Journal of Computer and System Sciences 79 (2013) 1243–1255

Contents lists available at SciVerse ScienceDirect

Journal of Computer and System Sciences

www.elsevier.com/locate/jcss

A survey on reliability in distributed systems

Waseem Ahmed *, Yong Wei Wu

Department of Computer Science and Technology, Tsinghua University, Beijing, China

ARTICLE INFO

Article history:
Received 11 November 2010
Received in revised form 15 April 2011
Accepted 22 February 2013
Available online 13 March 2013

Keywords:
Keyword
Reliability prediction
Assessment
Fault tolerant

ABSTRACT

Software's reliability in distributed systems has always been a major concern for all stake holders especially for application's vendors and its users. Various models have been produced to assess or predict reliability of large scale distributed applications including e-government, e-commerce, multimedia services, and end-to-end automotive solutions, but reliability issues with these systems still exists. Ensuring distributed system's reliability in turns requires examining reliability of each individual component or factors involved in enterprise distributed applications before predicting or assessing reliability of whole system, and Implementing transparent fault detection and fault recovery scheme to provide seamless interaction to end users. For this reason we have analyzed in detail existing reliability methodologies from viewpoint of examining reliability of individual component and explained why we still need a comprehensive reliability model for applications running in distributed system. In this paper we have described detailed technical overview of research done in recent years in analyzing and predicting reliability of large scale distributed applications in four parts. We first described some pragmatic requirements for highly reliable systems and highlighted significance and various issues of reliability in different computing environment such as Cloud Computing, Grid Computing, and Service Oriented Architecture. Then we elucidated certain possible factors and various challenges that are nontrivial for highly reliable distributed systems, including fault detection, recovery and removal through testing or various replication techniques. Later we scrutinize various research models which synthesize significant solutions to tackle possible factors and various challenges in predicting as well as measuring reliability of software applications in distributed systems. At the end of this paper we have discussed limitations of existing models and proposed future work for predicting and analyzing reliability of distributed applications in real environment in the light of our analysis.

1. Introduction

1.1. Overview

With significant advancement in the field of distributed environment in last few decades especially in providing software/hardware services to different kind of stake holders, has almost changed the computing environment with reference to data sharing, cycle sharing and other modes of interaction that involve distributed resources [29]. With cumulative number of online services or automotive end-to-end solutions and growing number of users using these services, requirements for highly reliable systems have become indispensable. Various surveys in highlighting or describing significance of reliability

* Corresponding author.
E-mail address: amw.inbox@gmail.com (W. Ahmed).

0022-0000/$ – see front matter
http://dx.doi.org/10.1016/j.jcss.2013.02.006

in enterprise applications have already been produced [15,16,20,38] discussing several fault tolerant techniques such as recovery, replication, etc. However highly reliable enterprise distributed systems are still a challenge for venders to achieve. This is the appropriate time to highlight missing part in previous papers for making highly reliable enterprise applications. In this paper we surveyed significant areas particularly highlighting reliability of individual components which needs researcher's attention for producing a generic model for predicting or measuring reliability of such systems and also tried to summarize various models produced in this regard and discussed their merits and demerits.

1.2. Pragmatic requirements

When it comes to reliability, efficiency, flexibility and extensibility users want highly reliable systems, because:

1. These software/hardware services have always been required to be efficient enough to compete with the current user demands in terms of quality of service and in increase of data volume.
2. Unreliable applications do not only shorten the age of application software but it also becomes a cause of massive loss both in terms of time and money for applications venders.
3. With the emergence of web based end-to-end applications, competition among organizations to provide better and reliable solutions have been substantially increased.
4. Customers with unreliable applications not only lose their users but also have to bear huge loss in their businesses.
5. Customers want their applications to be reliable all the time even with growing number of concurrent requests or massively increase in data.
6. Information security is also one of the major concerns for highly reliable systems for application providers and users when dealing with transactional systems over the internet, which is still an open issue for venders to secure their application from hackers. It is really important to safeguard against compromised data, denial of service attacks and other forms of intrusion [44]. Although sufficient techniques have been introduced even though still a comprehensive solution is required to provide end-to-end information security for distributed systems.
7. Resource sharing on large scale distributed systems may become a cause of various failures, such as timeout failures, blocking failures, program failures which need to be analyzed or predict for building highly reliable systems.
8. Coordination among various resources in a distributed environment have become complex due to heterogeneous nature of computing, data and network resources [45,44] because failure detection methods developed for current distributed systems have generally not been regarded as suitable for large-scale, heterogeneous, dynamic grid systems.

These evolutionary pressures have generated new requirements for companies to devise intelligent mechanism to analyze and predict distributed application's reliability at certain level before actual application deployment or before application start behaving abnormal to end users. This will certainly reduce re-engineering cost and save unexpected time by producing more reliable systems. Other than cost and time, customer's satisfaction in terms of system's reliability have also become one of the major concerns for companies, and they are investing lot of efforts to keep reliability of their applications up to the mark. We have analyzed several models for predicting or measuring reliability of distributed systems that can roughly be classified into User Centric, Architecture, State based models. These models were based on different measures such as Markov State [25], Factoring theorem [27], architectural level based approach [3,4,21], and users collaborative approach [3]. Although there are various models in this domain, nevertheless, they have certain deficiencies in their approaches. Such as they have considered the reliability of some important factors as constant (as mentioned in Table 3), like reliability of hardware. Furthermore failure data which they have collected for their experiments through in-house testing cannot be compared with failures that can occur under actual operational environment. We have discussed these models in terms of certain reliability factors later in this paper that assuming any of these factors as constant will affect reliability of the system, so we need to consider all factors for precise and accurate application's reliability. At the end we presented future work for predicting reliability of large scale systems in real environment considering real data.

2. Conceptual framework

In this section we have defined some basic concepts and highlighted various issues and challenges affecting reliability in distributed, grid, cloud and SOA based computing environments. On the basis of these basic concepts we then specified various reasons which are indispensable for a reliable distributed application. In the end of this section, we have mentioned certain factors affecting the reliability and elaborated the importance of hardware reliability with the help of bath tub curve.

2.1. Reliability

Software's reliability has always been a major concern for all stake holders (especially for system's vendors and system's users) and has been criticized over past few years. Among different other factors such as software extensibility, maintainability, and usability, etc., reliability has greater impact on software's life, because it can make the running application out of order. Reliability is a very broad term and any software application running in distributed environment can have various definitions [40]. Software application is said to be reliable if it can:

1. Perform well in specified time t without undergoing any halting state.
2. Perform exactly the way it is designed i.e. as per requirements.
3. Resist various failures and recover in case of any failure that occurs during system execution without producing any incorrect result.
4. Successfully run software operation or its intended functions for a specified period of time in a specified environment.
5. Have probability that a functional unit will perform its required function for a specified interval under stated conditions.
6. Have the ability to run correctly even after scaling is done with reference to some aspects.

So it is crucial to consider all above states for predicting reliability or providing a strong layer of tolerance in case of failure during live execution of any software application. In *software engineering* reliability can be defined as a probability of successfully running software operations or its intended functions for a specified period of time in a specified environment. For measuring and predicting reliability for large scale distributed systems it is necessary to fully scrutinize fundamentals of reliability. There can have four distinct dimensions to the above definition of reliability [39].

1. *Probability*: System shall meet specified probability of success, at a specified statistical confidence level.
2. *Intended function*: If a system is not providing proposed functionality without having any single failure, still it will be termed as unreliable system.
3. *Time dependent*: System shall perform well without having any single failure during specified time t or before specified time t.
4. *Specific conditions*: System shall work fine under specified conditions or in specific environment in terms of hardware or communication devices.

Existing papers [1–3,25,27] have been produced keeping these elements as a central theme of their models. Predicting reliability of any software application varies from one environment to another depending upon its type. From the last few decades, scope of enterprise applications in distributed environment have been substantially increased allowing venders to develop cross-platform individual components using different technologies that can interact transparently through a call-and-return mechanism. Measuring and predicting reliability of such applications is nontrivial because these systems serve various stake holders in their businesses or social life across the globe and also improves quality of various services of complex systems. In Table 1, we have mentioned various challenges for building large scale enterprise distributed applications in different computing environments.

2.2. Various challenges and factors affecting reliability

Emergence of various computing environments has changed the size and complexity of applications and problems from single PC to network of computers. These rapid changes have not only improved architecture of processors and network technologies but also brought various technical challenges for application venders in terms of developing highly reliable systems in distributed environment including asynchronism, heterogeneity, scalability, fault tolerance and failure management, security, etc.

One of the key issues for ensuring reliability of any enterprise level distributed applications is to understand variety of underlying technologies that can impact the reliability of such applications. For any enterprise level distributed applications we can divide the details of their architecture into two different levels both in terms of hardware and software i.e. Low level details and Broad level details as shown in Table 2.

Among various technologies for building enterprise distributed applications for instance Web technologies, IP technologies, DB or Web servers, file servers, authorization servers, clock synchronization, locking services, etc., in Fig. 1 we have shown various factors and highlighted importance of their reliability, and it is apparent from the figure that failure of any aspect can result in application level errors, inconsistency, denial of services, etc. Therefore it is necessary to consider each and every aspect to measure or predict reliability of these applications and must not ignore or consider as constant assuming it as a standard or reliable.

Performance, scalability, and reliability are important attributes of any enterprise distributed application [35], and to achieve performance, scalability, availability and reliability venders have used various techniques in different phases of Software Development Life Cycle (SDLC) including fault or failure management. In SDLC there exist following four fault life-cycle techniques [38].

Fault prevention: try eliminating errors during development phase.

Fault removal: Eliminating bugs or faults after having multiple testing phases.

Fault tolerance: To provide, by redundancy, service complying with the specification in spite of faults.

Fault/failure forecasting: Predicting or estimating faults at architectural level during design phase or before actual deployment.

Purpose of all types of fault life cycle techniques is to make systems reliable where first two types are typical fault life cycle techniques used in almost all sorts of software applications during SDLC; however fault tolerance and fault/failure forecasting have gained significant consideration as reliability attributes in industry and academia. To provide a seamless interaction to the end users, application venders have introduced various fault tolerance and recovery techniques to handle

Table 1
Various challenges in different computing environment.

Grid based applications	SOA based applications	Cloud based Applications
Grid resources are available across the world without any constraint of geographical boundary so there are immense chances that these geographically dispersed resources shall gain more priority to their local request over those which are coming across the network.	In Service Oriented Architecture (SOA) based complex distributed applications, predicting and measuring service reliability is comparatively difficult to that of traditional software applications because most of the times web services are owned and hosted by other organizations which may become unavailable quite easily because of unpredictable internet.	In cloud based distributed applications reliability attribute is very critical but hard to analyze due to its characteristics of massive-scale service sharing, wide-area network, heterogeneous software/hardware components and complicated interactions among them [22]. As in cloud system each application has different composition, configuration and deployment requirements so to analyze and predict reliability of different types of applications under different circumstances is really a challenging task.
Asynchronous nature of this environment, in which distributed components utilize independent clocks and messages which may be subjected to unbounded delay [16] that can lead to uncertainty in coordinating among various resources.	To conduct testing of remote services is not only time consuming but also an expensive activity to conduct testing of remote web services [3]. So it is very difficult to ensure the reliability of remote web service without undergoing through testing phase.	
In grid infrastructure resources are shared among multiple users, where waiting time of any single request can increase to an unacceptable value.	Messaging reliability in case of SOA based distributed applications is still a problem because developing cross-vendor and cross-platform inter-operability in Web services are using features beyond the basic Web Service Definition Language (WSDL) and Simple Object Access Protocol (SOAP) standards [43]. Although various techniques for ensuring reliable message delivery have been made available even though connections break, messages failed to get delivered or are delivered more than once or in the wrong sequence, and intercepted by hackers are still open issues [43] jeopardizing entire application.	
Grid has layered structure in heterogeneous components [17] i.e. Grid Fabric, Core grid middle ware, and User level grid middle ware. To predict or analyze reliability in grid environment, researchers must consider service reliability at each layer, as there may have different type of failures when these layered structure coordinates among each other for completion of job, for instance Blocking, Timeout, network, program failure, etc.		

Table 2
Various underlying technologies that can affect reliability of distributed applications.

	Hardware	Software
Low level details	Reliability of various components of computer systems such as CPU, Storage Devices, and Memory, etc.	Functionality/reliability of various services and its underlying OSI layered structure, application's methods, functions, routines, their response time, latency in networks, etc.
Broad level details	Reliability of server's availability and scalability (such as File Server, DB servers, Web servers, and email servers, etc.), communication infrastructure, and connecting devices.	Reliability of overall functionality of various features of distributed application, fault tolerant techniques and recovery techniques either at peak time when user load is at maximum or during normal execution.

certain hidden issues that can affect system's operations in real, such as [5,6,8–14]. These techniques intend to prevent dormant software faults from becoming active, and recover software operations from erroneous conditions by tolerating system level faults methodically [38]. They require systems to be designed and developed as fault tolerant systems i.e. detect faults, escalate it, and continue working without being losing any piece of information while repairing faults in off line mode. However, some mission critical and real time systems dealing with sensitive data cannot bear to have a downtime for a single moment, so it is nontrivial to predict or assess reliability of such systems well before time by examining reliability of individual components of any enterprise application before predicting or measuring reliability of whole system.

Potential factors that can influence reliability of any individual component or application system as mentioned in [37,42] are listed in Table 3. Almost all factors mentioned in Table 3 have significant affect on reliability of software applications.

Various factors mentioned in Table 3 such as 1, 2, 3, 6, and 7 can be analyzed in detail in different phases of software development life cycle (SDLC) as this phase structure provides a formal basis for control, so that risks whether in requirements, performance, or implementation could be determined and resolved timely. While some of above mentioned factors for instance testing environment, hardware (i.e. Processors, telecommunication/input, output/storage devices), program workload, interaction with external services have significant affect in predicting and measuring reliability of distributed applications early at design phase or before actual deployment of applications in real environment. And they can be used to measure or

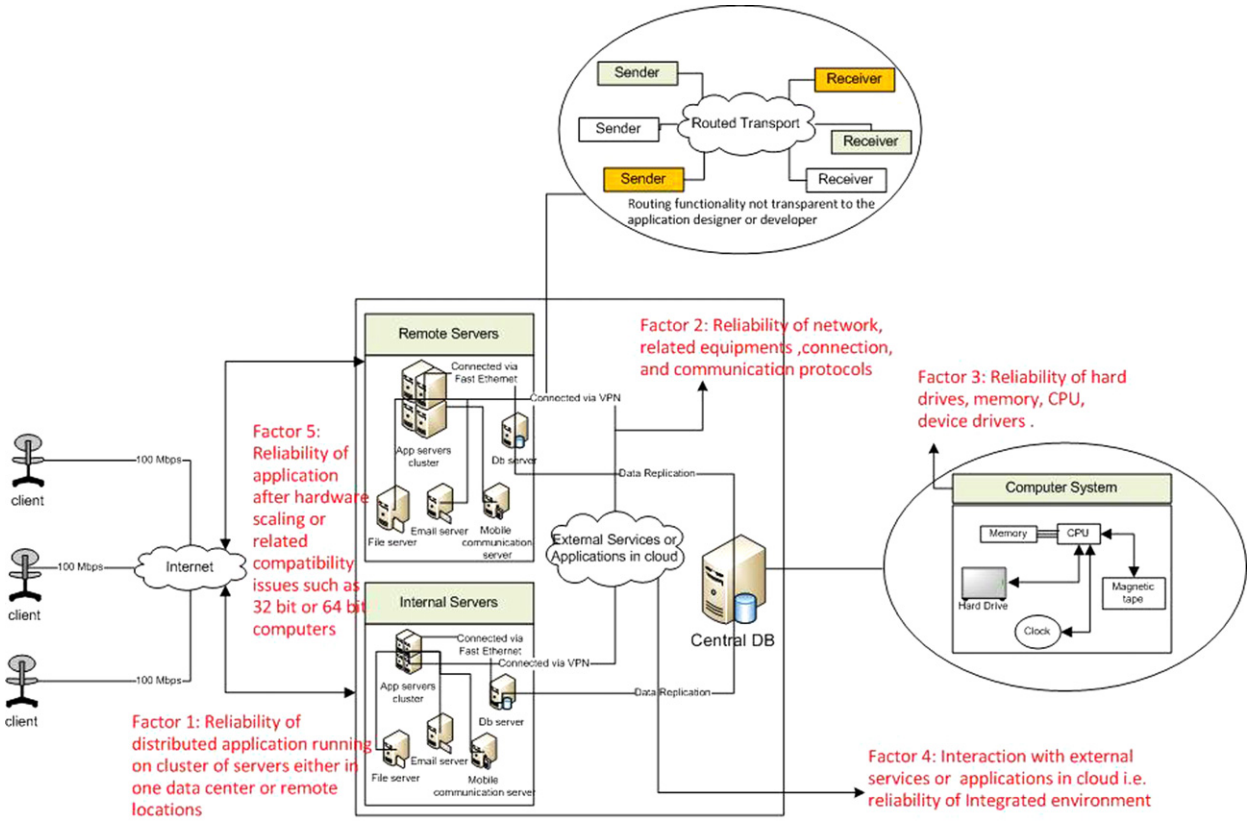

Fig. 1. Factors on which distributed applications may depend in order to provide correct, reliable behavior.

Table 3
Factors affecting software reliability.

Sr. No.	Factors effecting S/W reliability	Sr. No.	Factors effecting S/W reliability
1.	Inadequate testing.	6.	Issues of changing management.
2.	Operations errors.	7.	Low quality code source.
3.	Lack of a coherent process of quality assurance.	8.	Interactions with external services or applications.
4.	Different operating conditions – high levels of utilization and overload.	9.	Random events – security failures.
5.	Hardware failure – hard drives, network equipment, servers, power sources, memory, CPU.	10.	Issues related to the operational environment.

analyze software reliability by considering them in detail along with other essential parameters such as operational profile [2], past failure data [3]. Missing any one of these factors or considering their effect as a constant can lead to nebulous results.

Software reliability models usually make a number of common assumptions [38], such as. (1) The testing environment where reliability is to be measured or predicted is same as of the operational environment in which the reliability model shall be parameterized. (2) Failure data collected in in-house testing is supposed to represent failures of actual operational environment. (3) Reliability of hardware is considered to be constant nevertheless failure of hardware devices can cause certain contingent drivers hang or crash in a distributed environment [36] ultimately reducing reliability of running applications. However proper functioning of hardware along with the software is an important concern for millions of users using these systems. Characteristics of hardware failure represented by famous curve i.e. bath tub curve is shown in Fig. 2. As per this curve, chance of hardware failure approaches to maximum in the initial and end life of any specific hardware. At initial stage failures can occur because of any manufacturing fault where as in final stage after having useful life degradation of component characteristics will cause hardware modules to fail.

Based on such assumption models produced to predict reliability of distributed systems perhaps cannot give precise and accurate results. [37] showed that reliability of software applications is highly contingent to these factors so postulating them as constant or analogous or congruent to testing environment without considering life of hardware can lead us to unrealistic results.

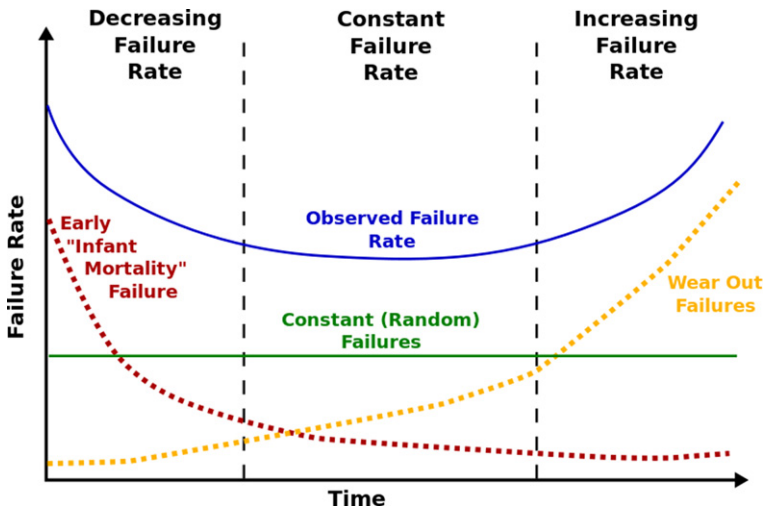

Fig. 2. An example bath tub curve showing characteristic of hardware failure.

3. Existing works

As described in Section 2 that purpose of all types of fault life cycle techniques [38] is to make systems reliable, so in this section we have further evaluated fault tolerance and fault prediction techniques in the light of various models and highlighted that how these models have used various methodologies to make distributed system more reliable. Approaches/models i.e. User centric, Architecture, State based approaches are used in accordance with the above mentioned reliability definitions.

3.1. Fault tolerant techniques

Fault tolerance mechanism operates as a backbone of distributed systems and is one of the important types of fault life-cycle techniques [38] which have an important role in the reliability of enterprise distributed applications. It deals with all those situations where failures or faults occurs during live execution of enterprise application and provides ability to the system to keep performing its function correctly. Based on various unanticipated situations in a distributed environment such as Service Oriented Architecture, Grid, and cloud environment, faults or failures can be classified into three categories i.e. Application level, System level, and Network level. To handle these faults timely, a fault tolerant system can be used either as spatial or temporal redundancy, including replication of hardware (with additional components), software (with special programs), and time (with the diversified of operations) [6]. Replication in terms of data has greater impact on the performance of large scale distributed systems. [45,28] have proposed an optimal solution for transparent data replication in a distributed environment, when data consists of multimedia objects. With significant advancement in the field of distributed computing, development of dynamic enterprise distributed applications have been substantially increased because of flexible architecture. Various fault tolerant techniques have been introduced to provide a reliable and seamless view to the end user. Below we have given an analysis of various models/strategies providing fault tolerance in different computing environment. We have also summarized possible shortcomings or open issues in these techniques.

Most of these techniques simply redirect the client's requests to backup servers in case of any failure without considering the current executing process. Requests which were in execution are effectively dropped and no mechanism has been defined to recover those transactions for seamless interaction. However [9] have specified a logging mechanism to monitor the state of executing processes and suggested modification in Linux kernel and apache web server to ensure correct handling of requests in process during failure time. However it has its own drawback as mentioned in Table 4.

3.2. Fault or failure forecasting techniques

We have analyzed several models in terms of various factors mentioned in Table 3 for predicting or measuring reliability of distributed systems that can roughly be classified into user centric based, architecture based, and state based models. Our analysis shows that almost all models based on these approaches deals with various factors such as different operating conditions – high level of utilization and overload, Hardware failure – (hard drives, network equipment, servers, power sources, memory, CPU), Interaction with external services or applications, random events – security failures and Issues related to the operational environment in predicting or measuring reliability of distributed systems. However two things are deficient almost in all of these models. 1) They did not consider all the factors (as mentioned in Table 3) collectively or considered some of them as constant. 2) They have considered hardware failure an important factor [37,42] as constant or ignored in measuring or predicting reliability of software application as shown in Fig. 3.

Table 4
Analysis of various models/strategies.

Papers	Strategy	Issues
[5]	This paper used both active and passive replication techniques along with business process specification and developed a fault tolerant scheme in web service orchestration. Based on three principle modules i.e. Composition and invocation of services, Failure detection and Fault tolerance and provides various replicas of same service.	[5,41,10,12,11] simply provides standby backup servers that can start processing new requests without maintaining services states. However for a real time distributed system dealing with financial transactions over the internet such as online banking, and e-commerce maintaining services states is still a problem to be solved. Solution designed in [41,10] for saving process's states is not feasible when a remote web service is involved where we cannot manage or track states of various processes inside the remote web service.
[41,10]	They suggested a logging mechanism, where errors are controlled by saving state of each process on stable storage and executing rollback and resuming from earlier checkpoints when an error is detected and to resume the process they have introduced Local and Global recovery mechanisms.	
[12]	This paper designed a model that suggests a client-transparent fault tolerant scheme for Web servers that ensures correct handling of requests in progress at the time of server failure.	
[6]	This paper designed a model that have used several replication schemes to increase system's availability, and identified various parameters that impact the web service dependability.	
[9]	They defined a model that suggested changes in Linux kernel and apache web server to provide implementation of a multi-cast mechanism allowing requests to be sent to a backup server and primary server.	Linux kernel or apache web server modification is itself an overhead in terms of reliability i.e Have to ensure reliability of modifications in kernel or web server before ensuring reliability of the applications. Model produced in [46] deals only with the selection of cloud, however it does not describe reliability parameters of any cloud based systems.
[11]	This paper used passive replication technique through notification mechanisms provided by the grid infrastructure. Replicas of same service were used to provide highly reliable or highly available grid service through primary-backup approach.	
[46]	It proposed a technique to improve fault tolerance and reliability against faulty or unreliable clouds and designed a model specifically to address the problem for selection of cloud environment to achieve better and reliable results.	

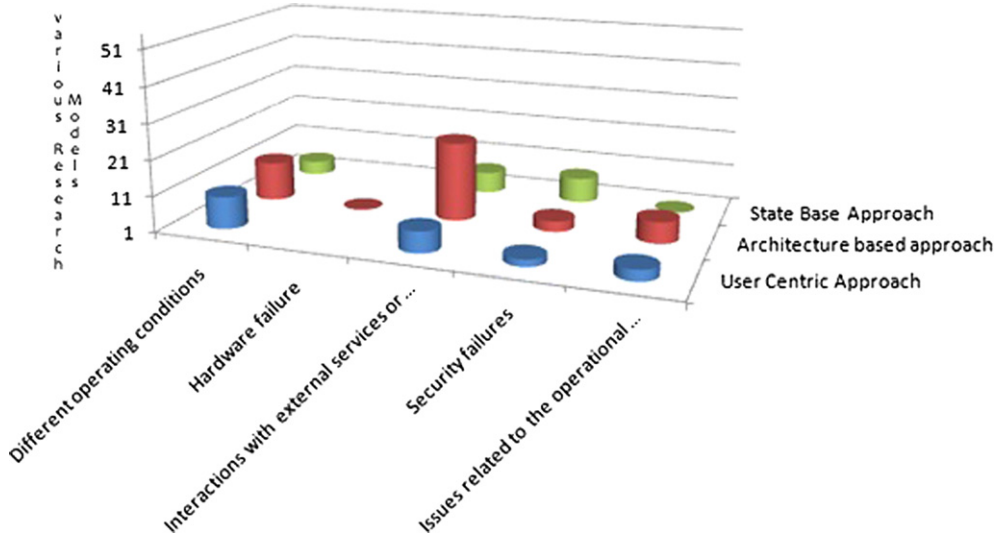

Fig. 3. Graph showing usage of various factors in different reliability prediction models.

3.2.1. User centric approaches

User centric approaches can be characterized as a multi stage problem solving processes where system is conceived in terms of user behavior. As reliability of any system has direct impact on the system usage so these models predict reliability considering all measures for both types of stake holders i.e. service users and service vendors. Assessing reliability of large scale systems at architectural level is a challenging task, because without having operational or usage profile it is difficult to know real impact of any given service(s) prior to its development [2]. So reliability of such software systems is heavily contingent to operational or usage profile of the given set of services. Time based model basically works on the principle of evaluating transmission time to compute execution time of each file or program under real conditions running in distributed environment [25]. But as systems behaves differently under different circumstances so various factors such as unpredictable

internet speed, network infrastructure having variable communication links intuitively affect the reliability performance of same set of services for different service users under different circumstances [3].

There are a lot of variations produced in predicting performance reliability under time based models and at architecture level prediction models. There are some other measurements made in this regard, which can be termed as user centric approaches such as [19,26] and Grid based systems [17]. These approaches elucidate various reliability issues both from end user point of view as well as service provider point of view and discussed how composition of services or grid environment affects reliability of applications. WS-DREAM model [26] is based on collaborative assessment mechanism to assess reliability in SOA based applications by providing real time test environment for testing service reliability to various users on different geographical dispersed locations. Here all users can share their results on a central server by making it easier to assess desired service in a distributed environment. This type of approach is normally termed as Black Box Testing approach where users are concerned with end result only. As they have provided only a testing environment over the internet ignoring various essential factors such as uncertainty in communication link and variable cumulative number of users or with different infrastructure at data center. Therefore it cannot be compared with real time environment to extrapolate service reliability. In congruent to WS-DREAM model, [19] also presented user centric approach for SOA based distributed application and highlighted dual behavior of services for different users and proved that reliability of SOA based distributed application reduces from vender's perspective with increasing number of services in composition however increases if consider the same structure is analyzed from end user point of view.

Grid environment is based on a layered structured in heterogeneous components [17]. Each layer works under its own jurisdiction for instance Grid Fabric normally deals with worker nodes, operating systems, local schedulers or network links, Core Grid Middle Ware deals with job submission, resource discovery and monitoring, or data management services, and User Level Grid Middle Ware is responsible for meta-schedulers, resource brokers or Grid portals. [17] presented a procedure for evaluating reliability of grid system and focused only on User Level Grid Middle Ware from user's point of view. As per their analysis reasonable level of reliability for end user can be attained through Grid-Way meta-scheduler over any Globus-based infrastructure.

3.2.2. Architecture based approaches

Predicting reliability at design phase in a service oriented architecture (SOA) [1–3,21] has not only greater impact on the quality of software application but it also helps in reducing re-engineering cost by providing more reliable software applications. In case of Service Oriented Architecture [3] used a user-collaborative failure data sharing approach to predict reliability for similar set of users and services based on their past experience. To find similar set of users and services Pearson Correlation Coefficient (PCC) is used which has already been recommended by various systems. This approach seems quite convincing but yet has some technical issues with it because apart from similar services and similar service users, similar infrastructure can also be a critical aspect in determining reliability performance based on old data. As system behavior is represented by system low level functionality and reliability of system's low level functionality will contribute to over all system's reliability [18]. Therefore similar user's experience for similar set of services is not sufficient in predicting reliability for similar set of services.

A lot of researchers are of the view that service providers must provide some other detail to compute reliability of both type of services i.e. atomic service and composite service. Such as, external services it uses, how service are glued together in composite service, how frequently they call each other, and flow graph describing behavior of service [2]. Service Oriented Reliability Model (SORM) [18] computed reliability of atomic and composite services exploiting distinct technique. It used highly efficient group testing to evaluate reliability of atomic service, based on majority voting and evaluated the reliability of composite service using architecture based model, reliability of each atomic service, execution scenarios and operational profiles.

In *Service Oriented Architecture* (*SOA*) based distributed applications services can be owned and hosted by different organizations. Developing distributed systems by integrating these services and predicting or analyzing reliability of such systems is a challenging task. Reliability prediction methodology presented in [1] analyzes impact on reliability of such services when they are assembled by:

1. Explicitly understanding dependency between input parameters of services and input parameters of cascading services.
2. Impact of service sharing.

One of the superseding features of this approach over others is exploitation of characteristics of both high level and low level services. It not only deals with the assembled services but also considers services offered by software component and the services offered by communication devices. However they assumed that partial information about reliability of remote web service must be published with each service, which is the most important aspect in SOA based applications.

ATOP (Activity diagram to PRISM) [21] is a probabilistic model driven approach that also support early Quality of Service assessment of a service composition at architectural level during design phase and automatically transforms a design model of service composition into an analysis model, which then feeds into the probabilistic model checker such as PRISM, for reliability checking. It starts from high level description of service composition that automatically derives random model which can be solve with different features of PRISM model checker. However this model highly depends on the results of

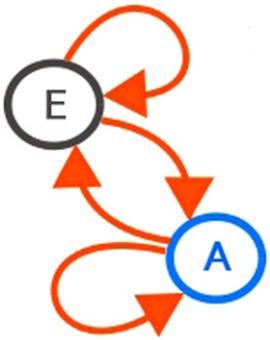

Fig. 4. Markov chain process.

PRISM probabilistic model used for checking reliability of external services and on the specification of external service when used in composition i.e. their functional and non-functional attributes.

3.2.3. State based approaches

Famous Markov chain process is a discrete random process that undergoes transitions from one state to another, as shown in Fig. 4, in a chainlike manner [30] to map layers into different physical states where every next state depends on the current state and not on entire system. Various models have used Markov chain process to analyze or predict reliability of distributed applications by computing time frame for each job in execution to determine whether selected task has successfully completed its job in specified time frame or not.

Reliability of entire application is highly dependent on the reliability of each of its individual component because failure of any component affects the reliability of other. To ignore any of the application's artifacts during reliability prediction can be fatal; any single failure can become a cause of failure of service. To deal with all sorts of errors such as time-out failures, blocking failures, network failures, etc., which can occur during certain operation of service invocation [23] described a hierarchy model. This hierarchal model suggests tackling various errors in different layers and used Markov state principle to map layers into different physical states. Failure occurrence during certain operation makes grid services unreliable [23]. This model used Markov models, queuing theory, graph theory, and Bayesian analysis to predict grid reliability performance by dealing with blocking, time-out, matchmaking, network, program and resource failure in hierarchical manner. Errors arising in different layers can be predicted by specifying certain pre-define criteria such as, in case of request layer, at any state, number of requests larger than length of request queue makes request overflow and as per this model steady probability of system at that state can be solved by the Chapman–Kolmogorov Equation. Systems with time constraints are critical because they have to complete their assigned task or job within defined time frame. Calculating time frame for each job in execution, network time and processing time of the job are key indicators which can be used. Transmission time based model [25] also work on the same principle of evaluating transmission time to compute execution time of each file or program under real conditions running in distributed environment, then on the basis of this time-constraint information and execution time of every program or file in distributed environment, corresponding Markov State can be define for reliability computation.

Communication time and processing time in grid computing environment are the important factors to analyze grid's reliability. During data transmission between two different nodes in a grid environment will be considered as failed if any failure occurs either on nodes or in the link between those nodes. Similarly program running on some node will be considered as failed if any failure occurs at that node. Algorithm defined for generating MFST [25] cannot be used to compute Minimum Resource Spanning Tree (MRST) as it do not consider one factor i.e., working time [24]. Model suggested in [24] searches MRSTs by taking care of all the factors involved such as Resource, Element, Working Time, Node, and Link. It is based on conditional probability mechanism to compute grid program reliability i.e. MRST-1 fails provided that MRST-2 is operational and suggests probability of intersections of the set of MRST's of each program because any one of intersected MRSTs to be operational can guarantee all computing programs in grid computing system.

Although substantial progress in grid computing system has already been done but still scalable reliability solutions for grid infrastructure is yet to be solved. Different models developed for predicting reliability of functional areas in grid resources are basically for developing methods for a fault tolerant system i.e. Detecting fault and failure in grid resources before its implementation and recovery to allow computation to continue without being crashed [16]. Certain limitations in existing models are analyzed with reference to the above functional areas in detecting fault in distributed environments for instance: failure detection mechanism developed for grid systems have not regarded as suitable for large scale, heterogeneous, dynamic grid systems [16] similarly it is impossible for a group of distributed failure detector to reach consensus on what resources have failed, if any component involved in the computation process also fails [31], etc. [16] highlighted and suggested some key specifications which need to be further analyzed in details to make grid system more reliable or fault tolerant. Key specification includes Architectural Approach, Quantitative Approach, and Detecting Individual component behavior.

After Intensive study in the field of virtualization, grid computing, and service oriented architecture (SOA), concept of cloud computing was evolved. Reliability analysis/modeling and performance indicators of cloud based distributed applications is challenging because these applications are not congruent to those of distributed computing based system or grid and SOA based applications in terms of their composition, configuration and deployment requirements. Apart from various issues in cloud based systems for instance:

1. How to collect provenance information in a standardized and seamless way.
2. How to present this information to the user in a logical manner i.e. an intuitive user web interface [32].

There exist some other factors such as reliability and availability of various cloud components and services. Failure of any resource in cloud system can collapse the whole application running under that environment. [33] presented a detailed analysis of hardware failure characteristics as well as preliminarily analysis on hardware failure predictors. [34] presented trust evaluation model for efficient reconfiguration and allocation of various cloud resources to several users for their multiple requests. Trust evaluation model is based on historical information collected from several servers' log in cloud environment to predict best available resources depending upon different user's requests in future, which can help cloud providers to utilize certain cloud resources efficiently by predicting uncertain behavior of multiple users' requests.

In cloud environment, it cannot simply utilize only one single model to predict software reliability, such as Network reliability, hardware reliability, software reliability because they are interrelated to each other, one can cause failure of other effecting overall system's reliability [22]. For system working with less reliable parts, reliability becomes a performance issue since reliability and recovery techniques have performance impact [16]. [22] modeled a two stage approach i.e. Request Stage and Execution Stage, to predict reliability of cloud based systems. As reliability somehow refers as successful execution of service even in the presence of faults so this two stage approach categorized various errors into their respective stage based on their nature, such as, those related to communication, database, hardware, or software, etc., are handled in execution stage and while others like timeout or overflow errors are handled at request stage. For a cloud service to be reliable, both request stage and execution stage need to successfully complete desired service in specific time limit under various circumstances.

4. Discussion and limitations of existing models

In this paper we have analyzed several research models to investigate distributed application's reliability and resilience to manage faults in various environments for instance SOA, Grid and Cloud. Analysis of these models revealed that reliability can be ensured either by:

1. Predicting reliability early at architectural level to better analyze the system before actual application development.
2. Providing a fault tolerant system to ensure seamless environment for end users in case if any unanticipated or unpredicted scenario has occurred during execution in real environment.

We have discovered that for SOA based distributed application, above fault tolerant techniques are based on various replication strategies including load balancer and data replications, most of these techniques simply redirects the client requests to backup servers in case of any failure, however for a real time distributed system dealing with financial transactions over the internet such as online banking, e-commerce, etc., we still have some open issues such as:

1. Maintaining services states is still a problem to be solved where sometime requests that's processing is in progress does not recover after failure occurrence. Still a lot of work is required to deal with such updates where updates were in process when failure had occurred or when various updates were issued simultaneously.
2. Situations where network problems may become a cause of system failure as even in modern network that consist of range of technologies and configurations cannot guarantee that messages shall get through in a timely manner or not subject to occasional disruptive events [44].
3. Services are sometime hosted by some other organization [3], it is therefore necessary to provide a strong layer of fault detection and recovery in case of failure of these services and where it is impossible to maintain transaction log for cascading service calls.

Several models has discussed maintaining transaction logs on backup servers to recover systems from same point of failure providing seamless interaction to end users, nevertheless these techniques have their own overheads of maintaining logs, modifying Linux kernel programs or apache web server's modules and utilizing various backup servers logging every single transactions either among services or between client and server. However, in case of remote web service, existing fault tolerant methods have tackled various failures by replicating same set of services on different geographically separated servers. For ensuring reliability in grid systems various fault tolerant techniques have been proposed focusing on various functional areas such as hardware, software, workflow services, networks, etc., of grid systems nevertheless still few areas need more attention for making grid system fault tolerant. Too much work has been done in developing metrics for grid networks, however:

1. Less focus is made in the area of developing metrics for other functional areas of grid environment and therefore without providing solution for various areas of grid computing collectively, it is hard to provide mechanism to measure reliability of whole grid environment.
2. Various models for fault tolerance in grid system have been proposed or experimented only for small scale grids however these models/techniques for large scale heterogeneous system still need to be further explored.
3. Limited data of testing environment was used to measure various existing reliability models for grid environment however various areas for scalable, heterogeneous, and dynamic grid systems based on geographically separated servers, need to be further tested with bulk data under real environment.

For cloud based enterprise applications users have shown serious concerns over application's reliability because their data, applications and computing resources will no longer be under their control. Although it have made significant improvement in processing large volumetric data using various commodity servers even though it causes serious problems when unexpected downtime occur either in operating system, Software, Hardware, or Networks, etc.

1. A lot of work is required to be done to provide reliable cloud services by allowing different alternatives in case downtime and outages occur in cloud services.
2. To overcome the issues of interoperability among various cloud applications although Cloud Computing Interoperability Forum has been formed [47] however this area needs to be further investigated from scalable and heterogeneous perspective.
3. For providing reliable scalable storage in cloud environment less work has been done to differentiate various scalability techniques for instance horizontal and vertical scalability in terms of their features and cost, more work is required to be done for ensuring reliability for scalable storage in cloud environment with minimum cost.

For predicting reliability of enterprise distributed application at early stages not only reduces re-engineering cost, time, and effort but also help in making applications reliable. And therefore it is necessary to examine reliability of all factors affecting over all systems reliability, however various techniques available did not discuss all the factors (as mentioned in Table 3) collectively. Approaches/models discussed above in terms of determining reliability of various factors can roughly be classified into User Centric, Architecture, State based models. As reliability can be considered as probability [40] which means it can be termed as random phenomenon so it is really difficult and challenging to measure or predict reliability of any scalable, heterogeneous distributed application in a precise and accurate manner and because of unpredictable internet, companies require that reliability of these applications must be assessed and predict continuously. However it is not intricate to provide adequate reliability measure for any highly reliable large scale enterprise application. Models produced in this regards tried to focus certain factors effecting reliability such as CPU processing time, network latency, interaction with external services in service orchestration, etc., however at early stage of any distributed application in execution, certain factors such as User Load, CPU Load, and Network Traffic has significantly less effect on over all reliability of application, which progressively increases. Therefore for predicting reliability:

1. It is necessary to consider all factors depending upon of the affect of their usage on system's reliability.
2. A lot of work is required to be done in combination with other reliability factors to check reliability of network topologies; compatibility of communication protocols such as OSI layered architecture with protocol used in the development of distributed application that might be some sort of operating system service.
3. As the most commonly used layer by various developers during development of distributed application in network protocol is communication layer which normally hides the properties of communication hardware, which can result in sending or receiving large messages normally not supported by underlying hardware. This can become a cause of failure of certain module of the running application due to failure of message delivery caused by buffering space. Although there are reliable communication channels that provide a layer over message reliability protocol and that can provide guarantee of messages delivery even though reliability of communication channel needs to be considered along with other factors, at the time of predicting reliability of whole system.

5. Conclusion and future work

As large scale projects such as e-commerce, e-government, end-to-end automotive systems, multimedia services [3], etc., are highly critical in terms of their reliability, therefore it is much more important to establish strong/reliable and fault tolerant architecture before actually start doing their development. The aim of this paper was to analyze and discuss different models presented for measuring and predicting reliability and to provide a comprehensive fault tolerant system for unanticipated or unpredicted issues in distributed/cloud/grid or SOA based enterprise applications. How these models handled different aspects effecting reliability and presented solutions/models based on their experiments. The common goal of all these models was to provide a seamless environment to end users either by predicting reliability early at design phase or provide a comprehensive fault tolerant system allowing application to keep performing well even in case of fault or failure. We analyzed all these models by studying reliability of various factors affecting reliability of whole system and concluded that existing models have highlighted significant areas and produced various models to improve system's reliability however

all these models did not discuss various factors collectively in predicting or measuring reliability of whole system and also made certain assumptions to prove their models. These models suggested different solutions on the basis of their experiments based on either old data of similar applications [3] or on some simulations results [7] or on building some testing environment similar to real environment to test their solutions. However,

1. Because of unpredictable internet, companies require that reliability of enterprise distributed applications must be assessed and predict continuously.
2. At early stage of any distributed application in execution, certain factors such as User Load, CPU Load, and Network Traffic has significantly less effect on over all reliability of application, which progressively increased requiring for comprehensive fault tolerant technique.

So keeping these arguments in consideration we must provide mechanism to assess or predict N-step reliability prediction of distributed applications in real environment under live circumstances considering all factors.

References

[1] R. de Lemos, et al., Architecture-based reliability prediction for service-oriented computing, in: Architecting Dependable Systems, III, Springer-Verlag, Berlin, Heidelberg, 2005, pp. 279–299.
[2] Vittorio Cortellessa, Vincenzo Grassi, Reliability modeling and analysis of service-oriented architectures, in: Test and Analysis of Web Services, 2007, pp. 339–362.
[3] Z. Zibin, R.L. Michael, Collaborative reliability prediction of service-oriented systems, in: Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering, vol. 1, ACM, Cape Town, South Africa, 2010.
[4] C. Leslie, et al., Early prediction of software component reliability, in: Proceedings of the 30th International Conference on Software Engineering, ACM, Leipzig, Germany, 2008.
[5] Jim Lau, Lau Cheuk Lung, J. da Fraga, G.S. Veronese, Designing fault tolerant web services using BPEL, in: Seventh IEEE/ACIS International Conference on Computer and Information Science, ICIS 08, 14–16 May 2008, pp. 618–623.
[6] Pat Pik-Wah Chan, Michael R. Lyu, Miroslaw Malek, Making services fault tolerant, in: Proceedings of the Third International Conference on Service Availability, Springer-Verlag, Helsinki, Finland, 2006.
[7] S. Chandrasekaran, et al., Performance analysis and simulation of composite web services, Electronic Markets 13 (2) (2003) 120–132.
[8] G.T. Santos, L. Lau Cheuk, C. Montez, FTWeb: A fault tolerant infrastructure for web services, in: Proceedings of the Ninth IEEE International EDOC Enterprise Computing Conference, IEEE Computer Society, 2005.
[9] N. Aghdaie, Y. Tamir, Implementation and evaluation of transparent fault-tolerant web service with kernel-level support, in: Proceedings of the IEEE International Conference on Computer Communications and Networks, Miami, Florida, 2002, pp. 63–68.
[10] V. Dialani, S. Miles, L. Moreau, D. De Roure, M. Luck, Transparent fault tolerance for web services based architectures, in: 8th International Europar Conference (EURO-PAR'02), Paderborn, Germany, 27–30 August 2002, pp. 889–898,.
[11] Xianan Zhang, D. Zagorodnov, M. Hiltunen, K. Marzullo, R.D. Schlichting, Fault-tolerant grid services using primary-backup: feasibility and performance, in: 2004 IEEE International Conference on Cluster Computing, 20–23 Sept. 2004, pp. 105–114.
[12] N. Aghdaie, Y. Tamir, Client-transparent fault-tolerant Web service, in: IEEE International Conference on Performance, Computing, and Communications, Apr. 2001, pp. 209–216.
[13] Frølund Svend, G. Rachid, Implementing e-transactions with asynchronous replication, IEEE Trans. Parallel Distrib. Syst. 12 (2) (2001) 133–146.
[14] Soonwook Hwang, C. Kesselman, Grid workflow: a flexible failure handling framework for the grid, in: Proceedings of the 12th IEEE International Symposium on High Performance Distributed Computing, 22–24 June 2003, pp. 126–137.
[15] S.O. Olabiyisi, et al., A survey of performance evaluation models for distributed software system architecture, in: Proceedings of the World Congress on Engineering and Computer Science, WCECS 2010, vol. 1, October 20–22, 2010, San Francisco, USA.
[16] D. Christopher, Reliability in grid computing systems, Concurr. Comput. Pract. Exper. 21 (8) (2009) 927–959.
[17] H. Eduardo, et al., Evaluating the reliability of computational grids from the end user's point of view, J. Syst. Archit. 52 (12) (2006) 727–736.
[18] W.T. Tsai, D. Zhang, Y. Chen, H. Huang, R. Paul, N. Liao, A software reliability model for web services, in: The 8th IASTED International Conference on Software Engineering and Applications, Cambridge, MA, November 2004, pp. 144–149.
[19] W. Abramowicz, M. Kaczmarek, D. Zyskowski, Duality in web services reliability, Guadeloupe, French Caribbean, 2006.
[20] A. Immonen, E. Niemel, Survey of reliability and availability prediction methods from the viewpoint of software architecture, Softw. Syst. Model. 7 (1) (2008) 49–65.
[21] G. Stefano, et al., Quality prediction of service compositions through probabilistic model checking, in: Proceedings of the 4th International Conference on Quality of Software-Architectures: Models and Architectures, Springer-Verlag, Karlsruhe, Germany, 2008.
[22] Y.S. Dai, B. Yang, J. Dongarra, G. Zhang, Cloud service reliability: Modeling and analysis, in: PRDC, 2009.
[23] Y. Dai, Y. Pan, X. Zou, A hierarchical modeling and analysis for grid service reliability, IEEE Trans. Comput. 56 (5) (2007) 681–691.
[24] Y.S. Dai, M. Xie, K.L. Poh, Reliability analysis of grid computing systems, in: Proceedings of the 2002 Pacific Rim International Symposium on Dependable Computing, 16–18 Dec. 2002, pp. 97–104.
[25] D.-J. Chen, M.-C. Sheng, M.-S. Horng, Real-time distributed program reliability analysis, in: Proceedings of the Fifth IEEE Symposium on Parallel and Distributed Processing, 1–4 Dec. 1993, pp. 771–778.
[26] Zibin Zheng, M.R. Lyu, WS-DREAM: A distributed reliability assessment mechanism for Web services, in: IEEE International Conference on Dependable Systems and Networks with FTCS and DCC, DSN 2008, 24–27 June 2008, pp. 392–397.
[27] D.J. Chen, M.S. Lin, On distributed computing systems reliability analysis under program execution constraints, IEEE Trans. Comput. 15 (12) (1993).
[28] L. Keqiu, et al., An effective cache replacement algorithm in transcoding-enabled proxies, J. Supercomput. 35 (2) (2006) 165–184.
[29] I. Foster, C. Kesselman, J.M. Nick, S. Tuecke, Grid services for distributed system integration, Computer 35 (6) (2002) 37–46.
[30] U.N. Bhat, Elements of Applied Stochastic Processes, 2nd ed., John Wiley, New York, 1984.
[31] J.F. Michael, A.L. Nancy, S.P. Michael, Impossibility of distributed consensus with one faulty process, J. ACM 32 (2) (1985) 374–382.
[32] M.A. Vouk, Cloud computing — Issues, research and implementations, in: 30th International Conference on Information Technology Interfaces, ITI 2008, 23–26 June 2008, pp. 31–40.
[33] K.V. Vishwanath, N. Nagappan, Characterizing cloud computing hardware reliability, in: ACM SoCC, 2010.
[34] Hyukho Kim, Hana Lee, Woongsup Kim, Yangwoo Kim, Internat. J. Grid Distrib. Comput. 3 (2010).
[35] http://msdn.microsoft.com/en-us/library/ff648802.aspx.

[36] K. Asim, J.R. Matthew, M.S. Michael, Tolerating hardware device failures in software, in: Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles, ACM, Big Sky, Montana, USA, 2009.
[37] Z. Xuemei, P. Hoang, An analysis of factors affecting software reliability, J. Syst. Softw. 50 (1) (2000) 43–56.
[38] M.R. Lyu, Software reliability engineering: A roadmap, in: Future of Software Engineering, FOSE '07, 23–25 May 2007, 153–170.
[39] R. Austen, H. Tracy, A quantitative and qualitative analysis of factors affecting software processes, J. Syst. Softw. 66 (1) (2003) 7–21.
[40] http://en.wikipedia.org/wiki/Reliability_engineering.
[41] Florin Popentiu Vladicescu, Pierre Sens, Software architecture for monitoring the reliability in distributed computing.
[42] M. Cristescu, L. Ciovica, Estimation of the reliability of distributed applications, Inform. Econ. 14 (2010) 19–29.
[43] L. O'Brien, L. Bass, P. Merson, Quality Attributes and Service-Oriented Architectures, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, 2005.
[44] http://Springerlink.com/content/xk081t/?p=5401588a43d24524a79928ed843c1a17&pi=0#section=535383&page=21_&locus_=61.
[45] Keqiu Li, Hong Shen, F.Y.L. Chin, Weishi Zhang, Multimedia object placement for transparent data replication, IEEE Trans. Parallel Distrib. Syst. 18 (2) (2007) 212–224.
[46] Jing Deng, S.C.-H. Huang, Y.S. Han, J.H. Deng, Fault-tolerant and reliable computation in cloud computing, in: 2010 IEEE GLOBECOM Workshops (GC Wkshps), 6–10 Dec. 2010, pp. 1601–1605.
[47] B.P. Rimal, Eunmi Choi, I. Lumb, A taxonomy and survey of cloud computing systems, in: Fifth International Joint Conference on INC, IMS and IDC, NCM '09, 25–27 Aug. 2009, pp. 44–51.