Towards a Taxonomy of Cloud Recovery Strategies

Min Fu, Anna Liu, Len Bass
Software Systems Research Group, NICTA, Sydney, Australia
School of Computer Science and Engineering, University of New South Wales, Sydney, Australia

Abstract—Dependability of cloud can be achieved by enabling cloud recoverability. Recovery for cloud from consumer’s perspective is challenging because cloud platforms only provide consumers with limited visibility and control. One research on consumer-initiated cloud recovery is recovery for the errors occurring during sporadic operations on cloud applications. This research relies heavily on the existing consumer-initiated cloud recovery methods. Hence, in this paper we provide the taxonomy for the existing consumer-initiated cloud recovery methods to facilitate this research. Existing consumer-initiated cloud recovery methods are serving for different purposes. For example, some of them are intended for recovery of cloud applications normal activities such as application workflow and some of them are designed for recovery of cloud applications sporadic activities such as sporadic operations on cloud applications. Meanwhile, existing cloud recovery methods are applicable to different life circle phases. Some of them are used to design a recovery strategy during design phase and some of them are used to trigger recovery during runtime phase. Based on a review on the existing recovery strategies, we classify all the methods into four groups according to their serving purposes and applicable phases. Not only does this taxonomy facilitate the research on recoverability of cloud sporadic operations but also it can help better understand the existing cloud recovery strategies.

Keywords—cloud recovery; consumer-initiated; taxonomy;

I. INTRODUCTION

Recoverability for cloud has become a heavy research focus for several years. From a cloud consumer’s perspective, doing recovery is challenging because cloud platforms only provide consumers with very limited visibility and control[1]. For example, in AWS consumers are not allowed to explicitly assign a specified IP address to an instance. One of the researches on cloud recovery is on the recoverability of cloud sporadic operations[2], which should recover from both operational and application errors occurring during the cloud sporadic operations. According to industry best practice, the recovery feature for cloud applications is turned off during the sporadic operations on them. Hence, the dependability of cloud applications will be decreased when operators conduct sporadic operations on them. However, existing recovery strategies for cloud sporadic operations do not consider application errors that happen during cloud sporadic operations. But a good recovery mechanism for cloud sporadic operations should not only cater for the operational errors but also cater for the application errors as well. As such, this type of recovery mechanism may integrate and combine a wide range of cloud recovery strategies. Considering that the various cloud recovery strategies can be leveraged and even refined for proposing better recovery mechanisms for cloud sporadic operations, those various existing cloud recovery strategies really need to be analyzed in advance.

Some of the cloud recovery methods serve for cloud normal activities[3] (e.g. cloud applications workflow). For example, virtual machine replication mechanism[4] can be employed to recover from errors in cloud applications by switching from an erroneous machine to its backup one. This mechanism takes effects during the runtime of cloud applications and it is intended for normal activities runtime phase. Some other cloud normal activities recovery methods such as fault-tolerance design[5] are implemented in the design phase of a cloud applications. Hence, such recovery methods are intended for cloud normal activities design phase. Apart from recovery methods for cloud normal activities, there are recovery methods for consumer-initiated sporadic activities[3] (e.g. upgrade operation on cloud applications). For a particular sporadic activity, such as upgrade, it can be conducted either manually or with the assistance of automation scripts such as Chef[6]. The sporadic activities won’t actually take effect until the scripts are being executed. When cloud operators are conducting the sporadic activities either manually or automatically, the recovery strategies such as process-oriented recovery methods[2] can be employed if runtime errors happen. Those recovery strategies are intended for sporadic activities runtime phase. When operators are designing the scripts for sporadic activities, they may utilize script mini test[7] to test the availability of the scripts or they may design exception handlers[7] to handle the potential errors. Those recovery strategies are intended for sporadic activities design phase.

Given so many cloud recovery strategies ranging from those for cloud normal activities to those for cloud sporadic activities and from design phase to runtime phase, analyzing them and making taxonomy for them is significant for facilitating the research on recoverability for sporadic cloud operations. Hence, we review and analyze those recovery strategies in order to figure out the taxonomy for them. Moreover, the taxonomy can also help better understand the purposes of existing cloud recovery methods.

Our research has two main contributions: 1) We made a summarization of the existing cloud recovery methods; 2) We make a classification for those cloud recovery methods.

The remainder of this paper is organized as follows. Section II is methodology of establishing the taxonomy; section III is literature review for cloud recovery; section IV is data analysis and taxonomy; section V is discussion; section VI is conclusion and future work.
II. METHODOLOGY OF ESTABLISHING THE TAXONOMY

We establish the taxonomy by a three-step approach. This approach is illustrated in Fig. 1. The first step is to define the research questions that need to be answered for our research. The second step is to extract the data of existing cloud recovery strategies coming from the literature review. The third step is to analyze the data to figure out the taxonomy.

Fig. 1. Methodology of Establishing Taxonomy

A. Defining Research Questions

Our research goal is to provide taxonomy for existing cloud recovery strategies for the purpose of facilitating research on recoverability for sporadic operations on cloud applications. As has been discussed in previous sections, cloud applications have different variety of activities, and those activities have different life cycle phases. The classification will reflect those activities and life cycle phases. As such, we have defined the below three research questions:

1) What are the existing cloud recovery strategies?

2) What types of activities of cloud applications are those methods applicable for?

3) What phases in the activities’ life circle are those methods applicable for?

B. Literature Review and Extracting Data

To answer the research question 1, we do it by reviewing the existing cloud recovery strategies from literatures. We have reviewed several existing papers related to cloud recovery, and have obtained several cloud recovery strategies from cloud consumer’s perspective. They can be categorized into cloud normal activities recovery methods and cloud sporadic activities recovery methods. We extract those existing cloud recovery methods and analyze them by characterizing them and categorizing them. We also make justifications for those recovery methods and discuss over their scopes and contexts.

C. Analysing Data

To answer the research question 2, we analyze what recovery methods are intended for what activities of cloud applications (normal activities or sporadic activities). Activities type can be one dimension of the classification. Normal activities mean the activities of cloud applications themselves, such as application workflow[3]. Among the recovery strategies for cloud normal activities, some strategies are doing recovery on infrastructure level and some methods are doing recovery on application level. Sporadic activities mean the consumer-initiated activities on cloud applications such as deployment and upgrade[3][8].

To answer the research question 3, we analyze what phases of life circle the cloud recovery methods can be applicable for, and what recovery methods are applicable for each life circle phase, for both cloud applications normal activities and cloud applications sporadic activities. Typically, a life circle for an application or a script contains several phases, such as design phase, runtime phase, downtime phase, and off-service phase. We found that some of the recovery strategies are applicable for normal activities design phase or sporadic activities design phase, and some of the strategies are intended for normal activities runtime phase or sporadic activities runtime phase. For the other phases such as downtime phase or off-service phase, the existing recovery methods are just out of the paper scope. Hence, the two phases of design phase and runtime phase can be put as another dimension for the classification.

If we organize these two dimensions as a two-dimension coordinate chart, they should form the below Fig. 2. Now the problem is to figure out what recovery strategies should be placed into each block of the below coordinate chart.

Fig. 2. Coordinate Chart for Dimensions in Taxonomy

III. LITERATURE REVIEW FOR CLOUD RECOVERY

We reviewed a wide range of cloud recovery strategies and they are organized into table I. In this table, recovery strategies and example of each strategy are provided.

<table>
<thead>
<tr>
<th>Recovery Strategies</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cloud Application Rollback</td>
<td>checkpoint based rollback[9]</td>
</tr>
<tr>
<td></td>
<td>Log based rollback[9]</td>
</tr>
<tr>
<td>Disaster Recovery</td>
<td>Geographical redundancy[10]</td>
</tr>
<tr>
<td></td>
<td>Cloud storage replication[11]</td>
</tr>
<tr>
<td>Virtual Machine Replication</td>
<td>Remus[12]</td>
</tr>
<tr>
<td>Fault-Tolerance</td>
<td>Recovery Block[13]</td>
</tr>
<tr>
<td></td>
<td>N-version Programming[14]</td>
</tr>
<tr>
<td></td>
<td>Parallel[15]</td>
</tr>
<tr>
<td></td>
<td>FTCloud[15]</td>
</tr>
<tr>
<td>Recovery for Cloud Internal Protocols</td>
<td>FATE&amp;DESTINI[16]</td>
</tr>
<tr>
<td>Test Driven Scripts</td>
<td>Chef mini test[7]</td>
</tr>
<tr>
<td>Cloud Operations Exceptions Handling</td>
<td>Scripts exceptions handling[23]</td>
</tr>
<tr>
<td></td>
<td>Cloud management tool exceptions handling[24]</td>
</tr>
<tr>
<td>Recovery for Cloud Operations as Transactions</td>
<td>Forward recovery[17]</td>
</tr>
<tr>
<td></td>
<td>Backward recovery[17]</td>
</tr>
<tr>
<td>Recovery for Cloud Operations as Processes</td>
<td>Operation undo &amp; redo[18]</td>
</tr>
<tr>
<td></td>
<td>Reparation[19]</td>
</tr>
<tr>
<td>User Guided Recovery for Cloud Applications</td>
<td>Recovery plan generation[20]</td>
</tr>
</tbody>
</table>
A. Cloud Application Rollback

Rollback recovery for cloud applications treats a cloud distributed system as a collection of application processes that communicate through the network in the cloud[9]. It can be classified into two categories: checkpoint based rollback[9] and Log based rollback[9]. For the checkpoint based rollback, upon a failure, a process will roll back to the previously saved consistent checkpoint which includes at a minimum the state of the participating processes. In contrast, the log based rollback recovery combines checkpointing with logging of nondeterministic events[9]. It logs all the nondeterministic events that a process executes as well as the information necessary to replay each event. By replaying those logged events when failure happens, a process can be recovered. Hence, log based rollback recovery is particularly attractive for applications that frequently interact with the outside world[9]. As such, both checkpoint based rollback and log based rollback are intended for applications running in cloud during the applications’ run time. The checkpoints as well as the event logs are generated at applications’ run time, and the recovery actions are taken at runtime as well. One challenge behind this strategy is the overhead of making checkpoints (e.g. checkpointing a VM) and the efficiency of retrieving and understanding the checkpoints in order to make a rollback recovery.

B. Disaster Recovery

Disaster recovery is significant for maintaining business continuity. There are several existing disaster recovery mechanisms for clouds, ranging from geographical redundancy[10] to cloud storage redundancy[11]. Geographical redundancy approach is used for the recovery of cloud system architecture by replicating datacenter between zones[10]. Cloud storage redundancy approach requires each storage to be equipped with at least three different geographical locations[11]. Moreover, some existing tools such as Zerto[21] and Yuruware[22] can support disaster recovery with in clouds. For example, Zerto enables replication across a heterogeneous range of storage devices and protocols for virtual IT environments based on a hypervisor-level replication solution[21]. And Yuruware enables cross-region replication for cloud applications[22]. Disaster recovery involves two phases: one is disaster recovery design phase where disaster recovery architecture and a disaster recovery plan are generated, and the other one is disaster recovery realization phase, where the disaster is recovered according to the DR plan. One concern about disaster recovery is the efficiency of replicating datacenter or storage from one platform to another, especially for cross-region replication. Another concern is how to retrieve the changed states and data in the previous data center or storage and move them into the backup site.

C. Virtual Machine Replication

Virtual machine replication is needed for cloud applications because of the uncertainty and instability of cloud resources. Virtual machine replication mechanism is also widely used in disaster recovery and fault tolerance[5]. Virtual machine replication in cloud requires that systems be constructed with redundant virtual machines that are capable of maintaining and switching to backups in the face of failure[5]. Replicating virtual machines is encouraged to be asynchronously done because this will have fewer interrupts on the cloud applications themselves. For stateful machines, it is also required that only the changed states and incremental data should be replicated to the machine replicas. One example of asynchronous VM replication strategy which fulfills those requirements is Remus[12], which encapsulates protected software in VM and asynchronously propagates changed state to a backup host at high frequencies. The technology behind Remus has been used in some of today’s commodity cloud applications, such as some e-business websites. Virtual machine replication strategy can be designed in cloud systems design phase and take place at systems runtime phase. One challenge in asynchronous virtual machine replication is the determination of the replication frequency and the avoidance of influencing the servicing system.

D. Fault-Tolerance

Fault-tolerance of cloud services has received great attention for many years, although it is still a strong research focus how to do fault-tolerance more efficiently for different types of cloud services. The idea behind fault-tolerance is to mask faults occurring in a system instead of removing them. For cloud there are mainly three existing fault-tolerance strategies: Recovery Block[13], N-version Programming[14] and Parallel[15]. Recovery block is a means of structuring redundant program modules, where standby components will be invoked if the primary component fails[13]. N-version Programming allows multiple functionally equivalent programs to be independently generated from the same initial specifications, and the functionally equivalent cloud components are invoked in parallel and the final result is determined by majority voting[15]. For Parallel, the multiple functional equivalent components will be invoked in parallel and the first returned response will be used as the final result[15]. In particular, it is recommended that not all the components within a cloud system need to be equipped with fault-tolerance mechanism. For example, FTCloud[15] proposes a component ranking based fault-tolerance framework for cloud applications. In FTCloud, it first employs the component invocation structures and invocation frequencies to identify the significant components in a cloud application, and then the optimal fault-tolerance strategy (RB, NVP, or Parallel) will be automatically selected for each significant component. By doing so, the reliability and efficiency of fault-tolerant cloud applications have been improved. It is obvious that fault-tolerance strategy is designed during cloud system design phase. And fault-tolerance will be triggered (e.g. backup component switching) during system runtime phase. The challenge lying in fault-tolerance is how to further reduce the service downtime since downtime could happen during the switching between two sites.

E. Recovery for Cloud Internal Protocols

For cloud internal protocols, such as the internal processes of HDFS[16], recoverability is enabled by specifying recovery specifications. One typical paradigm for cloud internal protocols recovery is FATE&DESTINI[16], a framework for cloud recovery testing. With FATE&DESTINI, recovery
actions are specified clearly, concisely and precisely by using datalog rules[16] in the domain of cloud internal protocols. For example, a datalog rule would ask the service to create a missing file on a certain node. The recovery action is comprised of a serious of datalog rules that composite the logic of recovery. One drawback of FATE&DESTINI is that it has a strong assumption on the visibility and control level provided by cloud, while in fact cloud platform only provides consumers with very limited visibility and control[1]. It is clear that FATE&DESTINI is intended for the design phase of cloud internal protocols which can be part of cloud systems or cloud normal activities. Since datalog rules cannot be executed without being translated into runnable codes, FATE&DESTINI is not intended for the runtime phase of cloud internal protocols unless the datalog rules are translated into real codes. The main concern with FATE&DESTINI is that it assumes a high visibility and control on cloud platform, while in fact cloud platforms such as AWS EC2 only provides us with very limited visibility and indirect controls[1].

F. Test Driven Scripts

DevOps scripts such as Chef[23] can be used for implementing consumer-initiated cloud sporadic operations, such as continuous deployment or upgrade. In order for the scripts to be more reliable, operation scripts can be written in a test driven manner[7]. In the scripts like Chef, there are several ways to make test driven infrastructure for scripts, for instance, by using scripts mini tests[7]. Specifically, mini tests are usually to test the functionality and availability of a module in the whole script infrastructure. For example, a mini test for a module of shutting down Tomcat service can be conducted to check if Tomcat service will really be shut down successfully. Mini tests are carried on under the test bed and are integrated into the structure of the scripts during the scripts design and testing phase. The errors arising in the test bed could be detected and recovered by the operator. However, the test bed is different from the actual operation run bed, and mini tests cannot guarantee a successful script execution in the runtime. Moreover, test driven scripts will introduce the overhead of implementing the mini tests scripts and running the mini test.

G. Cloud Operations Exceptions Handling

One way to handle the operations runtime error is by using exceptions handling[23][24]. For example, Chef is using exception handlers to implement error handling logics. In Asgard[24], one commonly used exception handling logic is to gracefully exit from the operation in the face of errors. Although exceptions handling is triggered during the runtime phase of scripts or operations, the detailed logic in exceptions handling has to be implemented during the scripts and operation design phase. One of the biggest challenges about cloud exceptions handling, according to [25], is that exceptions handling in cloud should deal with cross-platform and cross-language exceptions in a unified manner.

H. Recovery for Cloud Operations as Transactions

When analyzing cloud operations as transactions, e.g. long running transactions[17], recovery strategies usually involve backward recovery and forward recovery[17]. Backward recovery refers to the strategy which first revert the current erroneous state to a previous correct state before attempting to continue execution. Forward recovery attempts to correct the current erroneous state and then continues normal execution. One form of backward recovery is called rollback & replay[26], which makes the system go back to the previous consistent state and replay the step in the transaction. One form of forward recovery in long running transactions is called compensation[17], which means to attempt to correct the state of a system given some knowledge of the previous actions of the system[17]. When conducting sporadic operations such as rolling upgrade for a large-scale cloud application, backward recovery and forward recovery can be employed for recovering from errors happening during the operations. These recovery strategies are designed and implemented in the design phase of operations, and take effects during the operations runtime when failure presents. The most challenging part in this recovery strategy is cloud system state reachability checking[19] as well as making checkpointing more efficient.

I. Recovery for Cloud Operations as Process Models

When analyzing and modeling sporadic cloud operations as processes, several existing process-oriented recovery methods[2][18] can be utilized for recovery. Generally, process-oriented recovery methods take similarity to transaction-oriented recovery methods. For example, operation undo and redo[18] is similar to rewind and replay. One difference between existing process-oriented recovery methods and existing transaction-oriented recovery methods is that the recovery points should be determined prior to implementing the recovery actions. For example, paper [8] has presented a way of how to determine the recovery points for sporadic operation on cloud applications. For the operation recovery after a particular recovery point, several recovery mechanisms such as checkpoint-based undo and redo[18] or reparation[19] can be adopted. These recovery methods are following a non-intrusive manner without changing the original code of the operation and functioning as an external recovery service which handles recovery for cloud sporadic operations. Hence, this recovery strategy is targeted for operations runtime phase and provides a real time non-intrusive recovery, but its architecture and functionality are designed during operations design phase. One challenge behind this strategy is that the checkpoints have to be more effectively made, and another challenge is to make it fulfill the recovery objectives[28].

J. User Guided Recovery for Cloud Web Service Applications

The recovery strategies described in the above sections are all designed and implemented to recover from failures in an automatic way. However, for complex cloud systems it is difficult to always employ this type of automation recovery, especially when the cloud systems are comprised of large number of different components across different platforms or regions or even different clouds. As such, cloud users are required to be involved in the recovery for this kind of cloud applications[27]. In the paper of [20], a user-guided recovery framework for cloud web service applications is illustrated. Specifically, recovery plans generation[20] will be conducted when behavior correctness properties of an cloud web service application are violated at runtime. After obtaining generated
recovery plans, those plans will be ranked and the user to select the best plan for execution. Obviously, this recovery strategy is used for the runtime of cloud web service applications. And the logic behind is designed during the design phase of cloud web service applications. One challenge for this strategy is the efficient methodology of generating the recovery plans for operators.

IV. DATA ANALYSIS & TAXONOMY

From analyzing the afore-mentioned recovery strategies, we can obtain the activity types for cloud applications as well as their life cycle phases, as shown in Fig. 2. The activity types for cloud applications contain two flavors: normal activities and sporadic activities. Normal activities for cloud applications include applications workflow and internal processes and execution flow within certain system architecture. Sporadic activities for cloud applications include deployment, upgrade, reconfiguration, etc. The life circle phases for either activity type contain design phase and runtime phase. Hence, there are totally four categories for the classification of cloud recovery strategies: 1) Recovery for Normal Activities in Design Phase; 2) Recovery for Normal Activities in Runtime Phase; 3) Recovery for Sporadic Activities in Design Phase; 4) Recovery for Sporadic Activities in Runtime Phase. Based on the analysis of the existing cloud recovery strategies, they are classified these categories.

![Fig. 2. Cloud Activities and Time Phases](image)

**A. Recovery for Normal Activities in Design Phase**

Normal activities involve two levels: application level which means the aspects related to cloud applications execution (e.g. the execution of the program within a component of cloud system), and infrastructure level which means the aspects related to cloud datacenter components structure. In the existing recovery strategies, fault-tolerance design, virtual machine replication design, recovery design for cloud internal protocols and user guided recovery design can be implemented in the design phase of application level cloud normal activities. Hence, these four recovery strategies will be included into the recovery strategies for application level normal activities in design phase. Also, virtual machine replication design, disaster recovery design as well as fault-tolerance design can be implemented in the design phase of infrastructure level cloud normal activities. Hence, they will be included into the recovery strategies for infrastructure level normal activities in design phase.

**B. Recovery for Normal Activities in Runtime Phase**

In all the existing recovery strategies, cloud application rollback, fault-tolerance, virtual machine replication, recovery for cloud internal protocols and user guided recovery for cloud web service applications can be taking effects in the runtime phase of application level cloud normal activities. Hence, they will be included into the recovery strategies for application level normal activities in runtime phase. Also, virtual machine replication, disaster recovery and fault-tolerance can be taking effects in the runtime phase of infrastructure level cloud normal activities. Hence, they will be included into the recovery strategies for infrastructure level normal activities in runtime phase.

**C. Recovery for Sporadic Activities in Design Phase**

Sporadic activities also involve two levels: application level which means the sporadic operations on the applications inside virtual machines (e.g. installing Tomcat in a machine), and infrastructure level which means the sporadic operations on the infrastructure of cloud (e.g. upgrade a machine by using new AMI). In all the existing recovery strategies, test driven scripts can be prepared and conducted under test bed during the design phase of application level sporadic activities. And scripts exceptions handling design, recovery design for operations as transactions and recovery design for cloud operations as processes can be implemented in the design phase of application level sporadic activities. Hence, these four recovery strategies are included into the recovery strategies for application level sporadic activities in design phase. Also, cloud operation exceptions handling design, recovery design for cloud operations as transactions and recovery design for cloud operations as processes can be implemented in the design phase of infrastructure level sporadic activities. Hence, they are included into the recovery strategies for infrastructure level sporadic activities in design phase.

**D. Recovery for Sporadic Activities in Runtime Phase**

In all the existing recovery strategies, recovery for cloud operations as transactions, recovery for cloud operations as processes and scripts exceptions handling can be implemented in the runtime phase of application level sporadic activities. Hence, these three recovery strategies are included into the recovery strategies for application level sporadic activities in runtime phase. Also, cloud operation exceptions handling, recovery for cloud operations as transactions and recovery for cloud operations as processes can be implemented in the runtime phase of infrastructure level sporadic activities. Hence, they are included into the recovery strategies for infrastructure level sporadic activities in runtime phase.

Based on the above data analysis, we can classify the existing cloud recovery strategies into six groups as show in the below Fig. 3. Since in this diagram the normal activities are divided into application level and infrastructure level, we would like to call this diagram level 1 classification diagram.

![Fig. 3. Level 1 classification diagram](image)
Then, by combining application level and infrastructure level in both normal activities and sporadic activities for both design phase and runtime phase, we get the level 2 classification as shown in below Fig. 4. And this serves as the final taxonomy for cloud recovery strategies.

![Fig. 4. Level 2 Classification for Cloud Recovery Methods](image)

**V. DISCUSSION**

Our taxonomy for existing cloud recovery strategies is looking at different cloud application activities (normal activities and sporadic activities) in different life circle phases (design phase and runtime phase). It is noted that some recovery mechanisms are dedicated for operation simulation phases and we treat this as the design phase of operations. With this taxonomy of cloud recovery strategies, we are able to follow a clear guidance on literature review when researching on new cloud recovery mechanisms. For example, existing recovery strategies for cloud sporadic activities would be paid more attention to for proposing sporadic operations oriented recovery mechanisms. But recovery strategies for normal activities such as fault-tolerance might also be useful for sporadic operations recovery.

**VI. CONCLUSION & FUTURE WORK**

This paper presents a taxonomy method for classifying the existing cloud recovery strategies ranging from recovery mechanisms for cloud applications normal activities to recovery mechanisms for cloud applications sporadic activities. This taxonomy is obtained by analyzing those existing cloud recovery strategies and classifying them into four groups. This taxonomy provides a clear view on the purposes and scopes of the current existing cloud recovery strategies and it facilities the research on recoverability of cloud sporadic operations. Our future work include proposing non-intrusive recovery mechanisms for cloud sporadic operations with the guidance of this taxonomy.

**ACKNOWLEDGMENT**

NICTA is funded by the Australian Government through the Department of Communications and the Australian Research Council through the ICT Centre of Excellence Program.

**REFERENCES**