL.pdf. Accessed Oct 2016
34. Gonçalves G, Endo P, Rodrigues M, Sadok D, Curesco C Risk-based model for availability estimation of saf redundancy models. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7543848
35. Frîncu ME (2014) Scheduling highly available applications on cloud environments. Future Generation Comput Syst 32:138–153
36. Sripanidkulchai K, Sahu S, Ruan Y, Shaikh A, Dorai C (2010) Are clouds ready for large distributed applications? ACM SIGOPS Oper Syst Rev 44(2):18–23
37. Araujo J, Matos R, Alves V, Maciel P, Souza F, Trivedi KS, et al. (2014) Software aging in the eucalyptus cloud computing infrastructure: characterization and rejuvenation. ACM J Emerg Technol Comput Syst (JETC) 10(1):11