Fault Characterization and Mitigation Strategies in Desktop Cloud Systems

Carlos E. Gómez[1,2][0000−0002−5202−1167],
Jaime Chavarriaga[1][0000−0002−8372−667X], and
Harold E. Castro[1][0000−0002−7586−9419]

[1] Systems and Computing Engineering Department
Universidad de los Andes, Bogotá, Colombia
[2] Universidad del Quindío, Armenia, Colombia
{ce.gomez10,ja.chavarriaga908,hcastro}@uniandes.edu.co

Abstract. Desktop cloud platforms, such as UnaCloud and CernVM, run clusters of virtual machines taking advantage of idle resources on desktop computers. These platforms execute virtual machines along with the applications started by the users in those desktops. Unfortunately, although the use of computer resources is better, desktop user actions, such as turning off the computer or running certain applications may conflict with the virtual machines. Desktop clouds commonly run applications based on technologies such as Tensorflow or Hadoop that rely on master-slave architectures and are sensitive to failures in specific nodes. To support these new types of applications, it is important to understand which failures may interrupt the execution of these clusters, what faults may cause these errors and which strategies can be used to mitigate or tolerate these. Using the UnaCloud platform as a case study, this paper presents an analysis of (1) the failures that may occur in desktop clouds and (2) the mitigation strategies available to improve dependability.

Keywords: Desktop clouds, dependability, reliability, fault analysis, fault tolerance

1 Introduction

Volunteer computing platforms [12], desktop grid systems [8], and desktop clouds (DC) [1] demonstrate a lack of dependability and fault tolerance[1][4]. Different from other platform types using dedicated infrastructures, these platforms offer opportunistic services, taking advantage of unused computational capacities in desktop computers. Such platforms use software agents that detect inactive or idle desktop resources, and then execute several tasks and applications on those [8][12]. Unfortunately, due to the concurrent presence of users on the same desktop computers, these applications could stop or be affected if

This work has been partially carried out with resources provided by the CYTED cofunded Thematic Network RICAP (517RT0529).

users execute other applications, or worse, if they turn off the equipment. Consequently, opportunistic computing platforms are subject to failures that do not exist in other platforms.

Volunteer computing and desktop grid systems have successfully been used in executing Bag-of-Tasks (BoT) applications, a design pattern where processing is organized in such a way to divide the problem into smaller tasks that can be executed in parallel [3]. In such applications, tasks can be reallocated to another desktop computer in case of a failure in the platform nodes, thereby balancing dependability limitations.

More recently, DC platforms such as UnaCloud[3] [11] and CernVM[4][13] have allowed scientists and researchers to create clusters of virtual machines (VMs) taking advantage of idle resources in desktops spread throughout a campus. Such virtual clusters are able to execute certain applications with architectures different from a BoT. Clusters running applications such as Tensorflow[5] or Hadoop[6] use *master-slave* schemes, where only a few master nodes have control of the functions of many slaves. Although such schemes do tolerate certain faults in slave nodes, they have little tolerance for faults in masters. For such scenarios, if a dependable service is to be offered, failures must be detected, their causes determined, and mitigation procedures must be established.

Like in any cloud computing service provider, offering dependable service in DC platforms has become more important and appealing [7]. Currently, platforms such as UnaCloud offer VMs in a *best effort* service. Although UnaCloud is able to detect and notify a user about failures that occur during VM deployment and execution, users or applications are in charge of making decisions about those failures. For instance, although the UnaCloud platform is able to report that one or more cluster nodes have a fault, the user actually makes the choice to continue with the cluster execution with the failures or to restart the execution in a different desktop group. UnaCloud does not offer automatic services that increase dependability levels.

This paper presents a characterization of the faults that could occur in DC platforms, based on our analysis of the UnaCloud platform. Here, we analyze failures that could occur during normal operation, the detected failure states, and their possible causes. Finally, we present strategies that allow mitigation and tolerance of these faults.

This paper is structured as follows: Section 2 defines terms related to dependability. Section 3 describes UnaCloud and its architecture and VM deployment. Section 4 presents an analysis of failure states in UnaCloud and possible causes and some mitigation strategies for identified failure states. Section 5 concludes this paper and points toward future research.

[3] `https://sistemasproyectos.uniandes.edu.co/iniciativas/unacloud/`
[4] `https://cernvm.cern.ch/portal/publications`
[5] `https://www.tensorflow.org/`
[6] `https://hadoop.apache.org/`

2 Background

In this section, basic concepts related to dependability are presented. We discuss threats and the means to achieve dependability in computing systems design.

Definition. The concept of *dependability* arose in the 1950s when computers were made with components that tended to fail. Researchers wanted to determine if systems could depend on those components [10].

In one of the first definitions, Laprie [9] defined dependability as "the trustworthiness of a computer system such that reliance can justifiably be placed on the service it delivers". Considering that justification types vary according to user approaches, Prasad et al. [10] observed the difficulty in measuring dependability from a pre-established attribute set. They posited that dependability arises from knowing the faults that could occur in a system and establishing mechanisms for measurement, assessment, forecast and control. As a result of many IEEE initiatives, Avizienis et al. [2], defined dependability as: (1) the ability of a system to deliver a service that can be trusted and (2) the ability of that system to avoid failures that would be more frequent and severe than acceptable.

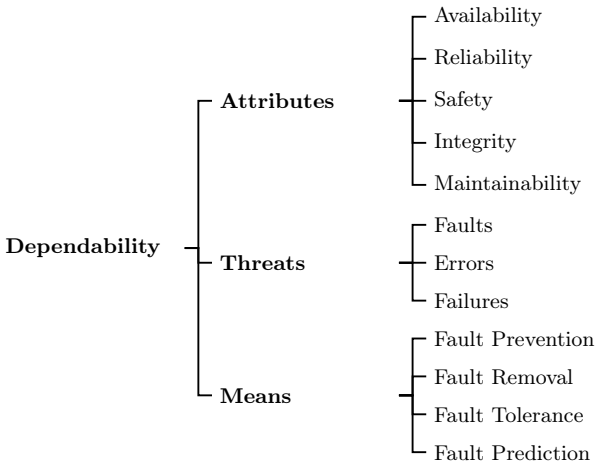


Fig. 1. Dependability concepts

Figure 1 summarizes dependability related concepts as proposed by Avizienis et al. [2]: (1) *attributes* used to analyze dependability, (2) *threats* that could impede dependability, and (3) *means* to achieve it.

Dependability Attributes. In order to study the dependability of a computing system, it is necessary to define the attributes to be analyzed. Since authors have proposed varied attributes [5][6][9][10], IEEE has suggested that dependability has five attributes [2]:

Availability: Capacity of a computing system to provide correct services when requests occur. It is the probability of a system's readiness to deliver a service.
Reliability: Probability of a system offering the correct services within a particular timeframe.
Safety: Absence of catastrophic consequences for users of a computing system, their data, and their surroundings whenever the computing service is offered.
Integrity: Ability of a system to prevent inappropriate alterations on itself.
Maintainability: Ability to make modifications and repairs to a system.

Threats against dependability. Avizienis et al. [2] refer to faults, errors, and failures as threats against dependability. They state that those three elements lead the computing system to deliver the wrong services or to be unable to deliver a service.

Fault: Defect in a computing system leading to an error.
Error: Manifestation of a fault in the system that could lead to a failure.
Failure: Event in the system that occurs when the service delivered is not correct, i.e. when the service does not comply with specifications or produces not-specified outputs.

Dependability threats define a sequence of events: Faults in a system and/or the components of a system could cause functioning errors, and these errors could lead to a failure to render services.

Means to achieve dependability. Many authors has proposed several ways to achieve dependability. For instance, Aviziens et al. [2] identifies four types of techniques: fault prevention, fault removal, fault tolerance, and fault prediction.

Fault prevention: Techniques applied during system design and development to prevent fault occurrence. These techniques include thorough *specification inspection*, *system simulations*, and use of *verification tools*.
Fault removal: Techniques aimed at deleting or reducing the number of faults that appear during system operation. These techniques involve *diagnostic activities*, *system modifications*, and *tests on changes that have been made*.
Fault tolerance: Techniques to assure that the computing system continues to function correctly despite the presence of faults. Usually, this can be done by using some type of *redundancy*. When a fault arises, the system can: (1) detect it, (2) locate the component where the fault is present, and (3) isolate this component. Once the faulty component is identified, a recovery is executed in one of two ways: the system may be *re-configured*, disabling the component, or perform a *graceful degradation*, rendering system services but with decreased capabilities.
Fault prediction: Techniques that aim to estimate faults as they arise during system operation and predict their possible occurrences and consequences. Normally, these techniques are used to assess the system or possible damages to it. Assessments are *qualitative* or *quantitative*.

3 UnaCloud

3.1 Overview

UnaCloud is a DC [11] that uses idle resources from desktops in an educational institution's computer room in order to provide IaaS. UnaCloud is currently used at Universidad de los Andes[7] to support research projects and PhD theses in areas such as Civil and Chemical Engineering, Image Processing, Bio-informatics, and Data Mining.

The UnaCloud platform aims for the lowest possible impact on tasks developed by desktop users. Therefore, this platform executes VMs with low priority and in the *background* on par with the users' applications. UnaCloud is normally used to execute BoT-type applications. However, UnaCloud has recently been used to run distributed apps for Bio-informatics and Data Mining that work under other schemes. These applications could require clusters where nodes coordinate assignments among themselves and require constant inter-communication.

In UnaCloud, we identify three main user types:

1. *Cloud User:* A person that uses the computing capacities offered by UnaCloud. A user is able to create and deploy VM clusters. Generally, cloud users are researchers without much knowledge in virtualization and distributed computing; as such, they require assistance.
2. *UnaCloud Administrator:* The person that manages UnaCloud. This person tracks laboratory information, hosts, repositories and deployments. The person manages users and performs configuration and monitoring tasks on the system.
3. *Desktop Users:* People, typically students, that are using a desktop that is part of UnaCloud. Although they are not platform users, their applications have a higher priority than VMs. Actions in their own desktop computers could affect the functioning of UnaCloud.

UnaCloud operation is built on three fundamental concepts:

1. *Virtual image:* A set of files that form a previously configured VM for execution by a hypervisor. It corresponds to the configuration files of a VM and to the files with disk images. For example, if Oracle VirtualBox hypervisor is used, the virtual image corresponds to *.vbox* and *.vdi* files. A completely functional VM can be created from a virtual image, provided that the cloud user complies with UnaCloud's required conditions for execution.
2. *Cluster:* A set of virtual images that a cloud user establishes to be used in one deployment on UnaCloud. For example, if a system formed by one *master* and ten *workers* is to be executed, the cluster is created with two virtual images: (1), the master and (2), the workers. These images are used as templates to create VMs and must be carefully prepared for them to be executed correctly on UnaCloud.

[7] `http://www.uniandes.edu.co`

3. *Deployment:* A request made by a cloud user to execute a cluster on UnaCloud. In this case, the user must specify how many instances of each virtual image are to be executed and define VM specifications in terms of processing cores and memory. A deployment is then materialized in the execution VMs on physical machines as allocated or appointed.

UnaCloud is managed through a web application through which users, according to their profile, make the requests. On the server side are the components that manage the business rules to make decisions about resource allocation (VMs to desktops), and the virtual images. On the client side, the entire operation is carried out through an Internet browser. On the desktop side, an agent is required in every desktop where UnaCloud executes the VMs. The agent receives commands from the UnaCloud server and translate them into commands that are understood by the hypervisor to create, configure, turn on, and turn off VMs, among others. Finally, the monitor is in charge of retrieving information on the resource consumption in a computer.

3.2 Virtual Machines Deployment

A deployment of a VM in UnaCloud starts with a request from the cloud user. When the UnaCloud server processes the request, the execution of each VM is controlled through a state machine.

After requesting deployment, the tasks performed for each VM can be classified into two categories or phases: *Preparation* and *Execution*.

Preparation. In the Preparation phase (§ Figure 2), a VM goes through the following states: REQUESTED, TRANSMITTING, CONFIGURING and DEPLOYING. Under normal operating conditions, at the end of the preparation phase, the VM enters into the DEPLOYED state and then enters into the Execution phase.

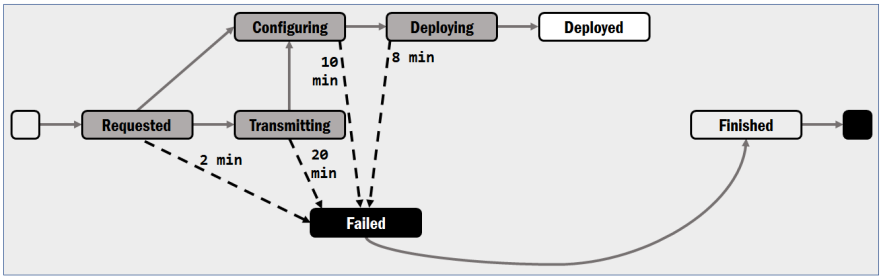


Fig. 2. States of a deployment in UnaCloud – Preparation phase

In each of the possible states a VM can be in during the deployment, a set of tasks must be performed within a given time. If the tasks are not carried out

within that time, a timeout is triggered and as a consequence, the VM enters a FAILED state. The VM then goes to a FINISHED state and is turned off.

As can be seen in Figure 2, the time allocated before moving to Failed state depends on the activity to be performed. For instance, while the timeout for the TRANSMITTING states is 20 minutes, the timeout for DEPLOYING is only 8 minutes.

In the Preparation phase, a deployment can fall into FAILED state for different reasons. For example, an image may be poorly configured, which is why it does not start execution; the network may have operating problems, so the virtual image cannot be copied to the computer where the VM will be deployed; the UnaCloud server cannot communicate with the agent on one or more computers; or the desktop user turns off the computer, restarts it, or executes demanding processes, so the VM is unable to execute as expected due to lack of resources.

Execution. On the other hand, in the Execution phase of a VM the following states can be found: DEPLOYED, RECONNECTING, FINISHING, and FINISHED (§ Figure 3).

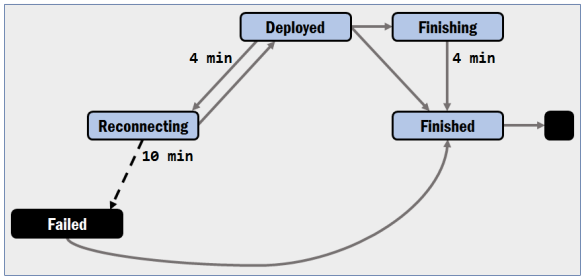


Fig. 3. States of a deployment in UnaCloud – Execution phase

In the Execution phase, a temporary fault may occur. These are generally associated with the loss of communication between the UnaCloud server and the agent running on a computer. If the communication breakdown persists for four minutes, it goes into the RECONNECTING state. It can stay like this for up to 10 minutes, at which point the VM goes to FAILED, then to the FINISHED state and is turned off.

4 Analysis of failure states and their possible causes

Like any computer system, UnaCloud can experience failures at any time. The cause of the failure cannot always be determined since different errors can cause the same failure in the service, and several faults can lead to the same error.

For example, a failure in the service can occur when the UnaCloud agent on the computer is inaccessible This can be caused because: (1) the desktop user

turned off the computer; (2) the desktop user restarted the computer; (3) the network cable of the computer was disconnected; (4) a maintenance policy in the computer room was applied and, as a result, the computer was restarted; or (5) there was a change in the configuration of the network. As these examples show, the same failure in the service can be due to faults caused by different actors including the user of the desktop or the administrator of the computer room.

It is important to note that a failure in the service can have different consequences depending on the state of deployment the VM is in. For example, if the UnaCloud agent on the computer is not accessible and the deployment is in the INITIAL state, the deployment cannot be requested; however, if the VM is in the REQUESTED state, the files that make up the virtual image cannot be requested.

This section presents our analysis of failures in the UnaCloud service for which following set of assumptions were taken into account:

1. The cloud user knows the UnaCloud platform and can use it correctly through the web interface.
2. The UnaCloud software has no development flaws.
3. UnaCloud services on the server side are not affected in their execution due to failures in the service of its infrastructure.
4. The software that the cloud user executes on the MVs has no defects nor it is time sensitive.
5. The hypervisors are installed correctly on the physical machines that are part of UnaCloud.
6. The UnaCloud administrator does its work without affecting its normal operation, despite its privileges.
7. Hardware defects and natural disasters are less frequent and will not be considered.

Below we present our analysis of four failures in the service when: (1) The UnaCloud agent on the computer is not accessible. (2) The UnaCloud agent on the computer is accessible, but the virtual machine is not. (3) The hypervisor cannot execute the boot configuration task of the virtual machine. (4) The virtual image cannot be copied to the desktop.

For each failure, we consider the causes (errors and faults) and the failure consequences, as well as the possible mitigation strategies, as shown in Figure 4.

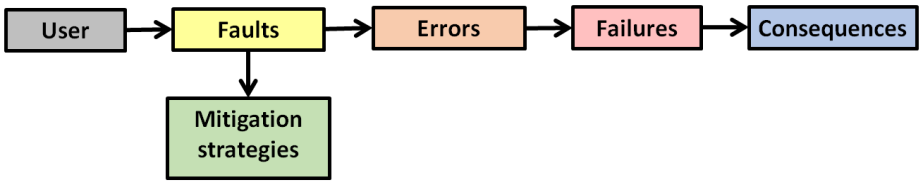


Fig. 4. Faults Propagation Chain

4.1 F1: The UnaCloud agent on the computer is not accessible

During UnaCloud execution, the agent located on each desktop may no longer be accessible. When this occurs, the system is unable to communicate with the agent, obtain information about processes running on the desktop, or send instructions to the hypervisor to control the VM.

This failure in the service occurs when there is a connection error with the agent. Table 1 shows the user that can cause a fault as well as mitigation strategies.

Error: e1 – Error connecting to the agent		
User	**Faults**	**Mitigation Strategy**
u1: Desktop User	**f1:** Shut down the machine.	**m1:** Save the execution context.
	f2: Restarted the machine.	
	f3: Disconnected the network cable from the machine.	**m2:** Wait. It is temporary.
u2: Computer Lab Administrator	**f4:** Restarted the machine for scheduled maintenance.	**m1:** Save the execution Context
	f5: Changed the network configuration.	If it is temporary, **m2:** Wait. Otherwise, **m3:** Migrate to other host.

Table 1. Causes and mitigation strategies of the failure in the service F1

The consequences of this failure depend on the execution state of the deployment. For example, when the VM is in the INITIAL state and has been assigned to the computer where the agent is not accessible, then the system cannot start deployment. Table 2 summarizes the consequences identified for this failure in the service.

State	**Consequence**
INITIAL	**c1:** Deployment task failure.
REQUESTED	**c2:** Files from the VM image cannot be requested.
TRANSMITTING	**c3:** Transmission can not be completed.
DEPLOYED	**c4:** The UnaCloud server can not receive the VM status report
RECONNECTING	

Table 2. States in which failure in the service F1 occurs and its consequences.

4.2 F2: The UnaCloud agent on the computer is accessible, but the virtual machine is not

During execution, it is possible that UnaCloud could not determine the execution state of some VMs despite it is able to communicate with the agent located in the corresponding desktops.

This failure in the service is due to a connection error with the VM. Table 3 shows possible faults, the user who generates them, and their corresponding mitigation strategies.

Error: e2 – Error connecting to the hypervisor or the virtual machine		
User	**Faults**	**Mitigation Strategy**
u2: Computer Lab Administrator	**f4:** Restarted the machine for scheduled maintenance.	**m1:** Save Execution Context
	f5: Changed the network configuration.	If it is temporary, **m2:** Wait. Otherwise, **m3:** Migrate to other host.
u3: Hypervisor	**f6:** Error in the hypervisor. The hypervisor is blocked.	If it is temporary, **m2:** Wait. Otherwise, **m1:** Save execution context.
	f7: Error in the application to communicate with the hypervisor.	**m3:** Migrate to other host.

Table 3. Causes and mitigation strategies of the failure in the service F2

The consequences of this failure in the service depend completely on the specific application that the cloud user is executing on the DC (§ Tabla 4).

State	**Consequence**
DEPLOYED	**c5:** Not determined. It depends on the cloud user's application.

Table 4. States in which failure in the service F2 occurs and its consequences.

4.3 F3: The hypervisor cannot execute the boot configuration task of the virtual machine

When a cloud user wants to run a cluster deployment in UnaCloud, the virtual images required must have been supplied. In this part of the process, the user intervention is key for the execution of deployment of the VM to succeed. When it is not possible to execute the configuration that allows the VM to start, possible causes for this have been identified. These are summarized in Table 5 along with the mitigation strategies.

Naturally, if the configuration task cannot be carried out, the VM cannot be configured and its execution will not start (§ Table 6).

Error: e3 – Unable to login to execute the configuration script		
User	**Faults**	**Mitigation Strategy**
u3: Hypervisor	**f8:** The image of the VM does not have the required complements for the configuration.	**m4:** Facilitate the cloud user (u4) the creation of their virtual images.
u4: Cloud user	**f9:** The VM image does not have software installed for the configuration procedure (E.g., remote access).	
	f10: The operating system installed in the VM is not compatible with the configuration procedure.	
	f11: The VM has a configuration that is incompatible with the configuration procedure (E.g., the type of network).	
Error: e4 – The configuration of the VM is incompatible.		
User	**Faults**	**Mitigation Strategy**
u4: Cloud user	**f12:** The password provided when uploading the image is not correct.	**m5:** Educate the cloud user (u4) the creation of their virtual images.
	f13: The image of the VM does not have a root user with a valid password to execute the configuration procedure.	

Table 5. Causes and mitigation strategies of the failure in the service F3

State	**Consequence**
CONFIGURING	**c6:** The VM cannot be configured.

Table 6. States in which failure in the service F3 occurs and its consequences.

4.4 F4: The image cannot be copied to the desktop

When requesting a deployment by the cloud user, it is possible that the files composing the virtual image required cannot be copied to the assigned desktop. When this occurs, the system is unable to create the VM.

This failure in the service F4 can occur either due to insufficient space in the computer assigned to receive the files, or due to the network's inability to complete the transmission. Table 7 shows the user that can generate the fault and its mitigation strategy.

Error: e5 – The hard drive is full.		
User	**Faults**	**Mitigation Strategy**
u5: UnaCloud development team	**f14:** The resource allocation algorithm assigned a machine without sufficient disk space.	**m6:** Design new resource allocation algorithms based on disk space monitoring information.
Error: e6 – Error during the transmission of a file.		
User	**Faults**	**Mitigation Strategy**
u6: Data network	**f15:** Congestion in the network and transmission fails.	If it is temporary, **m2:** Wait.

Table 7. Causes and mitigation strategies of the failure in the service F4

As a consequence of this failure, the transmission cannot be completed and the VM cannot be created. See Table 8.

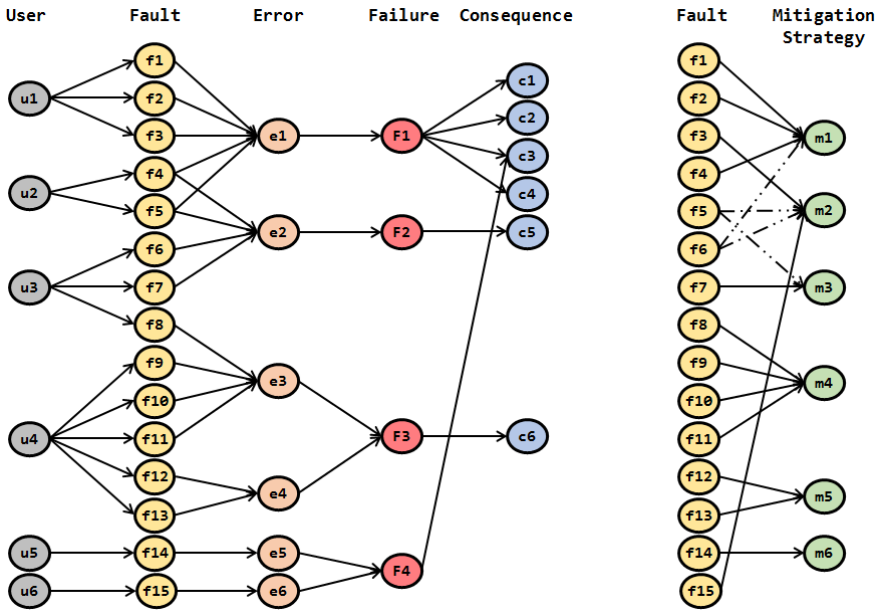
State	**Consequence**
TRANSMITTING	**c3:** Transmission can not be completed.

Table 8. States in which failure in the service F4 occurs and its consequences.

4.5 Summary

Figure 5 (a) shows the DC fault propagation including the user who causes the fault, the error, the failure in the service and its consequences. We note that the desktop user (user u1) and computer lab administrator (user u2) have much influence on faults despite not being cloud users. In addition, several faults produces the same error. For example, faults f1, f2, f3, f4 and f5 produces the error e1, and more than one error can cause the same failure. In Figure 5 (a), errors e3 and e4 cause the failure F3. Likewise, a failure can have several consequences. Failure F1 has four consequences, depending on the state of the VM deployment.

The analyzed failures can be mitigated using different strategies, e.g. saving the state of the execution of the system; migrating the compromised virtual machines to other desktops; facilitating the user of the cloud the creation of their virtual images, e.g. providing a catalog of images of VM to facilitate access to DC services; and educating the cloud user in the preparation of the VM so that they can be executed in UnaCloud, or waiting if the fault is temporary. Figure 5 (b) includes the six mitigation strategies that we propose for the 15 faults identified. The most relevant mitigation strategies are m1 (save the execution context) and m4 (facilitate the creation of the virtual images). It is important to highlight this analysis allows us to direct our efforts towards a solution that allows to save the state of an execution, along with to offer a easier way to use the platform for the cloud users, to improve the dependability.



(a) fault propagation (b) mitigation strategies

Fig. 5. Desktop Cloud fault propagation and mitigation strategies

5 Conclusions and future work

This article presents a characterization of the main failures in the service presented in desktop cloud systems using UnaCloud as a case study. This characterization describes the possible causes (faults and errors) and mitigation strategies to improve the dependability of the services offered to the cloud user. The failures identified are: the lack of access to computers and virtual machines; the inability to configure virtual machines; and the inability to complete the transfer of files with virtual images. Although failures in the service in the Preparation phase can be more frequent due to the cloud user and the desktop user intervention, the failures in service during Execution are of greater concern because completed work can be lost. The analysis carried out in this work can be applied to other cloud platforms in order to help improve dependability in the service. As future work, we plan to make new analyses, reducing the set of assumptions mentioned in Section 4. In addition, the possibility of integrating a solution of global snapshot to save the state of the system and implement new functions in the existing monitoring system have been considered, thus using this information in the allocation of virtual machines to computers.

References

1. Alwabel, A., Walters, R., Wills, G.: A view at desktop clouds. In: International Workshop on Emerging Software as a Service and Analytics (ESaaSA 2014). pp. 55–61 (2014)
2. Avizienis, A., Laprie, J.C., Randell, B., Landwehr, C.: Basic concepts and taxonomy of dependable and secure computing. IEEE Transactions on Dependable and Secure Computing **1**(1), 11–33 (2004)
3. Bakken, D.E., Schlichting, R.D.: Tolerating failures in the bag-of-tasks programming paradigm. In: 21th International Symposium on Fault-Tolerant Computing, FTCS-21. pp. 248–255. IEEE (1991)
4. Cunsolo, V., Distefano, S., Puliafito, A., Scarpa, M.: Volunteer computing and desktop cloud: The Cloud@Home paradigm. In: 8th IEEE International Symposium on Network Computing and Applications, NCA 2009. pp. 134–139 (2009)
5. Jonsson, E.: An integrated framework for security and dependability. In: The 1998 Workshop on New Security Paradigms, NSPW '98. pp. 22–29 (1998)
6. Jonsson, E.: Towards an integrated conceptual model of security and dependability. In: Availability, Reliability and Security, 2006. ARES 2006. The First International Conference on. pp. 8–pp. IEEE (2006)
7. Kangarlou-Haghighi, A.: Improving the reliability and performance of virtual cloud infrastructures. Ph.D. thesis, Purdue University (2011)
8. Kondo, D.: Scheduling task parallel applications for rapid turnaround on desktop grids. Ph.D. thesis, University of California, San Diego (2005)
9. Laprie, J.C.: Dependability: Basic Concepts and Terminology. Springer (1992)
10. Prasad, D., McDermid, J., Wand, I.: Dependability terminology: similarities and differences. In: 10th Annual Conference on Computer Assurance, COMPASS'95. pp. 213–221. IEEE (1995)
11. Rosales, E., Castro, H., Villamizar, M.: UnaCloud: opportunistic cloud computing infrastructure as a service. Cloud Computing pp. 187–194 (2011)
12. Sarmenta, L.F.G.: Volunteer computing. Ph.D. thesis, Massachusetts Institute of Technology (2001)
13. Segal, B., Aguado Sanchez, C., Buncic, P., Rantala, J., Mato, P., Blomer, J., Quintas, D.G., Weir, D.J., Yao, Y., Harutyunyan, A.: LHC cloud computing with CernVM. PoS p. 004 (2010)