Failure Management for Reliable Cloud Computing: A Taxonomy, Model and Future Directions

Sukhpal Singh Gill and Rajkumar Buyya

Cloud Computing and Distributed Systems (CLOUDS) Laboratory,
School of Computing and Information Systems
The University of Melbourne, Australia

The next generation of cloud computing must be reliable to fulfil the end-user requirements, which are changing dynamically. Presently, cloud providers are facing challenges to ensure the reliability of their services. In this paper, we propose a comprehensive taxonomy of failure management in cloud computing. The taxonomy is used to investigate the existing techniques for reliability that need careful attention and investigation as proposed by several academic and industry groups. Further, the existing techniques have been compared based on the common characteristics and properties of failure management as implemented in commercial and open source solutions. A conceptual model for reliable cloud computing has been proposed along with discussion on future research directions. Moreover, a case study of astronomy workflow is presented for reliable execution in cloud environment.

Keywords: Cloud Computing, Failure Management, Resilience, Montage Workflow, Reliable Computing

1. INTRODUCTION

Cloud computing paradigm delivers computing resources residing in providers' datacentres as a service over the Internet. The prominent cloud providers such as Google, Facebook, Amazon and Microsoft are providing highly available cloud computing services using thousands of servers, which consists of multiple resources such as processors, network cards, storage devices and disk drives [1]. With the growing adoption of cloud, Cloud Data Centres (CDCs) are rapidly expanding their sizes and increasing complexity of the systems, which increases the resource failures. The failure can be Service Level Agreement (SLA) violation, data corruption and loss and premature termination of execution, which can degrade the performance of cloud service and affect the business [2]. For next generation clouds to be reliable, there is a need to identify the failures (hardware, service, software or resource), their causes and manages them to improve their reliability [2]. To solve this problem, a model and system is required that introduces replication of services and their coordination to enable reliable delivery of cloud services in cost-efficient manner.

The rest of the paper is organised as follows: Section 2 presents a systematic review of existing techniques for reliable cloud computing and proposed a failure management based comprehensive taxonomy. Further, based on the taxonomy, techniques have been compared. Section 3 presents the failure management in open source technologies. Section 4 presents the fault tolerance resilience in practice. Section 5 covers approaches for creating reliable applications using modular microservices and cloud-native architectures. Section 6 presents the resilience on Exascale systems. Section 7 presents the conceptual model for reliable cloud computing. Section 8 presents the fault tolerance for scientific computing applications along with a case study of astronomy workflow. Section 9 presents the future research directions. Finally, Section 10 concludes the paper.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/MCSE.2018.2873866, Computing in Science & Engineering

2. RELIABLE CLOUD COMPUTING: A JOURNEY AND TAXONOMY

Reliability in cloud computing is defined as "the ability of a cloud computing system to perform the desired task or (provide a required service) for stated time period under predefined conditions" [4]. The reliability of cloud computing system depends on the different layers of cloud architecture such as software, platform and infrastructure.

2.1 State-of-the-Art

This section briefly describes the existing research work of reliable cloud computing. Deng et al. [11] proposed a Reliability-aware Resource Management (RRM) approach for effective management of hardware faults in scientific computation, which improves the reliability of cloud service. Further, it has been proved that RRM is effective in providing reliability and fault-tolerance against the malicious attacks and failures. Lin and Chang [3] proposed a Maintenance Reliability Estimation (MRE) approach for cloud computing network to measure the maintenance of data transfer with nodes failure and time constraints. Further, sensitive analysis has been done to improve the transmission time and data transfer speed by selecting shortest and reliable paths. Dastjerdi and Buyya [4] proposed a SLA based Autonomous Reliability-aware Negotiation (ARN) approach to automate the negotiation process between cloud service providers and requesters. Moreover, ARN can evaluate the reliability of proposals received from service providers. The proposed approach reduces the underutilization of resources and enables the parallel negotiation with many resource providers simultaneously. Xuejie et al. [5] developed a Hybrid Method based Reliability Evaluation (HMRE) model, which combines Continuous-Time Markov Chain (CTMC) and Mean Time To Failure (MTTF) metrics to measure the effect of physical-resource breakdowns on system reliability. HMRE model can be used to design a reliable system for cloud computing.

Chowdhury and Tripathi [6] proposed a security based Reliability-aware Resource Scheduling (RRS) technique to measure the reliability of cloud datacenter. Moreover, RRS updates the reliability of cloud resources continuously for further scheduling of resources for the execution of user workloads. Cordeschi et al. [7] developed an Adaptive Resource Management (ARM) model to improve the reliability of cloud services in cloud-based cognitive radio vehicular networks. ARM manages the resources effectively and provides the energy-efficient cloud service to perform traffic offloading. The distributed and scalable deployment of ARM offers the hard reliability guarantees to transfer data using wireless sensor network. Zhou et al. [8] proposed a Cloud Service Reliability Enhancement (CSRE) technique to improve the storage and network resource utilization. CSRE uses service checkpoint to store the state of all the Virtual Machines (VMs), which are currently processing user workloads. Further, a node failure predicator is developed to reduce the network resource consumption.

Li et al. [9] proposed a convergent dispersal based multi-cloud storage (CDStore) solution to provide the cost-effective, secure and reliable cloud service. CDStore provides deterministic-based deduplication to improve storage and bandwidth savings, which further protects the system from malicious attacks using two-stage deduplication. Azimzadeh and Biabani [10] proposed a Multi-Objective Resource Scheduling (MORS) mechanism to reduce execution time and improve reliability of cloud service. Further, a trade-off between execution and reliability has been established for the execution of High Performance Computing (HPC) workloads.

Calheiros and Buyya [13] proposed a Task Replication-based Resource Provisioning (TRRP) algorithm for execution of deadline-constrained scientific workflows. TRRP utilizes the extra budget and free time of resources to execute workflows within their deadline and budget. Poola et al. [14] proposed a spot and on-demand instances-based Adaptive and Just-In-Time (AJIT) scheduling algorithm to offer fault tolerance. AJIT minimizes execution cost and time through resource consolidation and experimental results prove that AJIT is an effective in execute workloads under short deadlines. Qu et al. [15] proposed a Heterogeneous Spot Instances-based Auto-scaling (HSIA) fault tolerant system for execution of web applications, which effectively reduces the cost of execution and improves the availability and response time. Liu et al. [16] proposed a replication-based state management system (E-Storm) for execution of streaming applications. E-Strom uses multiple state backups on different worker nodes to improve reliability of the system and performs better the existing techniques in terms of latency and throughput. Abdulhamid et al. [21] proposed a Dynamic Clustering League Championship Algorithm (DCLCA) based fault management technique, which schedule tasks on cloud resources for execution and focuses on fault reduction in task failure. The experimental results show that DCLCA performs better in terms of makespan and fault rate. Figure 1 shows the evolution of existing techniques for reliable cloud computing and their focus of study.

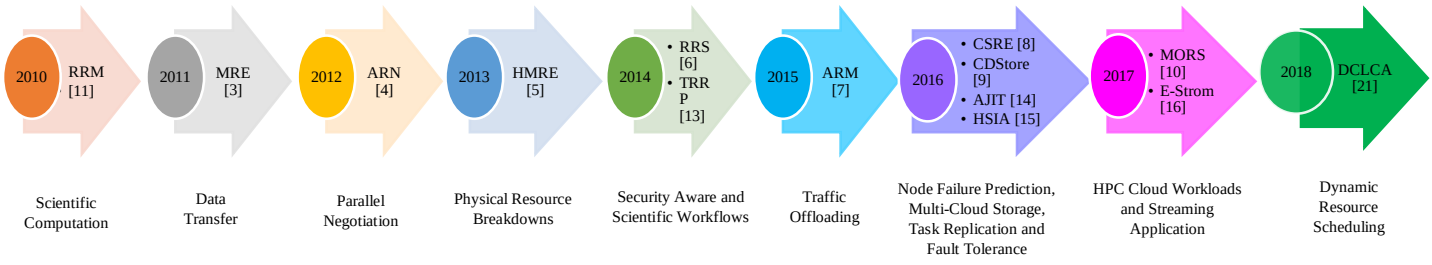

Figure 1: Evolution of Reliable Cloud Computing

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/MCSE.2018.2873866, Computing in Science & Engineering

2.2 Failure Management

To offer reliable cloud services, there is a need of an effective management of failures. Literature [14-20] reported that various failure management techniques and policies have been proposed for reliability assurance in cloud computing. A *failure* is defined as "when a cloud computing system fails to perform a specific function according to its predefined conditions". We have identified four types of failures (service failure, resource failure, correlated failure and independent failure) and classified these failures in into two main categories: 1) architecture based and 2) occurrence based. Table 1 describes the classification of failures and their causes.

Table 1: Classification of Failures and their Causes

Type of Failures	Classification	Cause of Failure	Percentage of Occurrence of Failure[1,2,3,4]
Service Failure	Architecture Based	• Software Failure ➢ Complex Design ➢ Software Updates ➢ Planned Reboot ➢ Unplanned Reboot ➢ Cyber Attacks • Scheduling ➢ Timeout ➢ Overflow	18%
Resource Failure		• Hardware Failure ➢ Complex Circuit Design ➢ Memory ➢ RAID Controller ➢ Dis Drive ➢ Network Devices ➢ System Breakdown ➢ Power Outage	58%
Correlated Failure	Occurrence Based	• Based on Spatial Correlation between Two Failures • Based on Temporal Correlation between Two Failures	14%
Independent Failure		• Denser System Packing • Human Errors • Heat Issue	10%

[1] https://blogs.gartner.com/thomas_bittman/2015/02/05/why-are-95-of-private-clouds-failing/
[2] https://esj.com/articles/2014/06/26/cloud-projects-fail.aspx
[3] http://www.datacenterknowledge.com/archives/2008/05/30/failure-rates-in-google-data-centers
[4] https://docs.microsoft.com/en-us/aspnet/aspnet/overview/developing-apps-with-windows-azure/building-real-world-cloud-apps-with-windows-azure/design-to-survive-failures

2.2.1 Taxonomy

Based on failure management techniques and policies for reliability assurance in cloud computing, the components of the taxonomy are: 1) design principle, 2) QoS, 3) architecture, 4) application type, 5) protocol and 6) mechanism (see Figure 2).

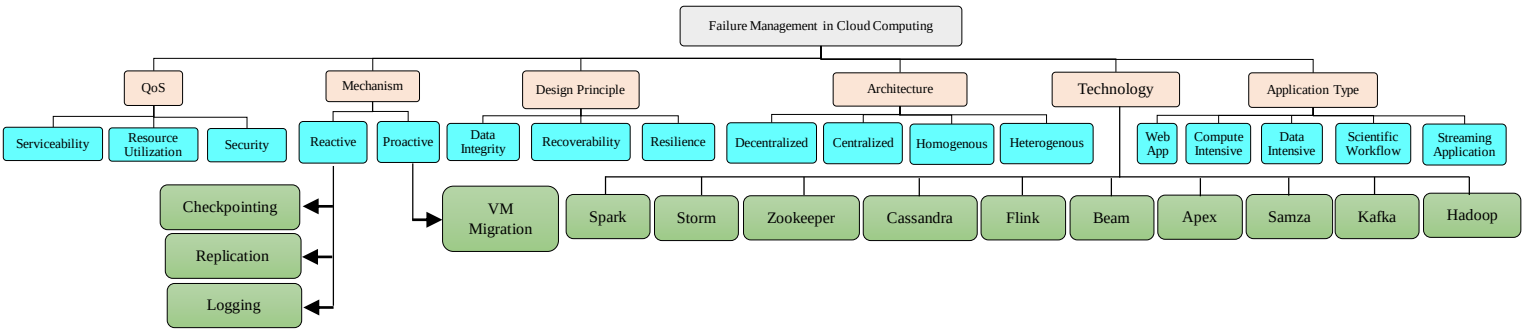

Figure 2: Taxonomy based on Failure Management in Clouds

2.2.1.1 Design Principle: Three different type of design principles are proposed for reliable cloud service such as: 1) design for *recoverability* i.e. recover system with minimum involvement of human, 2) design for *data integrity* i.e. to ensure the accuracy and consistency of data during transmission and 3) design for *resilience* i.e. enhance system resilience and reduce the effect of failure to there is lesser interruption to cloud service.

2.2.1.2 Quality of Service (QoS): Three QoS parameters are considered to measure the reliability of cloud service [12]: serviceability, resource utilization and security. *Serviceability* is defined in (Eq. 1), while *resource utilization* is defined in (Eq. 2). *Security* in cloud computing is a deployment of technologies or policies to protect infrastructure, applications and data from malicious attacks [2].

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/MCSE.2018.2873866, Computing in Science & Engineering

$$Serviceability = \frac{\text{Service Uptime}}{\text{Service Uptime} + \text{Service Downtime}} \quad (1)$$

$$Resource\ Utilization = \frac{\text{Actual Time Spent by a Resource to Execute Workload}}{\text{Total Uptime of a Resource}} \quad (2)$$

2.2.1.3 Architecture: There are four types of architecture: homogenous, heterogenous, centralized and decentralized. A *homogenous* architecture has the same type of configuration, such as operating systems, networking, storage and processors, while a *heterogeneous* datacenter combines different type of configurations of operating systems, networking, storage and processors to process user applications. In *centralized* architectures, there is a central controller, which manages all the tasks that are required to be executed, and further it executes the task using scheduled resources. The central controller is responsible for the execution of all tasks. In *decentralized* architectures, resources are allocated independently to execute the tasks without any mutual coordination. Every resource is responsible for their own task execution.

2.2.1.4 Application Type: For application management, there are five types of application that are considered for reliable cloud computing: web applications, streaming applications, compute-intensive, data-intensive and scientific workflows. The applications that can execute anytime but its execution should be completed before their deadline are called *compute-intensive* like HPC. Web applications are those applications which are required to run all time i.e. 24 X 7 like delay torrent, Internet services etc. The applications with lot of data crunching is called *data-intensive*. In *scientific* workflows, real-world activities can be simulated like flight control system, weather prediction and climate modelling, aircraft design and fuel efficiency, oil exploration etc., which requires high processing capacity to execute user requests. A *streaming application* is a program, which downloads the required components instead of installing components before its use and it is used to provide virtualized applications.

2.2.1.5 Mechanism: There are two types of mechanisms: reactive and proactive. *Reactive* management works based on feedback methods and manages the system based on their current state to handle faults. There is a need of continuous monitoring of resource allocation to track the system status. If there is some system error then corrective action will be taken to manage that fault. *Proactive* management manages the system based on the future prediction of the performance of the system instead of its current state. The resources are selected based on the previous executions of the system in terms of reliability, throughput etc. The predictions are required to be identified based on previous data, and plan their appropriate action to manage that fault during system execution.

2.2.1.6 Protocol: The mechanisms are further divided into different protocols: checkpointing, replication, logging and VM migration. To incorporate fault tolerance into system, a snapshot of the application's state is saved, so that system can reboot from that point in case of system crash, this process is called *checkpointing*. To improve the reliability of system, information is shared among redundant resources (hardware or software), is called *replication*. *Logging* is required to save the information related to cyber-attacks, auditing, anomalies, user access, troubleshooting etc. to building a reliable system. Failure can be avoided proactively by migrating the VM from one cloud datacenter to another is called *VM migration*.

The various open source technologies use by different reliability-aware approaches that are discussed in *Section 3*. Table 2 shows the comparison of reliability-aware approaches based on taxonomy of failure management.

Table 2: Comparison of Reliability-aware Approaches based on the Taxonomy

Technique	Author	Design Principle	QoS	Architecture	Application Type	Mechanism	Protocol	Technology	Open Issues
RRM	Deng et al. [11]	Design of Resilience	Serviceability	Decentralized	Scientific workflows	Reactive and Proactive	Logging and VM Migration	Hadoop	Privacy protection for cloud user information is not provided.
MRE	Lin and Chang [3]		Serviceability	Heterogenous	Data-Intensive	Reactive	Checkpointing	Spark	Secure data transmission paths are required.
TRRP	Calheiros et al. [13]		Serviceability	Centralized	Scientific Workflows	Reactive	Replication	Storm nad Hadoop	Execution cost can be reduced.
DCLCA	Abdulhamid et al. [21]		Resource Utilization	Centralized	Web Applications	Reactive	Replication	Kafka	Execution cost is not considered.
ARN	Dastjerdi and Buyya [4]	Design of Recoverability	Security and Resource Utilization	Homogenous	Scientific workflows	Reactive	Replication	Zookeeper	The effect of heterogeneous negotiation on the profit is needed to be analysed.
HMRE	Xuejie et al. [5]		Security and Serviceability	Centralized	Web Applications	Proactive	VM Migration	Cassandra	Resource utilization is not considered.
RRS	Chowdhury and Tripathi [6]		Security and Resource Utilization	Heterogenous	Compute-Intensive	Reactive	Checkpointing	Flink and Hadoop	This technique only considers homogenous workloads.
ARM	Cordeschi et al. [7]		Security and Serviceability	Homogenous	Compute-Intensive	Proactive	VM Migration	Beam and Hadoop	The bandwidth efficiency of network is required to be improved.
AJIT	Poola et al. [14]		Resource Utilization	Decentralized	Scientific workflows	Reactive	Replication	Apex and Zookeeper	Secure cloud services are required.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/MCSE.2018.2873866, Computing in Science & Engineering

HSIA	Qu et al. [15]		Serviceability	Heterogenous	Web Applications	Reactive	Replication	Samza and Strom	Resource utilization can be considered.
CSRE	Zhou et al. [8]	Design for Data Integrity	Resource Utilization	Decentralized	Web Applications	Reactive	Checkpointing	Spark	Resource utilization is lesser.
CDStore	Li et al. [9]		Resource Utilization	Centralized	Data-Intensive	Reactive and Proactive	VM Migration	Storm	Backup restore mechanism is a time-consuming process.
MORS	Azimzadeh and Biabani [10]		Serviceability and Resource Utilization	Homogenous	Compute-Intensive (HPC)	Proactive	VM Migration	Hadoop	Secure cloud services are required.
E-Strom	Liu et al. [16]		Serviceability and Resource Utilization	Centralized	Streaming Application	Reactive	Replication	Zookeeper and Hadoop	Execution cost can be reduced.

3. FAILURE MANAGEMENT IN OPEN SOURCE TECHNOLOGIES

In literature [5-15], the various types of open source technologies are identified for failure management in reliability-aware approaches such as Hadoop, Storm, Spark, Kafka, Zookeeper, Cassandra, Flink, Beam, Ape and Samza. Table 3 presents the description of open source technologies along with their comparison based on different parameters such as type of service, their features, language used to develop technology, type of data processing and fault tolerance mechanism by different technologies.

Table 3: Comparisons of Open Source Technologies based on Different Parameters

Name	Description	Type of Service	Feature	Language Used	Data Processing	Fault Tolerance Mechanism (FTM)
Hadoop	It uses different systems to handle massive amounts of data and computation	Data storage, data processing, data governance and security	Map-Reduce programming model based distributed storage and processing of big data	Java	Batch	Hadoop uses Hadoop Distributed File System (HDFS) to handle faults by the process of replica creation and data can be accessed from replication.
Spark	It provides APIs in Java, Scala and Python to allow data workers to execute streaming using in-memory.	To build applications that exploit machine learning and graph analytics	Runs iterative Map-Reduce jobs	Scala	Stream	Spark uses Resilient Distributed Dataset (RDD) to replicate data among multiple Spark executors in worker nodes in the cluster.
Storm	It processes unbounded streams of data.	Stream processing, continuous computation and distributed remote procedure call	Scalable and real-time computation systems	Clojure[1] & Java	Stream	Storm restarts automatically if a node dies, the worker will be restarted on another node and resets it to the latest successful checkpoint.
Kafka	It builds real-time data pipelines and streaming applications.	Message passing	High throughput, low latency and persistent messaging	Scala	Stream	Kafka maintains replication of data on a regular basis and cluster manager restarts automatic driver in case of failure and use checkpointing mechanism to start data processing form the place when it crashed.
Zookeeper	It is a centralized service for keeping configuration information and offers distributed synchronization.	i) Enables coordination using Locks and Synchronization and ii) naming service	Provides hierarchical namespace and form cluster of nodes	Java	Hybrid	It maintains replication using multiple servers and it makes client-server model for servers, which works in coordination manner to handle failure.
Cassandra	It handles a massive amount of data across many commodity servers	Provides high availability with no single point of failure	Low latency and masterless replication	Java	Hybrid	It maintains data replication and then it repairs the crashed node or replace with more reliable node while maintaining the consistency
Flink	It executes arbitrary dataflow programs in a data-parallel and pipelined manner.	Performs data analytics using machine learning algorithms	High-throughput and low-latency stream processing	Java and Scala	Stream	It captures consistent snapshots of the operator state and distributed data stream and which will act as checkpoints in case of failure
Beam	It defines and executes data processing workflows	Analyses data streams to solve real-world challenges of stream processing	Execute pipelines on multiple execution environment	Java and Python	Hybrid	The logging of the current pipeline state used for fault tolerance
Apex	It processes distributed big data-in-motion for real-time analytics	Distributed data processing	Scalable and secure	Java and Scala	Hybrid	It maintains checkpoints automatically and it recovers failed containers using Heartbeat mechanism [11].
Samza	It provides distributed stream processing using a separate Java Virtual Machine (JVM) for each stream processor container	Message passing	It runs multiple stream processing threads within a single JVM	Java and Scala	Stream	Whenever a machine in the cluster fails, Samza works with Yet Another Resource Negotiator (YARN) to transparently migrate user tasks to another reliable machine.

[1]Cloujure is a dynamic programming language for multithreading and it runs on Java virtual machine

4. FAULT-TOLERANCE AND RESILIENCE IN PRACTICE

There are various commercial clouds such as Amazon Web Services, Window Azure, Google App Engine, IBM Cloud, and Oracle, which focuses on fault tolerance to deliver reliable cloud service. In this section, we have explored the recent advances of commercial cloud providers based on eight different types of fault tolerance parameters [5] [6] [11] [13] [14] [17] [18] [22]. To improve the reliability of system, information is shared among redundant resources (hardware or software), is called *replication*. The capability of a system to deliver *24×7* service in case of failure - a disk, a node or a network is called availability. The capability of a system to protect against data loss during write, read, and rewrite operations on storage media is called *durability*. *Archiving-cool storage* means lower cost tier for storing data which is accessed infrequently and long-lived. *Backup* offers off-site protection against data loss by allowing data to be backed-up and recovered from the cloud at later stage. Disaster recovery

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/MCSE.2018.2873866, Computing in Science & Engineering

provides automatic replication and protection of VMs using recovery plans and its testing. *Relational database* provides organization of data to develop data-driven websites and applications without demanding to manage infrastructure. Caching offers effective storage space, which is used to off-load non-transactional work from a database. Table 4 shows the comparison of commercial clouds based on fault tolerance parameters.

Table 4: Comparison of Commercial Clouds based on Fault Tolerance Parameters

Cloud Provider	Replication Technique	Availability Zones	Durability Service	Archiving-Cool Storage	Backup	Disaster Recovery	Relational Database	Caching
Amazon Web Services	Zerto Virtual Replication	54 Availability Zones	Elastic Block Store (EBS)	Amazon Simple Storage Service (S3) Infrequent Access (IA) Glacier	Foolproof AWS Backup Strategy	Virtual Tape Library (VTL) and Virtual Tape Shelf (VTS)	Relational Database Service (RDS)	Elastic Cache
Windows Azure	Locally Redundant Storage (LRS) and Geo-Redundant Storage (GRS)	42 Availability Zones	Binary Large OBject (BLOB) Storage	Storage-Hot, Cool and Archive Tier	Volume Shadow Copy Service (VSS)	On-Site Recovery	SQL Database	Redis Cache
Google App Engine	Built-in Redundancy	45 Availability Zones	Google Cloud Storage	Google Cloud Storage Coldline	Snapshots	Google Cloud Storage Nearline	Google Cloud SQL	Memcache Cache
IBM Cloud	Zerto Virtual Replication	33 Availability Zones	Tivoli Storage Manager	IBM Cloud Object Storage standard, cold and vault tiers	Infraworx Cloud Backup	Off-Site Recovery	SQL Database	solidDB Universal Cache
Oracle	Snapshot Replication	23 Availability Zones	Enterprise Management Console (EMC) XremIO Optimized Flash Storage	Flashback Data Archive	CloudBerry Backup	Fusion Middleware Disaster Recovery	NoSQL Database	Oracle In-Memory Database Cache

5. RELIABILITY VIA MICROSERVICES AND CLOUD-NATIVE ARCHITECTURES

Microservice-based design of applications make them loosely coupled from other services, modular and independent. Therefore, a microservice will not impact on other services and thus improve the fault-tolerance and availability [7] of applications. To achieve fault-tolerance in microservice, it has to be designed with the following objectives: i) minimum interdependencies among services, ii) include built-in resilience using API gateway (e.g. Zuul) [8], iii) contains built in self-healing capabilities (e.g. Kubernetes) [9] and iv) protects against intermittent service failures or load spikes using cache request in stream processor (e.g. Apache Kafka) [11]. Further, automated testing mechanism should be incorporated to perform application testing with ultra-high loads or randomized input/wrong input, which can further improve the fault tolerance in microservices. There are two types of micro profiles can be used for microservice implementation for fault tolerance: CircuitBreaker and Fallback [23]. To prevent the repeated calls that likely to fail, CircuitBreaker service permits microservice to fail instantly. After main service failure, Fallback service runs to offer failure or may continue operation of the original microservice.

Cloud-native architectures enable the creation of applications using IaaS (Infrastructure-as-a-Service) and PaaS (Platform-as-a-Service) capabilities and services supported by Cloud computing platforms. Such applications are called Cloud-native applications [29], as they seamlessly benefit from reliability, scalability, and elasticity features offered by PaaS platforms. Moreover, many Cloud PaaS platforms are designed to run on a variety of computing infrastructures, from networked desktop computers to public Clouds. That means, engineering reliable system applications becomes easier, seamless, and cost-effective. For example, application designed using Cloud PaaS platforms such Aneka [28] can run on networked desktop computers within an enterprise, leased resources from public Clouds, or hybrid Clouds by harnessing both enterprise and public Cloud resources along with seamlessly benefiting from reliable and cost-efficient execution services offered by the platform.

6. RESILIENCE ON EXASCALE SYSTEMS

Exascale systems uses multicore processors to offer massive parallelism, which executes more than thousand floating point operations per second. The probability of partial failures will be increased due to participation of large number of heterogenous functional components such as network interfaces, memory chips and computing cores [3]. Therefore, fault tolerance at system level is required to handle dynamic reconfigurations at runtime. In past, checkpoint/restart technique is used to prevent computation to be lost due to failures for long running jobs, but this technique is not very effective due to slow communication channels between RAM and parallel file system [5]. Replication can be used in addition to checkpoint/restart to improve fault tolerance. In replication, same computation is performed by multiple processors, therefore, processor failure does not affect application execution [24]. There are two different types of approaches for replication has been developed: 1) process replication and 2) instance replication. In process replication, it replicates every process in a single instance of a parallel application while in instance replication, it replicates the instances of entire application. The trade-off between power consumption and cost for resilience on Exascale systems is an open issue.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/MCSE.2018.2873866, Computing in Science & Engineering

7. A CONCEPTUAL MODEL FOR RELIABLE CLOUD SERVICE

Figure 3 shows the conceptual model for reliable cloud computing in the form of layered architecture, which offers effective management of cloud computing resources, to make cloud services more reliable. The three main components of proposed architecture are discussed below:

1. *Cloud Users:* At this layer, cloud user submits their requests and defines required services in terms of SLA. Workload manager is deployed to handle the incoming user workloads, which can be interactive or batch style and transfer to the middleware for resource provisioning.

2. *Middleware:* This is the main layer of model, which includes five subcomponents such as accounting and billing, workload manager, resource provisoner, resource monitor and security manager.

   a) *Accounting and billing* module includes the information about expenses of cloud services, cost of ownership, user budget etc.

   b) *Workload Manager* manages the incoming workloads from the application manager and identifies the Quality of Service (QoS) requirement for every workload for their successful execution and transfer the QoS information of workload to the resource provisoner.

   c) *Resource provisoner* have three modules: SLA manager, VM manager and Fault manager. *SLA manager* module manages the official contract between user and provider in terms of QoS requirements. Based on the availability of VMs, *VM manager* provisions and schedules the cloud resources for workload execution based on QoS requirements of workload using physical machines or VMs. Fault manager keep tracks of system, detects the faults along with their causes and correct them without degradation of performance. Further, it finds the future faults and their impacts on the system's performance.

   d) *Resource monitor* keeps a continuous record of activities of underlying infrastructure to assure the availability of services. Moreover, it also monitors the QoS requirements of incoming workloads.

   e) *Security Manager* deploys the virtual network security policies to provide secure: 1) data transmission between cloud users and providers and 2) workload and VM migration between cloud datacenters.

3. *Physical Infrastructure:* This layer consists of cloud datacentres (which consists of multiple resources such as processors, network cards, storage devices and disk drives), which are used to execute cloud workloads. Based on the VM manager policy, VM migration or consolidation is performed for execution.

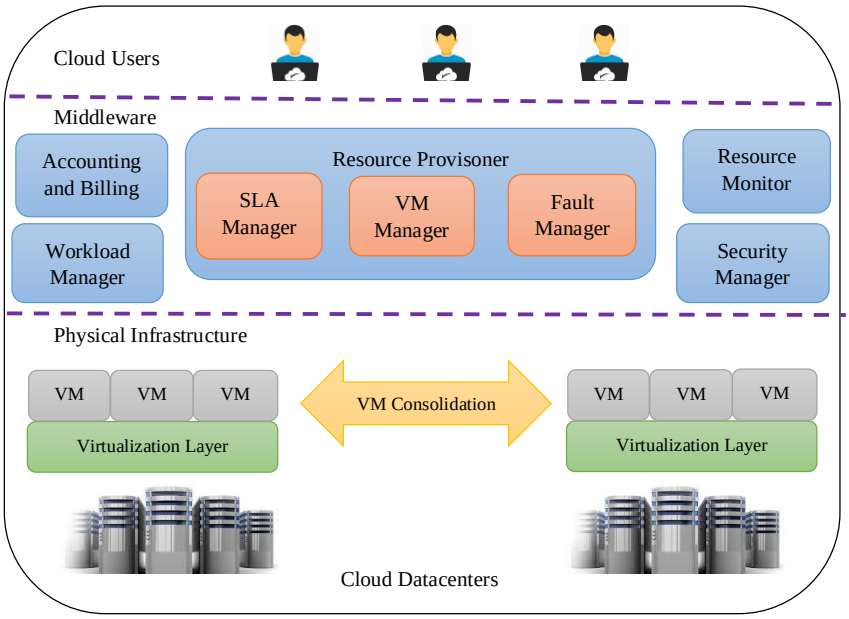

Figure 3: Conceptual Model for Reliable Cloud Computing

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/MCSE.2018.2873866, Computing in Science & Engineering

8. FAILURE MANAGEMENT FOR SCIENTIFIC COMPUTING APPLICATIONS

There are different areas such as astronomy, bioinformatics, genomics, quantum chemistry, life-sciences and high-energy physics represent their applications as scientific workflows. To obtain their scientific experimental results, these applications are executed using distributed systems [26]. These applications can be I/O or data or compute intensive applications, which have exponentially adopted cloud computing environments [25]. The workflow management systems use on-demand dynamic provisioning model to execute application on multi-cloud environment, which improves the fault tolerance in scientific workflow based applications [27]. The Cloudbus workflow management system execute applications on multiple clouds using dynamic provisioned resources.

8.1 Montage: A Case Study of Astronomy Workflow

This section presents the reliable execution of astronomy application on cloud environment to validate the conceptual model. Astronomy studies spiritual bodies and space through image datasets that cover a wide range of electromagnetic spectrum [27]. Further, astronomers use these images in different ways such as spatial samplings, pixel densities, image sizes and variety of map projections [25]. As astronomy application is expressed as workflow made up thousands of interrelated tasks; any failure in task execution as resources faults will have a cascading effect. Figure 4 shows the system architecture, which shows the interactions among different components for application execution and the need for handling failures explicitly. The system architecture comprises of following subcomponents:

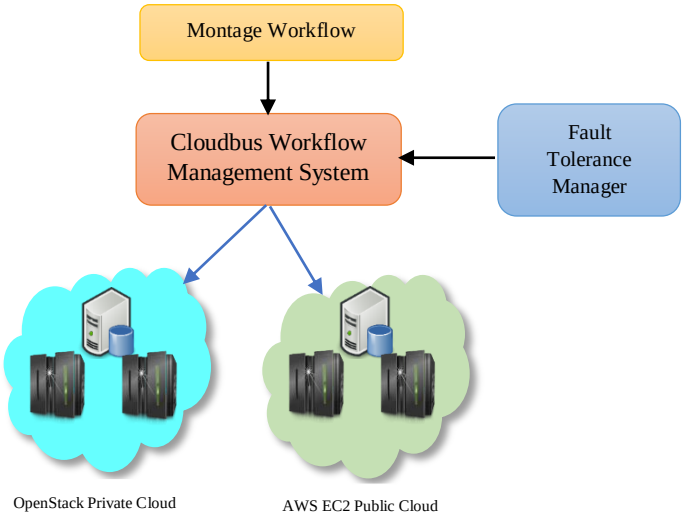

Figure 4: System Architecture

- *Montage Workflow:* Montage application is a complex astronomy workflow, which produces a mosaic of astronomic images.
- *Cloudbus Workflow Management System:* It uses decentralized scheduling architecture for workflow execution, which allows tasks to be scheduled by multiple schedulers.
- *Fault Tolerance Manager*: Two different types of fault tolerance techniques (retry and task replication) are used, which helps to mitigate failures during execution on distributed systems. *Retry* method reschedules a failed job to an available resource, while *task replication* method replicates a task on more than one resource.

In a demonstrated application, Melbourne CLOUDS Lab researchers [27] created a montage workflow consisting of 110 tasks, where the number of images used are represented by the number of tasks. Montage toolkit is used to process tasks that compute such mosaics through independent modules using simple executables. Workflow management systems requires three type of resources such as master node (hosted in the OpenStack private cloud), storage host (hosted in the AWS EC2 public cloud) and worker node (hosted in the AWS EC2 public cloud, which performs workflow execution). Resource failures was orchestrated to demonstrate the fault-tolerance of the workflow management system. The experimental results show that makespan (execution time) increases with the increase of the number of failures using retry fault-tolerant technique. After a resource fails, it remaps all tasks that where scheduled on the failed resource, thus saving execution time. The workflow makespan is higher as it schedules the resources on two cloud infrastructures because of data transfer time and the data movement time between tasks. Experimental results demonstrate that execution of an application using two cloud infrastructures would increase the time but will reduce the cost significantly than running the entire application on a public cloud. The interested readers can refer [27] for more details.

9. FUTURE RESEARCH DIRECTIONS

As discussed in Table 2, there are many open challenges in ensuring reliability of cloud computing services. To address them, we proposed the following directions that helps in practical realization of proposed conceptual model:

1. **Energy:** To provide a reliable cloud service, it is required to identify that how the occurrences of failures effect the energy efficiency of cloud computing system. Moreover, it is necessary to save the checkpoints with minimum overhead after predicting an occurrence of failure. Therefore, workloads or VMs can be migrated to more reliable servers, which can save the energy consumption and time. Further, consolidation the multiple independent instances (web service or email) of an application can improve the energy efficiency, which improves the availability of cloud service.
2. **Security:** Real cloud failure traces can be used to perform the empirical or statistical analysis about failures to test the performance in terms of the security of the system. Security during VM migration is also an important issue because a VM state can be hijacked during its migration. To solve this problem, there is a need of encrypted data transfer to stop user account hijacking, which can provide a secure communication between user and provider. To improve the reliability of cloud service to next level, homomorphic encryption methods can be used to provide security against malicious attacks like denial of service, password crack, data leakage, DNS spoofing and eavesdropping. Further, it is required to understand and address the causes of security threats such as VM level attacks, authentication and authorization and network-attack surface for efficient detection and prevention from cyber-attacks. Moreover, data leakage prevention applications can be used to secure data, which also improves the reliability of cloud computing system.
3. **Scalability:** The unplanned downtime can violate the SLA and effects the business of cloud providers. To solve this problem, a cloud computing system should incorporate dynamic scalability to fulfil the changing demand of users without the violation of SLA.
4. **Latency:** Virtualization overhead and resource contention are two main problems in computing systems, which increases the response time. Reliability-aware computing system can minimize the problems for real time applications such as video broadcast and video conference, which can reduce latency while transferring data.
5. **Data Management:** Computing systems are also facing a challenge of data synchronization because data is stored geographically, which overloads the cloud service. To solve this problem, rapid elasticity can be used to find the overloaded cloud service and it adds new instances to handle the current workloads. Further, there is a need of efficient data backup to recover the data in case of server downtime.
6. **Auditing:** To maintain the stable and health situation of the cloud service, there is a need of periodic auditing by third parties, which can improve the reliability and protection of computing system.

10. CONCLUSIONS

We proposed a taxonomy for identifying the research issues in reliable cloud computing. Further, the existing techniques of reliable cloud computing have been analysed based on the taxonomy of failure management. We have discussed the failure management in open source technologies and the fault tolerance resilience in practice for commercial clouds. Further, fault tolerance in modular microservices and the resilience on Exascale systems is discussed. We propose a conceptual model for effective management of resources to improve reliability of cloud services. Moreover, a case study of astronomy workflow is presented for reliable execution in cloud environment. Our study has helped to determine research gaps in reliable cloud computing as well as identifying future research directions.

ACKNOWLEDGEMENT

This work is supported by the Melbourne-Chindia Cloud Computing (MC3) Research Network and ARC (DP160102414).

REFERENCES

1. Sukhpal Singh and Inderveer Chana, “QoS-aware Autonomic Resource Management in Cloud Computing: A Systematic Review”, “ACM Computing Surveys”, Volume 48, Issue 3, pp. 1-46, 2016.
2. Gill, Sukhpal Singh, and Rajkumar Buyya. "SECURE: Self-Protection Approach in Cloud Resource Management." IEEE Cloud Computing 5, no. 1 (2018): 60-72.
3. Yi-Kuei Lin and Ping-Chen Chang. "Maintenance reliability estimation for a cloud computing network with nodes failure." Expert Systems with Applications 38, no. 11, 14185-14189, 2011.
4. Amir Vahid Dastjerdi and Rajkumar Buyya. "An autonomous reliability-aware negotiation strategy for cloud computing environments." In Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012), pp. 284-291. IEEE Computer Society, 2012.
5. Zhang Xuejie, Wang Zhijian and Xu Feng. "Reliability evaluation of cloud computing systems using hybrid methods." Intelligent automation & soft computing 19, no. 2, 165-174, 2013.
6. Abishi Chowdhury, and Priyanka Tripathi. "Enhancing cloud computing reliability using efficient scheduling by providing reliability as a service." In Parallel, Distributed and Grid Computing (PDGC), 2014 International Conference on, pp. 99-104. IEEE, 2014.
7. Nicola Cordeschi, Danilo Amendola, Mohammad Shojafar and Enzo Baccarelli. "Distributed and adaptive resource management in cloud-assisted cognitive radio vehicular networks with hard reliability guarantees." Vehicular Communications 2, no. 1, 1-12, 2015.
8. Ao Zhou, Shangguang Wang, Zibin Zheng, Ching-Hsien Hsu, Michael R. Lyu and Fangchun Yang. "On cloud service reliability enhancement with optimal resource usage." IEEE Transactions on Cloud Computing 4, no. 4, 452-466, 2016.
9. Mingqiang Li, Chuan Qin, Jingwei Li and Patrick PC Lee. "CDStore: Toward reliable, secure, and cost-efficient cloud storage via convergent dispersal." IEEE Internet Computing 20, no. 3, 45-53, 2016.
10. Fatemeh Azimzadeh and Fatemeh Biabani. "Multi-objective job scheduling algorithm in cloud computing based on reliability and time." In Web Research (ICWR), 2017 3rd International Conference on, pp. 96-101. IEEE, 2017.
11. Jing Deng, Scott C-H. Huang, Yunghsiang S. Han and Julia H. Deng. "Fault-tolerant and reliable computation in cloud computing." In GLOBECOM Workshops (GC Wkshps), 2010 IEEE, pp. 1601-1605. IEEE, 2010.

12. Sukhpal Singh and Inderveer Chana. "Q-aware: Quality of service based cloud resource provisioning." Computers & Electrical Engineering 47, 138-160, 2015.
13. Rodrigo N. Calheiros and Rajkumar Buyya, Meeting Deadlines of Scientific Workflows in Public Clouds with Tasks Replication, IEEE Transactions on Parallel and Distributed Systems (TPDS), Volume 25, Issue 7, Pages: 1787 - 1796, ISBN: 1045-9219, IEEE CS Press, Los Alamitos, CA, USA, July 2014.
14. Deepak Poola, Kotagiri Ramamohanarao, and Rajkumar Buyya, Enhancing Reliability of Workflow Execution Using Task Replication and Spot Instances, ACM Transactions on Autonomous and Adaptive Systems (TAAS), Volume 10, Number 4, Pages: 1-21, ACM Press, New York, USA, February 2016.
15. Chenhao Qu, Rodrigo N. Calheiros and Rajkumar Buyya, A Reliable and Cost-Efficient Auto-Scaling System for Web Applications Using Heterogeneous Spot Instances, Journal of Network and Computer Applications (JNCA), Volume 65, Pages: 167-180, Elsevier, Amsterdam, The Netherlands, April 2016.
16. Xunyun Liu, Aaron Harwood, Shanika Karunasekera, Benjamin Rubinstein and Rajkumar Buyya, E-Storm: Replication-based State Management in Distributed Stream Processing Systems, Proceedings of the 46th International Conference on Parallel Processing (ICPP 2017, IEEE CS Press, USA), Bristol, UK, August 14-17, 2017.
17. Sukhpal Singh, Inderveer Chana and Maninder Singh. "The Journey of QoS-Aware Autonomic Cloud Computing." IT Professional 19, no. 2, 42-49, 2017.
18. Gill, Sukhpal Singh, and Rajkumar Buyya. "A Taxonomy and Future Directions for Sustainable Cloud Computing: 360 Degree View." ACM Computing Surveys, 2018. URL: https://arxiv.org/ftp/arxiv/papers/1712/1712.02899.pdf
19. Singh, Sukhpal, and Inderveer Chana. "A survey on resource scheduling in cloud computing: Issues and challenges." Journal of grid computing 14, no. 2 (2016): 217-264.
20. Jadin, Mathieu, Gautier Tihon, Olivier Pereira, and Olivier Bonaventure. "Securing MultiPath TCP: Design & Implementation." In IEEE INFOCOM 2017. 2017.
21. Latiff, Muhammad Shafie Abd, Syed Hamid Hussain Madni, and Mohammed Abdullahi. "Fault tolerance aware scheduling technique for cloud computing environment using dynamic clustering algorithm." Neural Computing and Applications 29, no. 1 (2018): 279-293.
22. Jhawar, Ravi, and Vincenzo Piuri. "Fault tolerance and resilience in cloud computing environments." In Computer and Information Security Handbook (Third Edition), pp. 165-181. 2017.
23. Haselböck, Stefan, Rainer Weinreich, and Georg Buchgeher. "Decision guidance models for microservices: service discovery and fault tolerance." In Proceedings of the Fifth European Conference on the Engineering of Computer-Based Systems, p. 4. ACM, 2017.
24. Casanova, Henri, Frédéric Vivien, and Dounia Zaidouni. "Using replication for resilience on exascale systems." In Fault-Tolerance Techniques for High-Performance Computing, pp. 229-278. Springer, Cham, 2015.
25. Day, Charles. "Astronomical Images before the Internet." Computing in Science & Engineering 17, no. 6 (2015): 108-108.
26. Remmel, Hanna, Barbara Paech, Christian Engwer, and Peter Bastian. "A case study on a quality assurance process for a scientific framework." Computing in Science & Engineering 16, no. 3 (2014): 58-66.
27. Deepak Poola Chandrashekar, Robust and Fault-Tolerant Scheduling for Scientific Workflows in Cloud Computing Environments, Ph.D. Thesis, The University of Melbourne, Australia, August 2015.
28. Singh, Sukhpal, Inderveer Chana, and Rajkumar Buyya. "STAR: SLA-aware autonomic management of cloud resources." IEEE Transactions on Cloud Computing (2017).
29. Mahajan, Ajay, Munish Kumar Gupta, and Shyam Sundar. Cloud-Native Applications in Java: Build microservice-based cloud-native applications that dynamically scale. Packt Publishing Ltd, 2018.

ABOUT THE AUTHORS

Sukhpal Singh Gill is a Postdoctoral Research Fellow within the University of Melbourne's Cloud Computing and Distributed Systems (CLOUDS) Laboratory. Contact him at sukhpal.gill@unimelb.edu.au.

Rajkumar Buyya is a Redmond Barry Distinguished Professor and Director of the Cloud Computing and Distributed Systems (CLOUDS) Laboratory at the University of Melbourne, Australia. He is one of the most highly cited authors in computer science and software engineering worldwide. He was recognized as a "Web of Science Highly Cited Researcher" in both 2016 and 2017 by Thomson Reuters, a Fellow of IEEE, and Scopus Researcher of the Year 2017 with an Excellence in Innovative Research Award by Elsevier for his outstanding contributions to Cloud computing. Contact him at rbuyya@unimelb.edu.au.