Mesbahi *et al. Hum. Cent. Comput. Inf. Sci.* *(2018) 8:20*
https://doi.org/10.1186/s13673-018-0143-8

Human-centric Computing and Information Sciences

RESEARCH

Open Access

Reliability and high availability in cloud computing environments: a reference roadmap

Mohammad Reza Mesbahi[1], Amir Masoud Rahmani[1,2*] and Mehdi Hosseinzadeh[3]

*Correspondence: rahmani@srbiau.ac.ir
[1] Department of Computer Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran
Full list of author information is available at the end of the article

Abstract
Reliability and high availability have always been a major concern in distributed systems. Providing highly available and reliable services in cloud computing is essential for maintaining customer confidence and satisfaction and preventing revenue losses. Although various solutions have been proposed for cloud availability and reliability, but there are no comprehensive studies that completely cover all different aspects in the problem. This paper presented a 'Reference Roadmap' of reliability and high availability in cloud computing environments. A big picture was proposed which was divided into four steps specifying through four pivotal questions starting with 'Where?', 'Which?', 'When?' and 'How?' keywords. The desirable result of having a highly available and reliable cloud system could be gained by answering these questions. Each step of this reference roadmap proposed a specific concern of a special portion of the issue. Two main research gaps were proposed by this reference roadmap.

Keywords: Reliability, High availability, Cloud computing, Big picture, Failure

Introduction

It was not so long ago that applications were entirely developed by organizations for their own use, possibly exploiting components/platforms developed by third parties. However, with service-oriented architecture (SOA), we moved into a new world which applications could delegate some of their functionalities to already existing services developed by third parties [1].

For meeting ever-changing business requirements, organizations have to invest more in time and budget for scaling up IT infrastructures. However, achieving this aim by own premises and investments not only is not cost-effective but also organizations will not be able to have an optimal resource utilization [2]. Therefore, these challenges have forced companies to seek some new alternative technology solutions. One of these modern technologies is cloud computing, which focuses on increasing computing power to execute millions of instructions per seconds.

Nowadays, cloud computing and its services are at the top of the list of buzzwords in the IT world. It is a recent trend in IT that can be considered as a paradigm shift for providing IT & computing resources through the network. One of the best and most popular definitions of cloud computing is the NIST definition proposed in 2009 and updated in 2011. According to this definition, "Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction" [3].

Recent advances in cloud computing are pushing virtualization more than ever. In other words, cloud computing services can be considered as a significant step towards realizing the utility computing concept [4]. In such a computing model, services can be accessed by users regardless of where they are hosted or how they are delivered.

Over the years, computing trends such as cluster computing, grid computing, service-oriented computing and virtualization have gained maturity but cloud computing is still in infancy, and experiences lack of complete standards and solutions [5]. Therefore, the following critical issues are introduced by cloud business models and technologies including load balancing [6, 7], security [8, 9], energy-efficiency [10–12], workflow scheduling [13–17], data/service availability, license management, data lock-in and API design [1, 2, 5, 18–20].

High Availability (HA) and reliability in cloud computing services are some of the hot challenges. The probability that a system is operational in a time interval without any failures is represented as the system reliability, whereas the availability of a system at time 't' is referred to as the probability that the system is up and functional correctly at that instance in time [21, 22]. HA for cloud services is essential for maintaining customer's confidence and preventing revenue losses due to service level agreement (SLA) violation penalties [23, 24]. In recent years, cloud computing environments have received significant attention from global business and government agencies for supporting critical mission systems [25]. However, the lack of reliability and high availability of cloud services is quickly becoming a major issue [25, 26]. Research reports express that about $285 million have been lost yearly due to cloud service failures and offering availability of about 99.91% [26].

Cloud computing service outage can seriously impact workloads of enterprise systems and consumer data and applications. Amazon's EC2 outage on April, 2011 is an example of one of the largest cloud disasters. Several days of Amazon cloud services unavailability resulted in data loss of several high profile sites and serious business issues for hundreds of IT managers [27]. Furthermore, according to the CRN reports [28], the 10 biggest cloud service failure of 2017, including IBM's cloud infrastructure failure on January 26, GitLab's popular online code repository service outage on January 31, Facebook on February 24, Amazon Web Services on February 28, Microsoft Azure on March 16, Microsoft Office 365 on March 21 and etc., caused production data loss, and prevented customers from accessing their accounts, services, projects and critical data for very long and painful hours. In addition, credibility of cloud providers took a hit in these service failures and unavailability.

Although many research papers and studies have been conducted in recent years such as studies in [26, 29–34], but each of these previous works focused on a special aspect of HA in cloud computing and there are no comprehensive studies, which cover all aspects of HA problem in cloud computing environment based on all cloud actors' requirements. In this paper, a comprehensive study results about cloud computing reliability and HA problems as a 'Reference Roadmap' for future researchers will be proposed.

This research aims to offer a comprehensive strategy for providing high availability and reliability while guaranteeing less performance degradation in cloud computing datacenters. Therefore, by satisfying the users' requirements and providing services according to the SLA and preventing SLA violation penalties, providers can make a huge profit [35]. To achieve this goal, all possible aspects to this issue will be studied by proposing a big picture of the problem (for clarifying the possible future approaches). In this research, the proposed research gaps will be puzzled out by completing the pieces of our big picture through answering four questions starting with 'Where?', 'Which?', 'When?' and 'How?' question words. In the rest of this paper, a proposed reference roadmap will be introduced and its different aspects will be described. Then the possible primary solutions for these proposed questions will be introduced. The main contributions of this study can be considered as follows:

- A reference road map for cloud high availability and reliability is proposed.
- A big picture is proposed through dividing the problem space into four major parts.
- A comprehensive cloud task taxonomy and failure classification are presented.
- Research gaps which were neglected in the literature review are identified.

The rest of this paper is as follows. The research background and literature review is presented in "Research background". The proposed reference roadmap in terms of research "Big Picture" is presented in "Proposed reference roadmap". Discussion and open issues section is presented in "Discussion and open issues". The paper is concluded in "Conclusion".

Research background

Cloud computing provides the context of offering virtualized computing resources and services in a shared and scalable environment through the network on a pay-as-you-go model. By rapid adoption of cloud computing, a large proportion of worldwide IT companies and government organizations have adopted cloud services for various purposes including hosting the mission-critical applications and thus critical data [34]. In order to support these mission-critical applications and data, there is need to provide dependable cloud computing environments.

In order to study dependability of cloud computing, the major cloud computing system (CCS) dependability attributes should be identified which can quantify the dependability of cloud in different aspects. Some important attributes for dependable cloud environments have been mentioned in [36] and include availability, reliability, performability, security, and recoverability.

Five major actors and related roles in cloud environments are described in the NIST Cloud Computing Standards Roadmap document [37]. These five participating actors are cloud provider, cloud consumer, cloud broker, cloud carrier and cloud auditor [37]. Table 1 presents the definitions of these actors. Consumers and Providers are two main roles among these actors which are significantly considerable in the most cloud computing scenarios. Therefore, this research paper focuses on these two actors.

Pan and Hu [36] proposed a summary of the relative strengths of dependency on different dependability attributes for each class of actors shown in Table 1. This empirical

Table 1 Actors in cloud computing [37]

Actors	Definition
Cloud consumer	Any individual person or organization that has a business relationship with cloud providers and consumes available services
Cloud provider	Any individual entity or organization which is responsible for making services available and providing computing resources to cloud consumers
Cloud broker	An IT entity that provides an entry for managing performance and QoS of cloud computing services. In addition, it helps cloud providers and consumers with management of service negotiations
Cloud auditor	A party that can provide an independent evaluation of cloud services provided by cloud providers in terms of performance, security and privacy impact, information system operations and etc. in the cloud environments
Cloud carrier	An intermediary party that provides access and connectivity to consumers through any access devices such as networks. Cloud carrier transports services from a cloud provider to cloud consumers

analysis shows that there are three 'Availability', 'Reliability', and 'Performability' critical requirements for cloud consumers and providers. Thus, on one hand, from the consumers' viewpoint, there is a great requirement for highly available and reliable cloud services, but on the other hand, while cloud providers understand the necessity for providing highly available and reliable services for meeting quality of services (QoS) due to the SLA, they also prefer to have highly utilized systems to achieve more profits [35]. Under these considerations, providing dependable cloud environments which can meet the desires of both cloud consumers and providers is a new challenge.

There are many different design principles such as 'Eliminating Single Point of Failure' [27, 38], 'Disaster Recovery' [39, 40] and 'Real-Time and Fast Failure Detection' [41–43] that can help achieve high availability and reliability in cloud computing environments. The single point of failure (SPOF) in cloud computing datacenters can occur in both software and hardware level. Single point of failure can be interpreted as a probable risk which can cause the entire system failure. Applying redundancy is an important factor for avoiding SPOFs [27]. In simple words, every vital component should exist in more than one instance. At the occurrence of a disaster at a main component in a cloud datacenter, system operations can be switched to backup services and efficient techniques are required for data backup and recovery [40]. The time taken to detect a failure is one of the key factors in the cloud computing environments. So, fast and real-time failure detection to identify or predict a failure in the early stages is one of the most important principles to achieving high availability and reliability in cloud systems [42, 43]. Moreover, there are some new trends in cloud computing such as SDN-based technology like Espresso that makes cloud infrastructures more reliable and available in the network level [44].

A systematic review of high availability in cloud computing was undertaken by Endo et al. [45]. This study aimed to discuss high availability mechanisms and important related research questions in cloud computing systems. From the results, the three most useful HA solutions are 'Failure Detection', 'Replication', and 'Monitoring'. In addition, the review results show that 'Experiment' approach is used more than other approaches for evaluating the HA solutions. However, the paper proposed some research questions and tried to answer these questions, but the study results are more similar to the study mapping review. Furthermore, it did not consider the different cloud actors' requirements in terms of high availability.

Liu et al. [46] applied an analytical modeling and sensitivity analysis for investigating the effective factors on cloud infrastructure availability such as repair policy and system parameters. In this study, the replication method was used to provide physical machine availability. Two different repair policies were considered in this study, and a Stochastic Reward Nets availability model was developed for each policy. The numerical results of this study showed that both policies provide the same level of availability but with the different cost level. The system availability was assessed through modeling without considering the failure types in this paper. In addition, the limited number of hot pares and repair policies cannot be sufficient to evaluate the large-scale cloud environments.

The availability and performance of storage services in private cloud environments were evaluated in [47]. In this study, a hierarchical model was applied for evaluation which consists of Markov chain, stochastic Petri Nets and reliability block diagrams. The result of this study showed that the adoption of redundancy could reduce the probability that timeouts occurred and users were attended to due to failures.

An et al. [32] presented a system architecture and framework of fault tolerance middleware to provide high availability in cloud computing infrastructures. This middleware uses the virtual machine replicas according to a user defined algorithm for replication. Proposing an optimal replica placement in cloud infrastructure is an important issue in this study. So, authors developed an algorithm for online VM replica placement.

Snyder et al. [48] presented an algorithm for evaluating the reliability and performance of cloud computing datacenters. In this study, a non-sequential Monte Carlo simulation was used to analyze the system reliability. This paper demonstrated that using this approach can be more efficient and flexible for cloud reliability evaluation when there is a set of discrete resources.

A cloud scoring system was developed in [49] that can integrate with a Stochastic Petri Net model. In this study, while an analytical model was used for evaluating the application deployment availability, the scoring system can suggest the optimal HA-aware option according to the energy efficiency and operational expenditure. The proposed model in this study can consider different types of failures, repair policies, redundancy, and interdependencies of application's components. One of the main contributions of this study is proposing an extensible integrated scoring system for offering the suitable deployment based on the users' requirements.

A comparative evaluation of two redundancy and proactive fault tolerance techniques was proposed in [50], based on the cloud providers' and consumers' requirements. This evaluation was proposed in terms of cloud environment's availability based on the consumers' viewpoint and cost of energy from the providers' viewpoint. The result of this study showed that the proactive fault tolerance methods can be better than traditional redundancy technique in terms of cloud consumers' costs and execution success rate.

A framework for amending GreenCloud simulator was proposed in [51] to support the high availability features in simulation processes. In this study, the necessity of a simulator that can provide the HA features was addressed. Then, the phased communication application (PCA) scenario was implemented to evaluate the HA features, workload modeling and scheduling.

Table 2 presents a comparative analysis of some availability and reliability solutions. The comparative factors in this table are main idea, advantages, challenges and evaluation metrics.

The lack of evaluating solutions to quantitatively assess the availability of provided cloud services is one of the main issues and gaps. In addition, according to the literature review, it seems that VM performance overhead [52] can affect the system availability. Furthermore, the high availability common techniques such as VM migration can also affect the VM performance overhead. So, considering the mutual impact of VM performance overhead and high availability solutions in cloud environments is another important research gap in the current CCS study area.

Proposed reference roadmap

This section presents the proposed reference roadmap in terms of the big picture shown in Fig. 1. The goal of this section is to cover all different aspects of the reliability and availability issues in cloud computing research area. So, the area in the big picture is divided into four major steps. Figure 1 illustrates the general scheme of the proposed big picture of our reference roadmap. This big picture represents all aspects and factors of high availability and reliability issues in cloud computing environments. In this big picture, a eucalyptus-based architecture illustrates the major and key components of a cloud computing environment. By determining cloud key components in this architecture we will be able to identify the most important factors that can affect system high availability and reliability. The main components include VM, node controller, cluster controller, storage controller, cloud controller and Walrus [23]. Four major steps to achieve a high available and reliable cloud system are posed in terms of four pivotal questions. In addition, other related issues such as cloud computing nodes' performance have been considered in this big picture.

By proposing this big picture we aim to have a comprehensive look at the problem area and have a reference roadmap for taking each step to solve the problem in cloud computing environments. It is believed that by taking these four steps in terms of answering the four questions, a high available and reliable cloud computing system will be obtained and while satisfying the cloud consumers' requirements, the providers' concerns could also be considered.

In the rest of this section, the different parts of our proposed big picture will be explained in details. Each question will be discussed and current available approaches and solutions for each part will be proposed. In future works, we aim to study each part in more details as another independent research and use this big picture as a reference roadmap as mentioned earlier.

Where?

"Where are the vital parts for providing HA in the body of cloud computing datacenters?"

There are many different geographically distributed cloud clusters in a cloud environment. Cloud computing datacenters comprise of a different stack of components such as physical servers with heterogeneous hardware characteristics like different processor speed, disk and memory size [53]. In addition, based on the property of workload heterogeneity and dynamicity, datacenters run a vast number of applications with

Table 2 Comparative analysis of cloud availability and reliability solutions

Papers' reference	Main idea	Advantages	Challenges	Evaluation metrics
[46]	Investigating the repair policy and system parameters on cloud availability	Repair policy effectiveness is evaluated Using differential analysis for analyzing parameter sensitivity	Types of Failures are not considered Limitation on number of physical machines The lack of different types repair facilities in evaluation	MTTR Steady state availability MTTM Pool size
[47]	Storage services availability evaluation using hierarchical models	Adoption of availability importance index Critical components availability identification	Study case study and assessments are limited to the Eucalyptus platform	MTTF MTTR File Size MaxClients InService Throughput
[32]	Using VM replicas in cloud datacenter to provide high availability	Resource optimization while assuring availability	VM and application scheduling is not considered in the proposed method Evaluation on a small cloud infrastructure	Latency OMG DDS QoS Standard Deviation
[48]	Applying non-sequential Monte Carlo Simulation to reliability evaluation	A new cloud computing test-bed were developed A new algorithm for expansion planning were presented	This approach cannot be used for modeling other reliability features such as live VM migration	Number of failures Number of VM allocations
[49]	Using a combination of a stochastic Petri net model and a proposed cloud scoring system	Considering both cloud consumers and cloud providers in the proposed method Proposing a cloud scoring system	The proposed cloud scoring system overhead and cost is not considered The user requirements are limited to only cost and energy in this study	OPEX option Carbon footprint option Overload factor Deployment Distances Relative average utilization
[50]	Comparing two fault tolerance techniques according to the cloud consumers' and providers' requirements	Considering both cloud consumers' and cloud providers' requirements	Failure prediction mechanism is required	MTBF Electricity Bill Failure prediction accuracy Energy consumption Task completion rate
[51]	Amending the current cloud simulators to support HA features	Considering green computing	Limited availability evaluation metrics	Request per second Average service time Power consumption

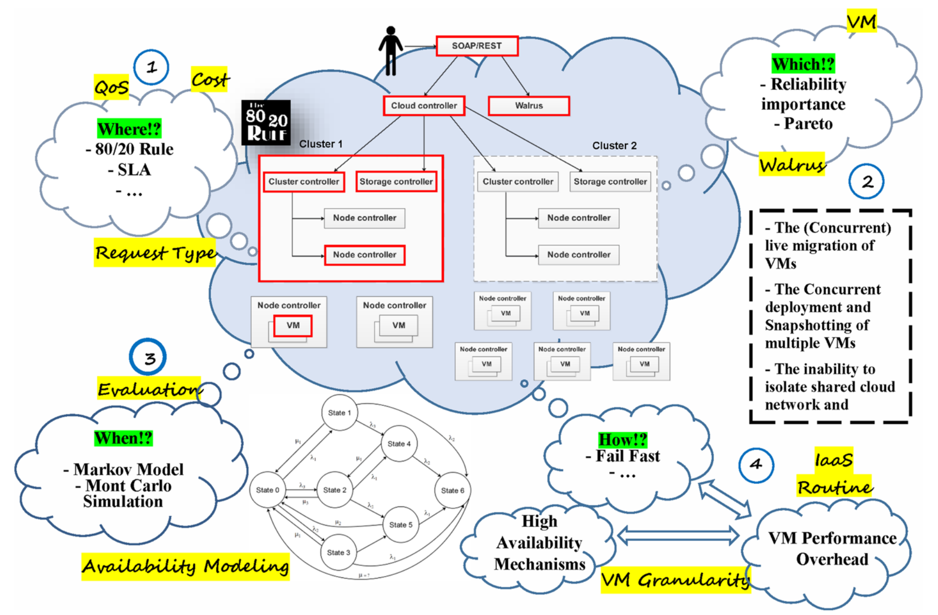

Fig. 1 Proposed big picture of high availability and reliability issue in cloud computing environments

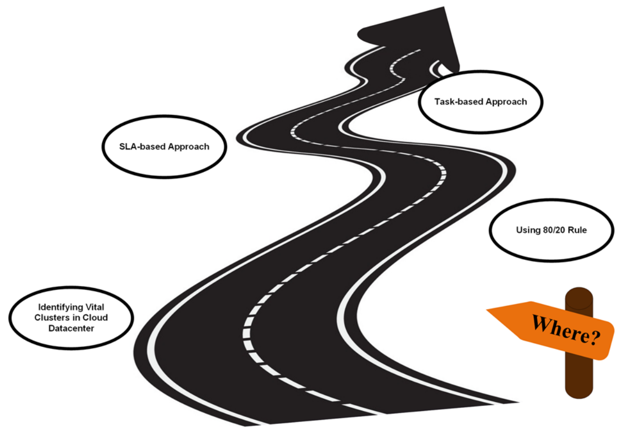

Fig. 2 Proposed roadmap of the "Where" step

diverse characteristics. Particularly, an application can be divided into one or more processes running on dedicated virtual machines (VM) and the resource requirement differs from VM to VM [53]. Therefore, by posing the where question at the first step, we are trying to find out where the vital parts are with high priority and basic requirement for high availability. Figure 2 illustrates the proposed roadmap of the "Where" step. The following possible approaches are available for offering primary solutions for the where question.

- SLA-based approach

A service level agreement is simply defined as a part of a standardized service contract where a service is formally defined. It is an agreement about the quality of a provided service.

The agreement describes terms regarding service usage rate and delivery which are agreed between the service providers and the consumers. SLAs contain the following parts [54]:

- The agreement context. Signatory parties, generally the consumer and the provider, and possibly third parties entrusted to enforce the agreement, an expiration date, and any other relevant information.
- A description of the offered services including both functional and non-functional aspects such as QoS.
- Obligations agreement of each party, which is mainly domain-specific.
- Policies: penalties incurred if a SLA term is not respected and SLA violation occurs.

Service level agreements can also be discussed at these three different levels:

- *Customer-based SLA* It is a type of agreement with a single customer that covers all the necessary services. This is similar to the SLA between an IT service provider and the IT department of an organization for all required IT services.
- *Service-based SLA* It is defined as a general agreement for all customers who are using the delivered services by the service provider.
- *Multi-level SLA* This kind of agreement can be split into different levels, with each level addressing a different set of customers for the same services.

By using this approach, we can focus on the SLA in the 'Multi-level SLA' to find the clusters which are offering services to the users whose basic requirement and first priority to run their tasks are highly available in the entire cloud system. Then by determining these clusters it can be said that the vital parts of the system are known and if high availability can be provided for these clusters, it can be said that while providing a high available system, more benefits are gained by preventing SLA violation penalties.

- Using 80/20 Rule

Pareto principle or 80/20 rule is a useful rule in computer science world [55–57]. In simple words, it says that in anything, a few (about 20%) are vital and many (about 80%) are trivial. In the subject of cost in cloud computing environments, 80/20 rule can be used which says that 80% of the outcomes will come from 20% of your effort [58]. In addition, the 80/20 rule is leveraged in our previous study to provide a highly reliable architecture in cloud environments [57]. According to our research results, there are a reliable sub-cluster and a highly reliable zone in cloud computing datacenters which can be used to serve the most profitable request. This rule can be applied from two different perspectives:

- 80% of cloud service providers' profits may come from 20% of customers.
- 80% of requested services consist of just 20% of the entire cloud providers' services.

So in this step, we can limit the domain of providing high availability in cloud computing clusters in our study based on this assumption that the majority of cloud providers' profits will be earned by offering 20% of the entire services to 20% of whole customers. Therefore, if the high availability in the clusters which belong to that 20%

of customers can be guaranteed, then it can be claimed that cloud providers will make the maximum profits while offering a system with high availability.

- Task-based approach

Providing high availability for different requests and incoming workloads according to the requested task classification and through various suitable mechanisms for each task's class is another approach for this step. So, if the cloud computing tasks can be classified according to their resource requirements (CPU, Memory, etc.), then cloud services and tasks high availability can be provided by hosting tasks in the suitable clusters to avoid task failure due to the resource limitation. For instance, a memory-intensive task will be forwarded into a cluster with sufficient memory resources. Therefore, tasks can be completed without any execution interruption related to the lack of memory errors.

There are many different types of application software and tasks that can be executed in a distributed environment such as high-performance computing (HPC) and high-throughput computing (HTC) applications [59]. It is required to provide a massively parallel infrastructure for running High-performance computing applications using multi-threaded and multi-process program models. Tightly coupled parallel jobs within a single machine can be executed efficiently using HPC applications. This kind of applications commonly uses message passing interface (MPI) to achieve the needed inter-process communication. On the other hand, distributed computing environments have had awesome achievements to execute loosely coupled applications using workflow systems. The loosely coupled applications may involve numerous tasks that can be separately scheduled on different heterogeneous resources to achieve the goals of the main application. Tasks could be large or small, compute-intensive or data-intensive and may be uniprocessor or multiprocessor. Moreover, these tasks may be loosely-coupled or tightly-coupled, heterogeneous or homogeneous and can have a static or dynamic nature. The aggregate number of tasks, the quantity of required computing resources and volumes of required data for processing could be small but also extremely large [59].

As mentioned earlier, the appearance of cloud computing guarantees providing highly available and efficient services to run applications like web applications, social networks, messaging apps etc. Providing this guarantee needs a scalable infrastructure including many computing clusters which are shared by various tasks with different requirements and quality of service in terms of availability, reliability, latency and throughput [60]. Therefore, to provide required service level (e.g. highly available services), a good understanding of task resource consumption (e.g. memory usage, CPU cycles, and storages) is essential.

As cloud computing has been in its infancy during the last years, the applications that will run on clouds are not well defined [59]. The Li and Qiu [61], expressed that the required amount of cloud infrastructure resources for current and future tasks can be predicted according to the major trends of the last decade of the large-scale and grid computing environments. First, singular jobs are mainly split into two data-intensive and compute-intensive tasks categories. It can be said that there are no tightly coupled parallel jobs. Second, the duration of individual tasks is dimensioning with every year; few tasks are still running for longer than 1 h and majority require

Table 3 Cloud task classification [60]

Final class	Duration (h)	CPU (Cores)	Memory (GBs)
1: sss	Small	Small	Small
2: sm*	Small	Med	All
3: slm	Small	Large	Small + med
4: sll	Small	Large	Large
5: lss	Large	Small	Small
6: lsl	Large	Small	Large
7: llm	Large	Large + med	Small + med
8: lll	Large	Large + med	Large

only a few minute to complete. Third, compute-intensive jobs will be divided into Dag-based workflow and bags-of-tasks (BoTs). But data-intensive jobs may utilize several and different programming models.

A task classification has been done in [60] based on the tasks' resource consumption running in the Google Cloud Backend. The results of this study are presented in Table 3. Cloud computing tasks are classified based on their execution duration, number of required CPU cores and amount of required memory in this study. The essential amount of resources for each class of cloud tasks is represented with three major 'Small', 'Med (Medium)', and 'Large' factors, which are abbreviated as 's', 'm' and 'l', respectively in the 'Final Class' column of Table 3. Three words in this column represent the duration, cores and memory, correspondingly. In addition, '*' means that all three factors are possible in this type of class. For instance, the Final Class 'sm*' refers to a class of tasks having an execution duration as 'small' (short-running tasks), the number of required CPU cores are 'Med' and they would consume 'small', 'Med' or 'large' amount of memory.

In addition, Foster et al. [59] characterize the clouds' application to be loosely coupled, transaction oriented (small tasks in the order of milliseconds to seconds) and likely to be interactive (as opposed to batch-scheduled).

The proposed taxonomy of cloud computing tasks is presented in Fig. 3. According to our studies, cloud tasks can be classified into two long-running and short-running

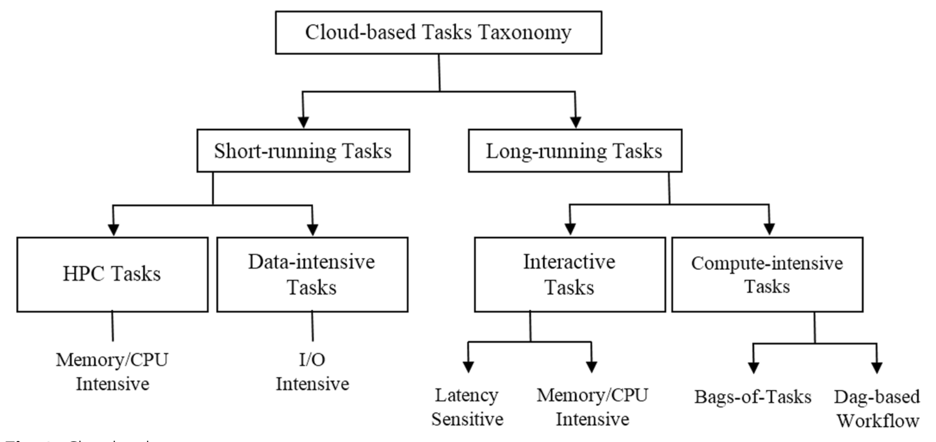

Fig. 3 Cloud tasks taxonomy

tasks in terms of tasks' duration. Therefore based on Table 3, it can be said that task durations are bimodal, either somewhat less than 30 min or larger than 18 h [60]. Such behavior results from the characteristics and types of application tasks running on cloud infrastructures. There are two types of long-running tasks. The first are interactive or user-facing tasks which run continuously so as to respond quickly to a user request. The second type of long-running tasks is compute-intensive, such as processing web logs [60]. Interactive tasks consume a large amount of CPU and memory during periods of a high user request rate. It means that they are CPU-intensive and memory-intensive. In addition since they handle end-user interactions, they are likely latency-sensitive. There are several types of short-running tasks. These tasks dominate the task population. Some short duration tasks are highly parallel operations such as index lookups and searches [60]. We can also mention HPC tasks. Some tasks can be considered as short memory-intensive tasks which include memory intensive operations like map reduce workers that compute an inverted index. These types of tasks are specified as class 2 in Table 3. Other tasks which include CPU-intensive operations are considered as short CPU-intensive tasks like map reduce workers which compute aggregation of a log. Finally, it can be said that typical data-intensive workloads consist of short-running, data-parallel tasks. For data-intensive applications, data should be moved across the network, which represents a potential bottleneck [62].

According to the proposed cloud tasks taxonomy and previous discussion we can have more understanding of cloud tasks' requirements and suggest the best approach for providing HA based on the tasks' types. As the conclusion of this section, it can be said that interactive tasks require more HA from the customer perspectives. Because they are interacting with end users and consuming a large amount of CPU and memory during periods of high user request rate, then by detecting and providing more CPU and Memory resources for these group of tasks, their availability can be improved. In addition, as they are latency sensitive, then providing these resources should be done during a specific threshold. Likewise, for other types of tasks, related requirement and resources for being highly available can be provided.

Which?

"Which components play key roles to affect cloud computing HA and reliability?"

As mentioned earlier we identified main constituent components of a cloud computing architecture which are inspired by the Eucalyptus architecture. By answering to "Which" question, we are trying to know the system's weak points of HA and reliability in this step as illustrated in Fig. 4. After knowing these weak points, we will be able to think about how we can improve them and find suitable solutions in the next steps. Therefore, to catch this goal all cloud failures and their causes should be classified first. Next, all important reliability and HA measuring tools and metrics to evaluate the importance degree of each cloud components in the proposed architecture will be specified. Some main failure modes of CCSs have been proposed in [36, 63, 64] but as one of the future works of this research, different viewpoints exist for proposing a comprehensive classification of cloud failures. Table 4 presents a summary of these studies. Six main failure modes include software failures, hardware failures, cloud management system failures, security failures, environment failures and human faults.

Table 4 Failure classification in cloud computing systems

Failure classification	Failure modes	Description
Software failures	System/application software failure [36]	The cloud tasks and VM hypervisors are actually software programs running on different computing nodes, which may contain software faults, bugs, and errors
	Database failure [36]	There is the possibility of hardware or software failure in each database system. So, database systems are prone to losing data
Hardware failures	Hardware component failure [65]	The computing resources, in general, have hardware components (such as storage devices, processing elements, and memory) which may also encounter hardware failures
	Network failure [65]	When cloud tasks access remote data sources, the communication channels could be broken, which causes the network failure, especially for the long time transmissions of large datasets
Cloud management system (CMS) failures	Overflow [66, 67]	There is usually a limitation on the maximal number of incoming requests in the queue. Waiting too long in the queue can cause the Timeout failure for new requests. So, if the queue is full, new requests will be dropped simply which is called an overflow failure
	Timeout [66, 67]	The cloud service commonly has its due time set by the owner or the service monitor. If the waiting time of the queued requests is over the due time, the Timeout failure occurs. Therefore, those timeout requests will be dropped from the queue
	Data resource missing [66, 67]	In CMS, the data resource manager should register data resources. However, it is possible that some previously registered data are removed but the data resource is not updated. So, data resource missing will happen
	Computing resource missing [66, 67]	The computing resource missing is another failure like data resource missing that can also happen in the cloud management system. This failure will happen because of the reasons like turning off the PC without notifying the CMS
Security failures	Customer faults [68]	The recent research results show that only a small portion of security failures impacting cloud services consumers have been due to the provider's fault. According to the Gartner's top predictions for IT users for 2016 and beyond, about 95% of cloud security failures through 2020 will be the customer's faults
	Software security breaches [69]	Software security breaches can lead to the cloud services failure. When the attackers can gain access to the customer information such as login data, credits and etc. through the cloud-based software security breaches, it can result in huge problems for the customers who rely on their daily cloud-based activities

Table 4 (continued)

Failure classification	Failure modes	Description
	Security policy failure [69]	Miscalculating the cloud security requirements in providing a security policy is really a hot challenge which leads to system failures. Common mistakes to define a comprehensive security policy are some of the main reasons for security failure
Human Operational Faults	Misoperation [67]	This kind of failure is related to accidental faults made by human personnel operating or configuring the system, for both updates of the system and during a repair process. The extent to which this misoperation affects the cloud system can depend on the level on which the fault has occurred
	Misconfiguration [67]	There is a possibility of affecting a whole cluster or even a whole datacenter in a cloud system in case network node software is misconfigured. The worst case, however, remains the misconfiguration of the cloud management software which leads to bringing down all the cloud at once
Environmental Failures	Environmental disasters [67]	Environmental disasters can play the main role in the dependability of a cloud system. Factors such as floods, power outages, fires etc. are although outside the control of the service provider but can always interrupt service provision. This is because these environmental disasters like floods and power outages affect a whole cloud datacenter and hence their consequences can be a very large-scale service disruption
	Cooling system failure [67]	The functionality of physical servers in a cloud datacenter also depends on the thermal conditions of the location where the servers are installed. So, failure in the air-conditioning system where servers are placed also causes failure in services provision. Therefore, the servers will either shut down completely or will be under-utilized for offering services and hence can be regarded as unavailable

One of the important aspects of software failures is database failure and data resource missing [36]. Therefore one of the HA and reliability issues in cloud computing systems will be based on the user requests to unavailable or removed data resources. For providing HA and reliability for data related services we can use 80/20 rule based on the fact that most of the data requests only access a small part of the data [55]. So, concentrating on hotspot data which are normally less than 20% of the whole data resource in terms of 80/20 rule for improving system's availability and reliability could be one of the future approaches of this research.

Hardware failure is another important failure mode in cloud computing datacenters. In [65], hardware failures of multiple datacenters have been examined for determining explicit failure rates for different components including disks, CPUs, memory, and RAID controllers. One of the most important results of this study is that disk failure is the major source of failures in such datacenters [65]. Therefore, cloud storage devices or storage controllers as pointed in the proposed architecture can be one of the main components that can affect system reliability. Based on the conclusion of [65], we propose the hypothesis that failures of components follow the Pareto principle in cloud computing datacenters. This hypothesis can be studied as another future work of this paper. Therefore it can be said that about 80% of cloud system failures are related to storage failures.

Some researches conducted about the networks of CCS show that most of the datacenter networks and switches are highly reliable and only load balancers most often experience faults due to software failures [26, 70].

Cloud management system (CMS) is the last important failure classification of Table 4 which will be considered in this section. Cloud management system can be considered as the manager of cloud computing services. A cloud management system uses the combination of software and technologies for handling the cloud environments. In other words, CMS can be considered as a response to the management challenges of cloud computing. At least, a CMS should be able to:

- Manage a pool of heterogeneous resources.
- Provide remote access for end users.
- Monitor system security.
- Manage resource allocation policies.
- Manage tracking of resource usage.

As aforementioned in [71], three different types of failures have already been identified for CMSs and include: (1) technological failures that result from hardware problems, (2) hardware limitations creating management errors as a result of limited capability and capacity for processing information and (3) middleware limitations for exchanging information between systems as a result of various technologies using different information and data models. Currently, CMSs can detect the failures and follow the predefined procedures to re-establish the communications infrastructure [71]. However, the most important type of failure is related to content and semantic issues. These failures will occur when management systems operate with incorrect information, when data is changed erroneously after a translation or conversion process, or when data is misinterpreted. So, this kind of problem is still largely unsolved and is a serious problem.

One of the fundamental reasons behind avoiding the adoption and utilization of cloud services is the issue of security. Security is often considered as the main requirement for hosting critical applications in public clouds. Security failure which is one of the most important modes of cloud computing failures can be divided into three general modes which include: customer faults, software security breaches and security policy failure.

According to the Gartner predictions for IT world and users for 2016 and beyond, through 2020, 95% of cloud computing security failures will occur because of customers' faults [68]. So, it can be said that customers' faults will be the most important reason of cloud security failures. In addition, cloud-based software security breach is another serious issue. When cloud services are unavailable because of cloud system failures, this can cause huge problems for users who depend on their daily cloud-based activities, loss of revenue, clients and reputation for businesses. The reported software security breaches in recent years such as Adobe's security breaches, resulted in service unavailability and cloud system failures [69].

Design and human-made interaction faults currently dominate as one of the numerous sources of failures in cloud computing environments [67]. In other words, an external fault can be considered as an improper interaction with system during the operational time by an operator. But because environmental failures and human operational faults are considered as external faults from the system viewpoint, they are beyond the scope of this paper. Based on the previous discussion it can be concluded that hardware component failures and database failures are two most important hotspots for more future studies in this step. Likewise, the most impressive factors among all cloud components can be specified as a new failure classification in future work of this research step. In addition, some reliability and availability measuring tools that would enable us to identify the most effective components in the cloud systems' availability and reliability will be used. So the second part of this step for answering "which?" question involves identifying and classifying these measures. In the rest of this section, some of these measures will be introduced.

- Recurring faults

As earlier discussed, another goal of this part is to classify all-important reliability and HA evaluation measures. The study in [72] proposes this point that when a PC component fails, it is much more likely to fail again. This is called recurring faults which can be helpful after identifying faulty components in this step.

- Reliability importance (RI)

Identifying the weaknesses of a complex system is not as easy as identifying these weak components in a simple system such as series systems. In the complex systems, the analyst requires a mathematical approach that will provide the means of

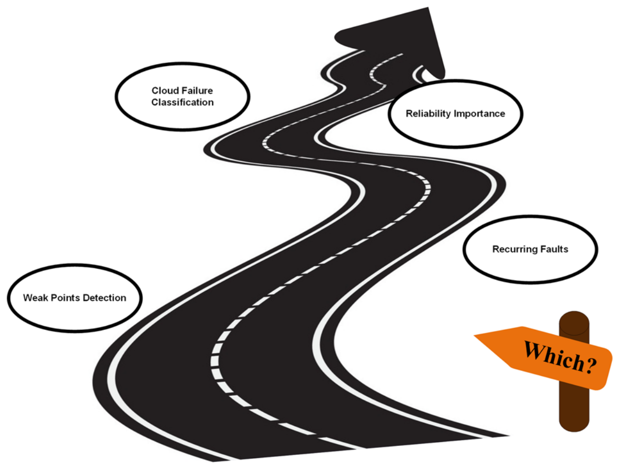

Fig. 4 Proposed roadmap of the "Which" step

identifying the importance of each system component. Reliability Importance (RI) is a suitable measure to identify the relative importance of each component according to the total reliability of the system. The reliability importance, IR_i, of component i in a system of n components is given as [73]:

$$RI_i(t) = \frac{\partial R_s(t)}{\partial R_i(t)} \tag{1}$$

where $R_s(t)$: is the system reliability at a certain time, t; $R_i(t)$: is the component reliability at a certain time, t.

The RI measures the rate of change at the certain time t of the system reliability regarding the components reliability change. The probability that a component can cause system failure at time t, can also be measured by the RI. Both reliability and current position of a system component can affect the calculated reliability importance in Eq. 1.

So, in this step, the most important components which can have a high impact on cloud computing reliability will be identified. Therefore this step focuses on reliability issues as the first concern.

When?

"When reliability and HA will decrease in cloud computing environments?"

Concentration will be on evaluating the different system states by answering the "When" question from the HA and reliability points of view to identify the causes and times of availability and reliability degradation at this step. Figure 5 shows the proposed roadmap of this step.

Availability and reliability models capture failure and repair behavior of systems and their components. Model-based evaluation methods can be one of the discrete-event simulation models, analytic models or hybrid models using both simulation and analytic parts. An analytic model is made up of a set of equations describing the system behavior. The evaluation measures can be obtained by solving these equations. Actually, analytical models are mathematical models that present an abstraction from the real world system in relation to only the system behaviors and characteristics of interest.

This section introduces the model types used for evaluating the availability and reliability of a system. These types can be classified into these three classes:

Combinatorial model types The three most common combinatorial model types which can be used for availability and reliability modeling under certain assumptions are "Fault Tree", "Reliability Block Diagram" and "Reliability Graph". These three models consider the structural relationships amongst the system components for analytical/numerical validations.

- Reliability block diagram

The reliability block diagram (RBD) is a type of inductive method which can be used for large and complex systems reliability and availability analysis. The series/parallel

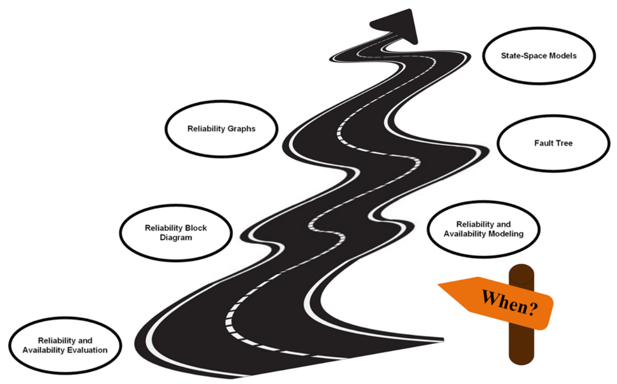
State-Space Models
Reliability Graphs
Fault Tree
Reliability Block Diagram
Reliability and Availability Modeling
When?
Reliability and Availability Evaluation

Fig. 5 Proposed roadmap of the "When" step

configuration of RBDs assist in modeling the logical interactions of complex system components failures [74]. RBDs can also be used to evaluate the dependability, availability and reliability of complex and large scale systems. In other words, it can be concluded that RBDs represent the logical structure of a system with respect to how the reliability of each component can affect the entire system reliability. In the RBD model, components can be organized into three: "Series", "Parallel" or "k-out-of-n" configurations. In addition, these combinations together can be used in a single block diagram. Each component with the same type that appears more than once in the RBD is assumed to be a copy with independent and identical failure distribution. Every component has a failure probability, a failure rate, a failure distribution function or unavailability attached to it. A study of all existing RBD based reliability analysis techniques has been proposed by [74].

- Reliability graphs

The reliability graph is a schematic way of evaluating the availability and reliability. Generally, a reliability graph model consists of a set of nodes and edges, where the edges (arcs) represent components that have failure distributions. There is one node without any incoming edges in a reliability graph and is called the reliability graph source. In addition, there is another node without any outgoing edges and is called the sink or terminal or destination node. When there is no path from the source to the sink in a reliability graph model of a system, it is considered as a system failure. The edges can have failure probabilities, failure rates or distribution functions same as the RBDs.

- Fault tree

Fault trees are another type of combinatorial models widely used for assessing the reliability of complex systems through qualitative or quantitative analysis [75]. Fault trees can represent all the sequences of components failures that cause the entire system failure and stop system functioning, in a treelike structure. This method visualizes the cause of failure which can lead to the top event at different levels of detail up to basic events. Fault Tree Analysis (FTA) is a common probabilistic risk assessment technique that enables investigation of the safety and reliability of systems [76].

The root of a fault tree can simply be defined as a single and well-defined undesirable event. In the reliability and availability modeling, this undesirable event is the system failure. To evaluate the safety of the system, the potentially hazardous or unsafe condition is considered as an undesirable event. The fault tree is a schematic representation of a combination of events that can lead to the occurrence of a fault in the system.

State-space models The models discussed in the previous section will be solved by using algorithms which assume that there is a stochastic independence interaction between different system components. Therefore, for availability or reliability modeling, it is assumed that the component failure or repair was not affected by other failures in the system. Therefore, to be able to model more complicated system interactions, other types of availability and reliability models such as state space models should be used.

State space models constitute a powerful method for capturing dependencies amongst system components. In other words, this model is a general approach for availability and reliability modeling which is a collection of stochastic variables which show the state of the system at any arbitrary time. The Markov chain is an example of this model type. A Markov model can be defined as a stochastic model which is used in the modeling of complex systems. It is assumed that future states in a Markov model only depend on the current state of the system. In other words, it is independent of the previous events. There are usually four known Markov models which can be used in different situations. The Markov chain is the simplest type.

A simple type of stochastic process which has the Markov properties can be considered as a Markov chain. The "Markov F Chain" term is used to refer to the sequence of the stochastic variables in this definition.

Reliability/availability modeling of any system may produce less precise results. The use of graceful degradation will make it possible for a system to provide its services at a reduced level in the existence of failures. The Markov reward model (MRM) is the usual method of modeling gracefully degradable systems [77–80]. In addition, the other common model types are: (1) Markov reward model or irreducible semi-Markov reward model and (2) stochastic reward nets.

Non-state-space models can help to achieve efficiency in specifying and solving the reliability evaluation, but these models assume that components are completely independent. For example, in RBDs, fault-tree or reliability graph, the components are considered as some completely independent entities in a system in terms of failure and repair behavior. It means these models assume that a component failure in a system cannot affect the function of another component. However, Markov models have the ability to model systems that reject the assumptions made by the non-state-space models but at the cost of the state space explosion.

Hierarchical models Hierarchical models can assist in solving the state space explosion problem. They can allow the combination of random variables that show different system parameters to have a general distribution [81]. Therefore, by this approach, we can obtain more realistic reliability/availability models to analyze the complex systems. In other words, large-scale models can be avoided using hierarchical model composition. If the storage issue can be solved, the specification problem will be solved by using briefer model specifications which are transformable to the Markov models.

As a kind of summary for this section, we can express that in the context of availability and reliability modeling, many of the initial works focus on the use of Markov chains [82–86], because a cloud computing system is effectively considered as a complex availability problem. Other studies concentrate on the conceptual problems, priority and hierarchical graphs or the development of performance indices [87]. As stated earlier, combinatorial model types like RBDs consider the structural relationships amongst the system components which can represent the logical structure of a system with respect to how the reliability of its components affects the system's reliability. In addition, state space models like Markov chains can be used for the modeling of more complicated interactions between components. Therefore, it seems that using hierarchical hybrid models [88], for example, combining RBDs and Generalized Stochastic Petri Nets (GSPNs) [89], RBDs and an MRM [90] are more suitable for evaluating cloud-based data centers' availability and reliability.

To have a better understanding of reliability/availability models, the relationship between availability and reliability and their relationship with maintainability for determining the degradation states in details need to be considered, like the study proposed by [91]. Table 5 shows a summary of this study on these relationships.

Availability is defined as the probability that the system is operating properly at a given time t and when it is accessed for use. In other words, availability is defined as the probability that a system is functional and up at a specified time. At first, it would seem that if a system has a high availability, then it is also expected to have a high reliability. However, this is not necessarily true [91], as shown in Table 5.

Reliability shows the probability of a system/component for performing the required functions in a period of time without failure. It does not contain any repair process in its definition. It also accounts for the period of time that it will take the system/component to fail during its operating time. Therefore, according to the definition of availability, it can be said that availability is not only a function of reliability, but it is also a function of maintainability. Table 5 shows the reliability, maintainability, and availability relationships. From this table, it should be noted that an increase in maintainability means a decrease in the repair time and maintenance actions period. As shown in Table 5, if the reliability can be held constant, even at a high value, this does not directly mean providing high availability, because as the time to repair increases, the availability will decrease. Therefore, even a system with a low reliability could have a high availability provided that the repair time does not take too long.

Table 5 The relationship among availability, reliability, and maintainability [91]

Reliability	Maintainability	Availability
Constant	Decrease	Decrease
Constant	Increase	Increase
Increase	Constant	Increase
Decrease	Constant	Decrease

Finally, for answering the 'When' question is to have an appropriate definition of availability. The definition of availability can be flexible, depending on the types of downtimes and actor's points of view considered in the availability analysis. Therefore, different definition and classifications of availability can be offered. A classification of availability types has been proposed by [91, 92]. In the following, four common types of availability will be introduced briefly:

- Instantaneous availability

Instantaneous availability is defined as the probability that a system is operating at any given time. This definition is somehow similar to the reliability function definition which gives the probability that a system will function at the given time t. However, different from the reliability definition, the instantaneous availability contains information on maintainability. According to the instantaneous availability, the system will be operational provided the following conditions could be met:

The required function can be properly provided during time t with probability R(t);

or

It can provide the required function since the last repair time like u, $0 < u < t$, with the following probability:

$$\int_{0}^{t} R(t-u)m(u)du \tag{2}$$

where m (u) is the system renewal density function.

The instantaneous availability is calculated as:

$$A(t) = R(t) + \int_{0}^{t} R(t-u)m(u)du \tag{3}$$

- Mean availability

The average uptime availability or mean availability can be defined as the proportion of mission time or time period that a system is available for use. The mean value of the instantaneous availability over a specific period of time such as (0, T), is defined by this availability.

$$A_m(T) = \frac{1}{T}\int_{0}^{T} A(t)dt \tag{4}$$

- Steady-state availability

The steady state availability can be determined by calculating the limit of the instantaneous availability as the time goes to infinity. In fact, when the time approximates to about four times the MTBF in instantaneous availability, it can be said that the instantaneous availability function approaches the steady state value. Therefore, the steady state availability can be calculated by using the following equation:

$$A(\infty) = \lim_{T \to \infty} A(T) \tag{5}$$

- Operational availability

Operational availability can be considered as an important measurement for evaluating the system effectiveness and performance. It evaluates the system availability which includes all the downtime sources, such as diagnostic downtime, administrative downtime, logistic downtime, etc. The operational availability is calculated as:

$$A_0 = \frac{\text{Uptime}}{\text{Operating cycle}} \tag{6}$$

where the operating cycle is the overall time of the investigated operation period whereas Uptime is the system's total functional time during the operating cycle. It is important to note that the availability that a customer actually experiences in the system is operational availability which is the availability according to the events that happened to the system.

How?

"How to provide high availability and reliability while preventing performance degradation or supporting graceful degradation?"

As the last important step that should be taken, suitable solutions will be chosen to provide HA and reliability, while supporting graceful degradation based on the previous results as shown in Fig. 6. Table 6 presents fault tolerance (FT) methods classification in cloud computing environments.

By identifying the states which have higher failure rate and specifying the causes based on the results obtained from "Where?", "Which?" and "When?" steps, we will be able to provide high available and reliable services in our cloud computing systems. Therefore, appropriate action can be taken according to the specific occurrence and failure type because there is no one-size-fits-all solution in the HA and reliability issues area.

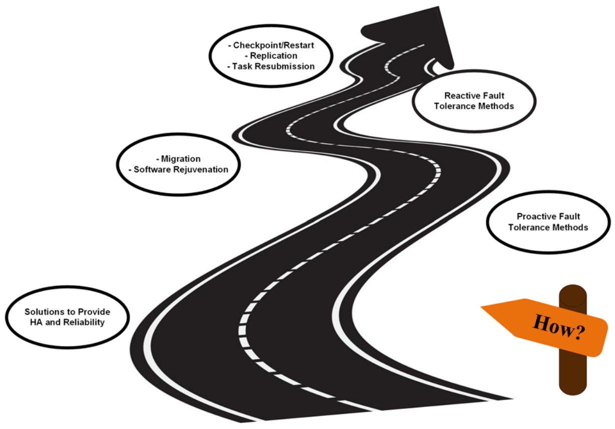
- Checkpoint/Restart
- Replication
- Task Resubmission
Reactive Fault Tolerance Methods
- Migration
- Software Rejuvenation
Proactive Fault Tolerance Methods
Solutions to Provide HA and Reliability
How?

Fig. 6 Proposed roadmap of the "How" step

Table 6 Fault tolerance methods in cloud computing systems [93, 94]

FT policy	FT technique	Description
Proactive FT	Preemptive migration	Preemptive migration involves suspending a process, recording its state, transferring it to another node and resuming operation of the process in the new node. It makes use of a feedback-loop control system where applications are constantly monitored and analyzed
	Software rejuvenation	Software rejuvenation technique can be applied proactively as inescapably software aging can lead to the software systems failures. In fact, it is a technique in which periodic reboots are scheduled for the system. After each reboot, the system resumes with a clean state
Reactive FT	Checkpointing/restart	Application checkpoint/restart technique allows saving the state of a running application to resume its execution later from the time at which it was checkpointed, on any arbitrary machine After a failure has occurred, the application software will be restarted from the point of failure, instead of rerunning the whole application from the scratch. It is an efficient fault tolerance technique for high computation intensive applications hosted in the cloud
	Replication	Replication is one of the most popular techniques which can be used according to the reactive policy. In cloud computing fault tolerance techniques, replication can be applied by keeping multiple replica of data and services. So, when an incoming request is received, it can be handled by a set of available replicas. Several different replicas are running through different computing resources to complete the requested task
	Task resubmission	The failed task can be resubmitted either to the same or to a different host at system runtime without any interruption during the system workflow

Fault tolerance mechanisms deal with quick repairing and replacement of faulty components for retaining the system. FT in cloud computing systems is the ability to withstand the abrupt changes which occur due to different types of failures. Recovery point objective (RPO) and recovery time objective (RTO) are two major and important parameters in fault management study. The RPO shows the amount of data to be lost as a result of a fault or disaster, whereas RTO shows the minimum downtime for recovering from faults [93].

Ganesh et al. [93] presented a study on fault tolerance mechanisms in cloud computing environments. As noted in this paper, there are mainly two standard FT policies available for running real-time applications in the cloud; they are "Proactive Fault Tolerance Policy" and "Reactive Fault Tolerant Policy". These FT policies are mainly used to provide fault tolerance mechanisms in the cloud.

The proactive fault tolerance policy is used to avoid failures by proactively taking preventive measures [95–97]. These measures are reserved by studying the pre-fault indicators and predicting the underlying faults. The next step is applying proactive fault tolerance measures by refactoring the code or failure prone components replacement at the development time. Using proactive fault tolerance mechanisms can guarantee that a job execution will be completed without any further reconfiguration [98].

The principle of reactive fault tolerance policy is based on dealing with measures applied for reducing the effects of the faults that already occurred in the cloud system. Table 6 shows a summary of this study on FT policies and techniques in cloud computing systems.

Although cloud computing systems have improved significantly over the past few years, improvements in computing power depend on system scale and complexity.

Modern processing elements are extremely reliable, but they are not perfect [99]. The overall failure rate of high-performance computing (HPC) systems such as cloud systems, increases with their size [99]. Therefore, fault tolerating methods are important for systems with high failure rate.

Proactive FT policies in cooperation with failure prediction mechanisms can avoid failures by taking proactive actions. The recovery process overheads can be significantly reduced by using this technology [99]. Task migration is one of the widely used techniques in Proactive FT techniques. While task migration can avoid failures and achieve much lower overheads than Reactive FT techniques like Checkpointing/Restart, there are two shortcomings in this technique. First, migration will not be performed if there is no spare node for accommodating the processes of the suspect node. The second problem is that task migration is not an appropriate method for software errors [100]. Thus, a proactive fault tolerance method which only rely on migration cannot gain all advantages of the failure prediction.

One of the other techniques to provide fault tolerance in cloud environments is by using VM migration. Jung et al. [101] propose a VM migration scheme using Checkpointing to solve the task waiting time problem. A fault tolerant mechanism usually triggers the VM migration before a failure event. It actually backs VM to the same physical server after system maintenance ends [102].

Redundancy is a common technique for providing high available and reliable systems [103], but it could not be as effective as expected as a result of particular types of failures such as transient failures and software aging. The most common procedure for combating software aging is to apply software rejuvenation. A software rejuvenation process can be done by restarting the software application. Therefore, this technique will help to restart the application to its standard level of performance and effectively solve the software aging problem. In other words, software rejuvenation is a preventive and proactive technique which is particularly useful for counteracting the appearance of software aging, aimed at cleaning up the internal states of the system to prevent further and future failure events. Without any rejuvenation, both the OS software and the hosted running application software will be degraded in performance with time due to the exhaustion of system resources such as free memory that could finally lead to a system crash, which is very undesirable to HA systems and fatal to mission-critical applications. The disadvantage of rejuvenation is the temporary outage of service. To completely eliminate the service outage during the rejuvenation process, the use of virtualization technology is one of the possible approaches [104].

The software rejuvenation process can be applied to the system according to the different parameters of software aging or the elapsed time since the last rejuvenation event. Software rejuvenation can be used at different scopes like system, application, process, or thread [105].

A classification of techniques has been proposed by [106] which distinguishes between two classes of failure handling techniques, namely, task level failure handling and workflow level failure handling. The recovery techniques that can be performed at the task level for masking the fault effects are called task level techniques. These types of recovery techniques have been widely studied in distributed and parallel environments. These recovery techniques are classified into four different types: retry, alternate resource, checkpoint/restart, and replication techniques. Generally, it can be concluded that the two simplest task level techniques are retry and alternate, as they simply try to execute the same task on the same or alternate resource again after failure.

The checkpoint is mostly efficient for long-running applications. The checkpointing approach has attracted significant attention over the recent years in the context of fault tolerance research like studies that have been carried out by [107–110]. Generally, the checkpoint technique periodically saves the state of an application. After the checkpoint, failed tasks can restart from the failure point by moving the task to another resource. There are several different checkpointing solutions for large-scale distributed computing systems. A classification of checkpointing mechanisms have been proposed in [111]. It shows that these mechanisms can be classified based on four different viewpoints. From one viewpoint, the checkpointing mechanism can be classified into two groups called the incremental checkpointing mechanisms and the full checkpointing mechanisms, depending on whether the newly modified page states are saved or the entire system running states are stored. The second viewpoint classifies the checkpointing mechanisms into two local and global checkpointing mechanisms, according to how checkpointing data is saved, locally or globally. Depending on whether the checkpointing data is saved coordinately or not, the third classification can be proposed, which is organized into two coordinated checkpointing mechanisms and uncoordinated checkpointing mechanisms. Finally, it can be concluded that the last classification is based on saving the checkpointing data on disk or not, and they are called disk and diskless checkpointing mechanisms.

Finally, as the last task level technique, we point to the replication. Running the same task on different computing resources and simultaneously ensures that there is at least one successful completed task which is the replication technique [112, 113].

As previously mentioned, workflow-level techniques are the second class of failures handling in distributed systems. Manipulating workflow structure for dealing with erroneous conditions is referred as workflow level technique. In other words, workflow level FTTs change the flow of execution on failure according to the knowledge of task execution context. They can also be classified into four different types: alternate task, redundancy, user defined exception handling and rescue workflow [106]. The only difference between alternate task and retry technique is that alternate task exchanges a task with a different implementation of the same task with different execution characteristics on the failure of the first one. The rescue workflow technique allows the workflow to continue even when task failure occurs. In another technique, the particular treatment to the task workflow is specified by the user and it is called user defined exception handling technique.

As a conclusion of the previous discussion, it can be said that both proactive and reactive fault tolerance policies have advantages and disadvantages. The results of the experiment show that migration techniques (proactive FT policy) are more efficient than checkpoint/restart techniques (reactive FT policy) [98]. Even though proactive techniques are more efficient, they are not frequently used. This is because the system is less affected by incorrect predictions due to proactive fault tolerance and also reactive methods are relatively simple to implement as FT techniques are not applied during the development time.

There is a serious gap in cloud computing fault tolerance research area, that is, the mutual impact of cloud performance overhead [52, 114–117] and HA solutions.

There are hot topics like VM Granularity, IaaS Routing Operations (such as live VM migration, concurrent deployment and snapshotting of multiple VMs …), and the inability to isolate shared cloud network and storage resources in the cloud performance overhead research area. All these factors can impact on VM performance, total system performance and therefore VM's availability and reliability. On the other hand, using additional mechanisms for providing HA and reliability in cloud computing systems can be considered as a system overhead which can have negative effects on system performance.

It seems that interesting and desired results would be achieved as the contribution of this research step by studying these mutual impacts for providing highly available and utilized cloud computing systems and considering performability concerns.

Discussion and open issues

Many research works have been carried out on cloud computing HA and reliability, but there is no comprehensive and complete overview of the entire problem domain. Attempt was made to propose a reference roadmap that covers all the aspects of the problem from different cloud actor's viewpoints, especially cloud consumers and providers. This study proposed a big picture which divided the problem space into four main steps to cover the various requirements in the desired research area. A specific question was posed for each step that answering these questions will enable cloud providers to offer high available and reliable services. Therefore, while cloud providers can satisfy the cloud consumers' requirements, they can also have highly utilized resources to achieve more business profits.

The four major questions starting with "Where?", "Which?", "When?" and "How" keywords form the main steps of the proposed roadmap. By taking these steps, our purposes can be achieved. In fact, taking each step means understanding the main concern presented by the specific question and proposing suitable solutions for that issue. Some suggestions were proposed as the primary answers for each of these questions which can be helpful for more future work in this research area. Making future research suggestions and proposing efficient solutions for each of these steps in the big picture is one of the important open issues in this study. Therefore, the proposed big picture can lead to future researches in cloud computing in the field of high availability and reliability.

Two main research gaps were specified through the proposed big picture, which have been neglected in the literature review. The first one is related to the providing HA and reliability solutions regardless of considering all cloud actors' requirements and viewpoints. By proposing "Where" question, attempt was made to consider both cloud consumers and providers which are most important actors in the cloud computing environments. Therefore, not only was attention given to have a high available and reliable system, but also provider's demands for having highly utilized systems were considered. The second research gap was proposed in "How" question section. As pointed out, adding an additional mechanism to the system can have negative effects on the total system performance. In addition, some performance overhead issues can degrade system availability and reliability. Therefore, this scientific hypothesis that system performance overhead and HA solutions can have the mutual impact on one another was proposed.

Conclusion

This paper presented a reference roadmap of HA and reliability problem in cloud computing systems that can lead to future research papers. By proposing this roadmap, all the different aspects of HA and reliability problem in cloud computing systems were specified. Furthermore, the effects of cloud computing main components on total system failure rate were studied according to the proposed eucalyptus-based architecture in our big picture. In this study, we focus on the comprehensive roadmap and covering all the different related aspects of HA and reliability in cloud environments. In addition, attempt was made to consider not only techniques ('When' and 'How' steps) that improve availability and reliability, but also characteristics ('Where' and 'Which' steps) of cloud computing systems. However, we will go into the details of specific technologies for each step as the future work of this study. Therefore, we can assess each step of the proposed solution in a more specific way. Evaluating the mutual impact of system performance overhead and HA solutions using the OpenStack platform is one of the main future works. SDN-based technology such as Espresso in Google cloud is one of the latest technology trends in cloud computing that can make cloud computing environments faster and more available. The application of this approach will be considered as a solution to 'How' step in future work. Furthermore, the contributions in this paper raise some other future works of reliability and HA in cloud computing through interesting proposed aspects of the problem. Some other significant future works are as follows:

- A comprehensive study of cloud failure modes, causes and failure rate, and reliability/availability measuring tools;
- A highly utilized and more profitable cloud economy that can guarantee the provision of highly available and reliable services.
- Evaluating availability and reliability of cloud computing system and components based on the proposed architecture;
- Studying the mutual impact of HA mechanisms and VM performance overhead.

Authors' contributions

All authors contributed equally to this manuscript. All authors read and approved the final manuscript.

Author details

[1] Department of Computer Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran. [2] Computer Science, University of Human Development, Sulaimanyah, Iraq. [3] Iran University of Medical Sciences, Tehran, Iran.

Acknowledgements

Not applicable.

Competing interests

The authors declare that they have no competing interests.

Availability of data and materials

Not applicable.

Funding

No funding was received.

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Received: 5 February 2018 Accepted: 16 June 2018
Published online: 06 July 2018

References

1. Ardagna D (2015) Cloud and multi-cloud computing: current challenges and future applications. In: 7th international workshop on principles of engineering service-oriented and cloud systems (PESOS) 2015. IEEE/ACM, Piscataway, pp 1–2
2. Rastogi G, Sushil R (2015) Cloud computing implementation: key issues and solutions. In: 2nd international conference on computing for sustainable global development (INDIACom). IEEE, Piscataway, pp 320–324
3. Mell P, Grance T (2011) The NIST definition of cloud computing. Commun ACM 53(6):50
4. Buyya R et al (2009) Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility. Future Gener Comput Syst 25(6):599–616
5. Puthal D et al (2015) Cloud computing features, issues, and challenges: a big picture. In: International conference on computational intelligence and networks (CINE). IEEE, Piscataway, pp 116–123
6. Mesbahi M, Rahmani AM (2016) Load balancing in cloud computing: a state of the art survey. Int J Mod Educ Comput Sci 8(3):64
7. Mesbahi M, Rahmani AM, Chronopoulos AT (2014) Cloud light weight: a new solution for load balancing in cloud computing. In: International conference (ICDSE) on data science and engineering. IEEE, Piscataway
8. Saab SA et al (2015) Partial mobile application offloading to the cloud for energy-efficiency with security measures. Sustain Comput Inf Syst 8:38–46
9. Keegan N et al (2016) A survey of cloud-based network intrusion detection analysis. Hum cent Comput Inf Sci 6(1):19
10. Younge AJ et al (2012) Providing a green framework for cloud data centers. Handbook of energy-aware and green computing-two, vol set. Chapman and Hall, UK, pp 923–948
11. Yuan H, Kuo C-CJ, Ahmad I (2010) Energy efficiency in data centers and cloud-based multimedia services: an overview and future directions. In: Green computing conference, 2010 international. IEEE, Piscataway
12. Zakarya M, Gillam L (2017) Energy efficient computing, clusters, grids and clouds: a taxonomy and survey. Sustain Comput Inf Syst 14:13–33
13. Zhang Q et al (2014) RESCUE: an energy-aware scheduler for cloud environments. Sustain Comput Inf Syst 4(4):215–224
14. Bielik N, Ahmad I (2012) Cooperative game theoretical techniques for energy-aware task scheduling in cloud computing. In: Proceedings of the 2012 IEEE 26th international parallel and distributed processing symposium workshops and Ph.D. forum. IEEE Computer Society, Piscataway
15. Zhu X et al (2016) Fault-tolerant scheduling for real-time scientific workflows with elastic resource provisioning in virtualized clouds. IEEE Trans Parallel Distrib Syst 27(12):3501–3517
16. Moon Y et al (2017) A slave ants based ant colony optimization algorithm for task scheduling in cloud computing environments. Hum Cent Comput Inf Sci 7(1):28
17. Motavaselalhagh F, Esfahani FS, Arabnia HR (2015) Knowledge-based adaptable scheduler for SaaS providers in cloud computing. Hum Cent Comput Inf Sci 5(1):16
18. Gajbhiye A, Shrivastva KMP (2014) Cloud computing: need, enabling technology, architecture, advantages and challenges. In: 2014 5th international conference confluence the next generation information technology summit (Confluence). IEEE, Piscataway, pp 1–7
19. Durao F et al (2014) A systematic review on cloud computing. J Supercomput 68:1321–1346
20. Modi C et al (2013) A survey on security issues and solutions at different layers of cloud computing. J Supercomput 63(2):561–592
21. Dubrova E (2013) Fault-tolerant design. Springer, Berlin
22. Shooman ML (2002) Reliability of computer systems and networks. Wiley, Hoboken
23. Dantas J et al (2015) Eucalyptus-based private clouds: availability modeling and comparison to the cost of a public cloud. Computing 97:1121–1140
24. Son S, Jung G, Jun SC (2013) An SLA-based cloud computing that facilitates resource allocation in the distributed data centers of a cloud provider. J Supercomput 64(2):606–637
25. Gagnaire M et al (2012) Downtime statistics of current cloud solutions. In: International working group on cloud computing resiliency. Tech. Rep. pp 176–189
26. Snyder B et al (2015) Evaluation and design of highly reliable and highly utilized cloud computing systems. J Cloud Comput Adv Syst Appl 4(1):11
27. Ranjithprabhu K, Sasirega D (2014) Eliminating single point of failure and data loss in cloud computing. Int J Sci Res (IJSR) 3(4):2319–7064
28. Tsidulko J (2017) The 10 biggest cloud outages of 2017 (So far). 2017; https://www.crn.com/slide-shows/cloud/300089786/the-10-biggest-cloud-outages-of-2017-so-far.htm. Accessed 1 Aug 2017
29. Celesti A, Fazio M, Villari M, Puliafito A (2016) Adding long-term availability, obfuscation, and encryption to multi-cloud storage systems. J Netw Comput Appl 59(C):208–218
30. Sampaio AM, Barbosa JG (2014) Towards high-available and energy-efficient virtual computing environments in the cloud. Future Gener = Comput Syst 40:30–43
31. Pérez-Miguel C, Mendiburu A, Miguel-Alonso J (2015) Modeling the availability of Cassandra. J Parallel Distrib Comput 86:29–44

32. An K et al (2014) A cloud middleware for assuring performance and high availability of soft real-time applications. J Syst Archit 60(9):757–769
33. Dwarakanathan S, Bass L, Zhu L (2015) Cloud application HA using SDN to ensure QoS. In: 8th international conference on cloud computing. IEEE, Piscataway, pp 1003–1007
34. Brenner S, Garbers B, Kapitza R (2014). Adaptive and scalable high availability for infrastructure clouds. In: Proceedings of the 14th IFIP WG 6.1. International conference on distributed applications and interoperable systems, vol 8460. Springer, New York, pp 16–30
35. Leslie LM, Lee YC, Zomaya AY (2015) RAMP: reliability-aware elastic instance provisioning for profit maximization. Journal Supercomput 71(12):4529–4554
36. Pan Y, Hu N (2014) Research on dependability of cloud computing systems. In: International conference on reliability, maintainability and safety (ICRMS). IEEE, Piscataway, pp 435–439
37. Hogan M et al (2011) NIST cloud computing standards roadmap. NIST Special Publication, 35
38. Kibe S, Uehara M, Yamagiwa M (2011) Evaluation of bottlenecks in an educational cloud environment. In: Third international conference on intelligent networking and collaborative systems (INCoS), 2011. IEEE, Piscataway
39. Arean O (2013) Disaster recovery in the cloud. Netw Secur 2013(9):5–7
40. Sharma K, Singh KR (2012) Online data back-up and disaster recovery techniques in cloud computing: a review. Int J Eng Innov Technol (IJEIT) 2(5):249–254
41. Xu L et al (2012) Smart Ring: A model of node failure detection in high available cloud data center. In: IFIP international conference on network and parallel computing. Springer, Berlin
42. Watanabe Y et al (2012) Online failure prediction in cloud datacenters by real-time message pattern learning. In: IEEE 4th International Conference on cloud computing technology and science (CloudCom), 2012. IEEE, Piscataway
43. Ongaro D et al (2011) Fast crash recovery in RAMCloud. In: Proceedings of the twenty-third ACM symposium on operating systems principles. ACM, Cascais
44. Amin Vahdat BK (2017) Espresso makes Google cloud faster, more available and cost effective by extending SDN to the public internet. https://www.blog.google/topics/google-cloud/making-google-cloud-faster-more-available-and-cost-effective-extending-sdn-public-internet-espresso/. Accessed 4 Apr 2017
45. Endo PT et al (2016) High availability in clouds: systematic review and research challenges. J Cloud Comput 5(1):16
46. Liu B et al (2018) Model-based sensitivity analysis of IaaS cloud availability. Future Gener Comput Syst
47. Torres E, Callou G, Andrade E (2018) A hierarchical approach for availability and performance analysis of private cloud storage services. Computing 1:1–24
48. Snyder B et al (2015) Evaluation and design of highly reliable and highly utilized cloud computing systems. J Cloud Comput 4(1):1
49. Jammal M, Kanso A, Heidari P, Shami A (2017) Evaluating High Availability-aware deployments using stochastic petri net model and cloud scoring selection tool. IEEE Trans Serv Comput (1):1–1
50. Sampaio AM, Barbosa JG (2017) A comparative cost analysis of fault-tolerance mechanisms for availability on the cloud. Sustain Comput Inf Syst. https://doi.org/10.1016/j.suscom.2017.11.006
51. Sharkh MA et al (2015) Simulating high availability scenarios in cloud data centers: a closer look. In: IEEE 7th international conference on cloud computing technology and science (CloudCom), 2015. IEEE, Piscataway
52. Xu F et al (2014) Managing performance overhead of virtual machines in cloud computing: a survey, state of the art, and future directions. Proc IEEE 102(1):11–31
53. Zhang Q, Boutaba R (2014) Dynamic workload management in heterogeneous Cloud computing environments. In: Network operations and management symposium (NOMS). IEEE, Piscataway
54. Touseau L, Donsez D, Rudametkin W (2008) Towards a sla-based approach to handle service disruptions. In: IEEE international conference on services computing, SCC'08. IEEE, Piscataway
55. Wang FZ, Zhang L, Deng Y, Zhu W, Zhou J, Wang F (2014) Skewly replicating hot data to construct a power-efficient storage cluster. J Netw Comput Appl 7(1):1–12
56. Xie T, Sun Y (2009) A file assignment strategy independent of workload characteristic assumptions. ACM Trans Storage (TOS) 5(3):10
57. Mesbahi MR, Rahmani AM, Hosseinzadeh M (2017) Highly reliable architecture using the 80/20 rule in cloud computing datacenters. Future Gener Comput Syst 77:77–86
58. Moor Hall C (2009) How to analyse your business sale—80/20 rule. The chartered institute of marketing, UK, pp 1–6
59. Foster I et al (2008) Cloud computing and grid computing 360-degree compared. In: Grid computing environments workshop, 2008. GCE'08. IEEE, Piscataway
60. Mishra AK et al (2010) Towards characterizing cloud backend workloads: insights from Google compute clusters. ACM SIGMETRICS Perform Eval Rev 37(4):34–41
61. Li X, Qiu J (2014) Cloud computing for data-intensive applications. Springer, Berlin
62. Jha S et al A tale of two data-intensive paradigms: applications, abstractions, and architectures. In: 2014 IEEE international congress on big data (BigData Congress), 2014. IEEE, Piscataway
63. Yang X et al (2014) Cloud computing in e-Science: research challenges and opportunities. J Supercomput 70(1):408–464
64. Barbar JS, Lima GDO, Nogueira A (2014) A model for the classification of failures presented in cloud computing in accordance with the SLA. In: International conference on computational science and computational intelligence (CSCI), 2014. IEEE, Piscataway
65. Vishwanath KV, Nagappan N (2010) Characterizing cloud computing hardware reliability. In Proceedings of the 1st ACM symposium on cloud computing. ACM, Indianapolis, pp 193–204
66. Serrano M (2012) Applied ontology engineering in cloud services, networks and management systems. Springer Science & Business Media, Berlin
67. Pan Y, Hu N (2014) Research on dependability of cloud computing systems. In: International conference on reliability, maintainability and safety (ICRMS), 2014. IEEE. Piscataway

68. Woods V (2015) Gartner reveals top predictions for it organizations and users for 2016 and beyond. http://www.gartner.com/newsroom/id/3143718. Accessed 6 Oct 2015
69. Price D Five high profile cloud-based failures. May 20, 2014; Available from: http://cloudtweaks.com/2014/05/five-high-profile-cloud-based-failures/
70. Gill P, Jain N, Nagappan N (2011) Understanding network failures in data centers: measurement, analysis, and implications. In: ACM SIGCOMM computer communication review. 2011, ACM, Toronto, pp 350–361
71. Serrano M, Orozco JMS (2012) Applied ontology engineering in cloud services, networks and management systems. Springer Science & Business Media, Berlin
72. Nightingale EB, Douceur JR, Orgovan V (2011) Cycles, cells and platters: an empirical analysis of hardware failures on a million consumer PCs. In: Proceedings of the sixth conference on Computer systems. 2011, ACM, New York, pp 343–356
73. Wang W, Loman J, Vassiliou P (2004) Reliability importance of components in a complex system. In: Reliability and maintainability, 2004 annual symposium-RAMS. IEEE, Piscataway, pp 6–11
74. Hasan O et al (2015) Reliability block diagrams based analysis: a survey. Analysis 1:1
75. Ni J, Tang W, Xing Y (2013) A simple algebra for fault tree analysis of static and dynamic systems. IEEE Trans Reliab 62(4):846–861
76. Behringer, B, Lehser M, Rothkugel S (2014) Towards feature-oriented fault tree analysis. In: 38th International computer software and applications conference workshops (COMPSACW). IEEE, Piscataway
77. Telek M, Horváth A, Horváth G (2004) Analysis of inhomogeneous Markov reward models. Linear Algeb Appl 386:383–405
78. Baier C et al (2010) Performability assessment by model checking of Markov reward models. Formal Methods Syst Des 36(1):1–36
79. Hong Z, Wang Y, Shi M (2012) CTMC-Based Availability Analysis of Cluster System with Multiple Nodes. In: Jin D, Lin S (eds) Advances in Future Computer and Control Systems. Advances in Intelligent and Soft Computing, vol 160. Springer, Berlin, Heidelberg
80. Leangsuksuna C, Shen L, Songa H, Scottb SL, Haddacf I (2003) The Modeling and dependability analysis of high availability OSCAR cluster system. In: Comptes Rendus Du 17ième Symposium Annuel International Sur Les Systèmes Et Applications Du Calcul de Haute Performance Et Le Symposium OSCAR. NRC Research Press, Ottawa, Canada, p 285
81. Veeraraghavan M, Trivedi K (1988) Hierarchical modeling for reliability and performance measures. Concurrent computations. Springer, Boston, MA, pp 449–474
82. Kim DS, Machida F, Trivedi KS (2009) Availability modeling and analysis of a virtualized system. In: 15th IEEE pacific rim international symposium on dependable computing, 2009. PRDC'09. IEEE, Piscataway
83. Ghosh R et al (2012) Interacting Markov chain based hierarchical approach for cloud services. Technical report, IBM (April 2010). http://domino.research.ibm.com/library/cyberdig.nsf/papers/AABCE247ECDECE0F8525771A005D42B6. Accessed Feb 2018
84. Che J et al (2011) A markov chain-based availability model of virtual cluster nodes. In: 2011 Seventh international conference on computational intelligence and security (CIS). IEEE, Piscataway
85. Zheng J, Okamura H, Dohi T (2012) In: Component importance analysis of virtualized system. In: 9th international conference on ubiquitous intelligence and computing and 9th international conference on autonomic and trusted computing (UIC/ATC), 2012. IEEE, Piscataway
86. Ghosh R et al (2014) Scalable analytics for IAAS cloud availability. IEEE Trans Cloud Comput 2(1):57–70
87. Ferrari A, Puccinelli D, Giordano S (2012) Characterization of the impact of resource availability on opportunistic computing. In: Proceedings of the first edition of the MCC workshop on Mobile cloud computing. ACM, New York
88. Chuob S, Pokharel M, Park JS (2011) Modeling and analysis of cloud computing availability based on eucalyptus platform for e-government data center. In: Fifth international conference on innovative mobile and internet services in ubiquitous computing (IMIS). IEEE, Piscataway
89. Wei B, Lin C, Kong X (2011) Dependability modeling and analysis for the virtual data center of cloud computing. In: 13th International conference on high performance computing and communications (HPCC). IEEE, Piscataway
90. Dantas J et al (2012) An availability model for eucalyptus platform: an analysis of warm-standy replication mechanism. In: International conference on systems, man, and cybernetics (SMC). IEEE, Piscataway
91. Relationship between availability and reliability. 2003. http://www.weibull.com/hotwire/issue26/relbasics26.htm. Accessed Feb 2018.
92. Katukoori VK (1995) Standardizing availability definition. University of New Orleans, New Orleans
93. Ganesh A, Sandhya M, Shankar S (2014) A study on fault tolerance methods in cloud computing. In: IEEE international advance computing conference (IACC), 2014. IEEE, Piscataway
94. Jhawar R, Piuri V, Santambrogio M (2013) Fault tolerance management in cloud computing: a system-level perspective. IEEE Syst J 7(2):288–297
95. Egwutuoha IP et al (2012) A proactive fault tolerance approach to high performance computing (HPC) in the cloud. In: second international conference on cloud and green computing (CGC). IEEE, Piscataway
96. Jhawar R, Piuri V, Santambrogio M (2013) Fault tolerance management in cloud computing: a system-level perspective. Syst J IEEE 7(2):288–297
97. Ji X, Ma Y, Ma R, Li P, Ma J, Wang, G et al (2015) A proactive fault tolerance scheme for large scale storage systems. In: International conference on algorithms and architectures for parallel processing. Springer, Cham, pp 337–350
98. Ganesh A, Sandhya M, Shankar S (2014) A study on fault tolerance methods in cloud computing. In: International advance computing conference (IACC). IEEE, Piscataway
99. Zhu L et al (2015) Optimizing the fault-tolerance overheads of HPC systems using prediction and multiple proactive actions. J Supercomput 71(10):3668–3694
100. Cappello F et al (2009) Toward exascale resilience. Int J High Perform Comput Appl 23:374–388
101. Jung D, Chin S, Chung KS, Yu H (2013) VM migration for fault tolerance in spot instance based cloud computing. In: International conference on grid and pervasive computing. Springer, Berlin, Heidelberg, pp 142–151

102. Ahmad RW et al (2015) Virtual machine migration in cloud data centers: a review, taxonomy, and open research issues. J Supercomput 71:2473–2515
103. Zhang J, Li S, Liao X (2016) REDU: reducing redundancy and duplication for multi-failure recovery inerasure-coded storages. J Supercomput 72(9):3281–3296
104. Yang C-T et al (2014) On improvement of cloud virtual machine availability with virtualization fault tolerance mechanism. Journal Supercomput 69(3):1103–1122
105. Yang M et al (2014) Software rejuvenation in cluster computing systems with dependency between nodes. Computing 96(6):503–526
106. Qureshi K et al (2011) A hybrid fault tolerance technique in grid computing system. J Supercomput 56(1):106–128
107. Nazir B, Qureshi K, Manuel P (2009) Adaptive checkpointing strategy to tolerate faults in economy based grid. Journal of Supercomput 50(1):1–18
108. Liu H et al (2012) VMckpt: lightweight and live virtual machine checkpointing. Science China Information Sciences 55(12):2865–2880
109. Du Y et al (2014) FITDOC: fast virtual machines checkpointing with delta memory compression. In: IEEE 17th international conference on computational science and engineering (CSE), 2014. IEEE, Piscataway
110. Losada N, Cores I, Martín MJ, González P (2017) Resilient MPI applications using an application-levelcheckpointing framework and ULFM. J Supercomput 73(1):100–113
111. Sun D et al (2013) Analyzing, modeling and evaluating dynamic adaptive fault tolerance strategies in cloud computing environments. J Supercomput 66(1):193–228
112. Nazir B, Qureshi K, Manuel P (2012) Replication based fault tolerant job scheduling strategy for economy driven grid. J Supercomput 62(2):855–873
113. Tos U et al (2015) Dynamic replication strategies in data grid systems: a survey. J Supercomput 71(11):4116–4140
114. Koh Y et al (2007) An analysis of performance interference effects in virtual environments. In: IEEE International symposium on performance analysis of systems and software, 2007. ISPASS 2007. IEEE, Piscataway
115. Wang P, Huang W, Varela CA (2010) Impact of virtual machine granularity on cloud computing workloads performance. In: 11th IEEE/ACM international conference on grid computing (GRID), 2010. IEEE, Piscataway
116. Liu X et al (2014) Performance analysis of cloud computing services considering resources sharing among virtual machines. J Supercomput 69(1):357–374
117. Yang B, Tan F, Dai Y-S (2013) Performance evaluation of cloud service considering fault recovery. J Supercomput 65(1):426–444